INFORMATION AND CONTROL 58, 113–136 (1983)

Parallel Computation for Well-Endowed Rings and Space-Bounded Probabilistic Machines

A. BORODIN AND S. COOK

Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

AND

N. PIPPENGER

IBM Corporation, 5600 Cottle Road, San Jose, California 95193

It is shown that a probabilistic Turing acceptor or transducer running within space bound S can be simulated by a time S^2 parallel machine and therefore by a space S^2 deterministic machine. (Previous simulations ran in space S^6.) In order to achieve these simulations, known algorithms are extended for the computation of determinants in small arithmetic parallel time to computations having small Boolean parallel time, and this development is applied to computing the completion of stochastic matrices. The method introduces a generalization of the ring of integers, called *well-endowed* rings. Such rings possess a very efficient parallel implementation of the basic $(+, -, \times)$ ring operations.

1. INTRODUCTION

Gill (1977) introduced the concept of language acceptance by space-bounded probabilistic Turing machines, and showed that such machines with space bound S could be simulated by deterministic machines with space bound $\exp O(S)$. Simon (1981a) improved this result to achieve a simulation within deterministic space bound S^6. In this paper, we shall again improve this result so as to achieve a simulation within deterministic space bound S^2. Our result can then be viewed as a generalization of Savitch's (1970) simulation of space-bounded nondeterministic machines, since nondeterministic computations can be viewed as a special case of Gill's probabilistic computations. It is, of course, a long standing open problem as to whether or not Savitch's result can be improved.

Our deterministic space bound S^2 also applies to the simulation of probabilistic Turing transducers bounded by space S. The previous best deterministic simulation bounds for such transducers were first space S^{36},

0019-9958/83 $3.00

due to Gill, Hunt, and Simon (1980), later improved to space S^6 by Hunt (1978).

We choose to carry out our development within the context of parallel computation (by uniform families of Boolean circuits), the relation between parallel computation and space-bounded computation having already been established (see, e.g., Borodin (1977) for parallel computation by Boolean circuits). We only need to note that a function computable in depth S can be computed in deterministic space S, but apparently not conversely. Pippenger (1979) introduced the class NC^k, as those functions computable by Boolean circuits whose depth and size are (simultaneously) bounded by $O((\log n)^k)$ and $n^{O(1)}$, respectively, where n is the number of inputs.

It turns out that the class $NC = \bigcup_{k \geqslant 0} NC^k$ is a very stable class, quite independent of the choice of any "reasonable" parallel model of computation (see, e.g., Cook, 1981). The classes NC^1 and NC^2 are of particular interest, even though the definition of these classes does depend on the choice of model; i.e., Boolean circuits.

In terms of circuits, the complexity of the basic ring operations of $+$, $-$, $\times$ for the integers has been well studied (see Savage, 1976); in particular it is known that these operations can all be realized within NC^1. If one insists on a nonredundant representation for the integers, then Winograd (1965; 1967) shows that a constant-depth implementation for addition is impossible. However, if one allows a redundant representation, then a constant depth implementation for addition is known (Avizienis, 1961).

In order to achieve our simulation of probabilistic machines, we need to utilize a representation which makes possible NC^0 and NC^1 representations for $+$ and $\times$ (respectively) and apply this to the problem of computing the determinant. Simply stated, there are known *arithmetic* circuits (see Berkowitz, 1982) for the determinant which have $O((\log n)^2)$ depth of additions but only $O(\log n)$ depth of multiplications, and the determinant is central to all the known simulations of space bounded probabilistic machines. It turns out that there are sufficiently many details to warrant a careful and somewhat abstract development.

We are thus led in Sections 2 and 3 to introduce the concept of a *well-endowed ring*. Such rings have representations in which addition and multiplication can be very efficiently implemented; that is, in NC^0 and NC^1, respectively. The development is used to extend known implementations to polynomial and matrix rings. We then apply these results in section 4 to the Boolean complexity of computing the determinant to obtain an NC^2 implementation. We use this in Section 5 when we establish an NC^2 implementation for the problem of computing the completion of a stochastic matrix. Finally, Section 6 applies the previous results to establish the desired simulation of probabilistic space-bounded machines.

The reader, not interested in the development of well-endowed rings, who

is willing to accept Corollary 4.4 (*stating that determinants over rational functions are in* NC^2), *can skip directly to Section* 5.

Motivated by a seminar talk by Simon, our depth $O(S^2)$ simulation of an $O(S)$ space-bounded probabilistic acceptor was originally obtained in the spring of 1978, and referred to in (Simon *et al.*, 1978). In 1981, Jung (1981) independently obtained an $O(S^2)$ deterministic space bounded simulation in which matrix powers were computed using modular techniques instead of redundant notation.

2. Well-Endowed Rings

Briefly, a well-endowed ring is one in which elements can be represented by bit strings in such a way that addition can be computed in constant depth and multiplication can be computed in depth $O(\log n)$. The representation can be redundant, but it must be *succinct* in the sense that "small" ring elements have short representative bit strings. Thus ring elements must have a notion of size, or *length*.

Formally, let $\mathscr{A}$ be a ring (not necessarily commutative, and not necessarily with unity). Let $\mathscr{N}$ be the monoid (semigroup) of natural numbers under addition. A *length function* for $\mathscr{A}$ is a function $\alpha: \mathscr{A} \to \mathbb{N}$ satisfying

(1) $\alpha(x+y) \leqslant \max\{\alpha(x), \alpha(y)\} + O(1)$

(2) $\alpha(xy) \leqslant \alpha(x) + \alpha(y) + O(\log \max\{\alpha(x), \alpha(y)\})$

for all x iand y in $\mathscr{A}$. For brevity, we shall say "the ring $(\mathscr{A}, \alpha)$" to mean "the ring $\mathscr{A}$ with length function α."

Let $\mathbb{Z}$ be the ring of integers. Let $p \geqslant 2$ be a natural number. It is easy to verify that $\zeta(x) = \lceil \log_p(|x|+1) \rceil$ (the number of p-ary digits needed to represent x) is a length function for $\mathbb{Z}$. In fact, ζ satisfies a stronger inequality than (2), namely $\zeta(xy) \leqslant \zeta(x) + \zeta(y)$. However, the weaker inequality is needed for polynomial rings and other examples.

From (1) and (2) it follows that

(3) $\alpha(\sum_{1 \leqslant i \leqslant n} x_i) \leqslant \max_{1 \leqslant i \leqslant n} \alpha(x_i) + O(\log n)$

(4) $\alpha(\prod_{1 \leqslant i \leqslant n} x_i) \leqslant \sum_{1 \leqslant i \leqslant n} \alpha(x_i) + O(n(\log n + \log \max_{1 \leqslant i \leqslant n} \alpha(x_i)))$.

These formulas are proved by induction, using balanced binary trees to compute the sum and product.

Let $f: \mathscr{A} \to \mathscr{B}$ be a map from the ring $(\mathscr{A}, \alpha)$ to the ring $(\mathscr{B}, \beta)$. A *bound function* for f is a function $\phi: \mathbb{N} \to \mathbb{N}$ such that

$$\beta(f(x)) \leqslant \phi(\alpha(x))$$

for all x in $\mathscr{A}$. More generally, if k is a natural number and $f: \mathscr{A}^k \to \mathscr{B}$ then ϕ should satisfy

$$\beta(f(x_1,..., x_k)) \leqslant \phi(\max\{\alpha(x_1),..., \alpha(x_k)\})$$

for all $x_1,..., x_k$ in $\mathscr{A}$. For brevity, we shall say "the map (f, ϕ)" to mean "the map f with bound function ϕ."

For a ring $(\mathscr{A}, \alpha)$ and a natural number n, we shall let $\mathscr{A}_n$ denote the set of elements x in $\mathscr{A}$ such that $\alpha(X) \leqslant n$.

A *representation* for a ring $(\mathscr{A}, \alpha)$ is a pair (l, r) comprising a function $l: \mathbb{N} \to \mathbb{N}$ and a sequence $r = \{r_1, r_2,...\}$ of functions such that $r_n: \{0, 1\}^{l(n)} \to \mathscr{A}$ and $\mathscr{A}_n$ is included in $r_n(\{0, 1\}^{l(n)})$ for all n in $\mathbb{N}$. A representation (l, r) will be called *succinct* if $l(n) = n^{O(1)}$ (i.e., if $l(n)$ is bounded above by a polynomial in n), and *uniform* if for all $x \in \mathscr{A}$ there is a log-space uniform sequence $\{u_1, u_2,...\}$ such that u_k is in $\{0, 1\}^{l(\alpha(x)+k)}$ and $r_{\alpha(x)+k}(u_k) = x$ for all $k \geqslant 1$. (Here log-space uniform means that some Turing machine can, for all k, generate u_k on a write-only output tape using workspace $O(\log |u_k|)$.) For brevity, we may say "the ring $(\mathscr{A}, \alpha, l, r)$" to mean "the ring $(\mathscr{A}, \alpha)$ with representation (l, r)."

An *implementation* for a map (f, ϕ) from a ring $(\mathscr{A}, \alpha, l, r)$ to a ring $(\mathscr{B}, \beta, m, s)$ is a sequence $F = \{F_1, F_2,...\}$ of functions such that $F_n: \{0, 1\}^{l(n)} \to \{0, 1\}^{m(\phi(n))}$ and

$$s_{\phi(n)}(F_n(u)) = f(r_n(u))$$

for all n in $\mathbb{N}$ and u in $\{0, 1\}^{l(n)}$. The notion of implementation applies to the more general case $f: \mathscr{A}^k \to \mathscr{B}$ in the obvious way.

For a natural number k, an implementation F will be said to be in NC^k if there is a log-space uniform sequence $N = \{N_1, N_2,...\}$ of Boolean networks such that N_n computes F_n, is of size $n^{O(1)}$ and is of depth $O((\log n)^k)$ for all n in $\mathbb{N}$. (See Borodin (1977) and Ruzzo (1981) for more details concerning uniform sequences of networks.) An implementation F will be said to be in NC if it is in NC^k for some k in $\mathbb{N}$. (See Pippenger (1979) and Ruzzo (1981) for further results on these classes.) By a result of Borodin (1977), if a function has an implementation in NC^k for some $k \geqslant 1$, then it is computable in space $\log^k$.

For a natural number k and a function $R: \mathbb{N} \to \mathbb{N}$ satisfying $R(n) \geqslant n$, an implementation F will be said to be in $NC^k(R)$ if there is a $(\log R)$-space uniform sequence $N = \{N_1, N_2,...\}$ of networks such that N_n computes F_n, is of size $R(n)^{O(1)}$ and is of depth $O((\log R(n))^k)$ for all $n \geqslant 1$ in $\mathbb{N}$. An implementation F will be said to be in $NC(R)$ if it is in $NC^k(R)$ for some k in $\mathbb{N}$. (The classes $NC^k(R)$ are defined as NC^k and NC, with n replaced by $R(n)$ in all resource bounds.) If a function has an implementation in $NC^k(2^S)$

for some $k \geqslant 1$ and some function S satisfying $S(n) \geqslant \log_2 n$, then it is computable in space S^k.

By *addition* in $\mathscr{A}$ we shall mean the map $f: \mathscr{A} \times \mathscr{A} \to \mathscr{A}$ for which $f(x, y) = x + y$, by *negation* in $\mathscr{A}$ we shall mean the map $f: \mathscr{A} \to \mathscr{A}$ for which $f(x) = -x$ and by *multiplication* in $\mathscr{A}$ we shall mean the map $f: \mathscr{A} \times \mathscr{A} \to \mathscr{A}$ for which $f(x, y) = xy$. An implementation for addition or negation will be called *efficient* if it is in NC^0 and an implementation for multiplication will be called *efficient* if it is in NC^1. We shall say that a ring $(\mathscr{A}, \alpha)$ is *well endowed* if there is a succinct uniform representation with respect to which it has efficient implementations for addition, negation, and multiplication.

Returning to the ring of integers, recall $\zeta(x) = \lceil \log_p(|x| + 1) \rceil$ (the number of p-ary digits needed to represent x). For $p \geqslant 2$ standard p-ary notation is (essentially) a succinct represenation for $(\mathbb{Z}, \zeta)$, but it does not yield an implementation for addition in NC^0. For $p \geqslant 3$ we shall now construct a succinct representation (l, r) for $(\mathbb{Z}, \zeta)$ with respect to which there are implementations for addition and negation in NC^0 and an implementation for multiplication in NC^1.

For $n \geqslant 1$ in $\mathbb{N}$, define $\rho_n: \{-(p-1), \ldots, -1, 0, 1, \ldots, (p-1)\}^n \to \mathbb{Z}$ by $\rho_n(u_1 \cdots u_n) = \sum_{1 \leqslant k \leqslant n} u_k p^{k-1}$ for $u_1, \ldots, u_n$ in $\{-(p-1), \ldots, -1, 0, 1, \ldots, (p-1)\}$. Clearly $\mathbb{Z}_n$ is included in $\rho_n(\{-(p-1), \ldots, -1, 0, 1, \ldots, (p-1)\}^n)$ for all $n \geqslant 1$ in $\mathbb{N}$. By encoding each symbol of $\{-(p-1), \ldots, -1, 0, 1, \ldots, (p-1)\}$ as a string of $\lceil \log_2(2p-1) \rceil$ symbols from $\{0, 1\}$, we can obtain from $\rho = (\rho_1\ \rho_2, \ldots)$ a representation (l, r) with $l(n) = \lceil \log_2(2p-1) \rceil n$. This representation is clearly succinct, and will be called the *balanced p-ary* representation for $(\mathbb{Z}, \zeta)$. This representation is due to Avizienis (1961).

To see that addition (with the bound function $\phi(n) = n + 1$) has an implementation in NC^0, suppose that we are given $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ in $\{-(p-1), \ldots, -1, 0, 1, \ldots, (p-1)\}$ and that we wish to compute $w_1, \ldots, w_{n+1}$ in $\{-(p-1), \ldots, -(p-1), \ldots, -1, 0, 1, \ldots, (p-1)\}$ such that

$$\rho_{n+1}(w_1 \cdots w_{n+1}) = \rho_n(u_1 \cdots u_n) + \rho_n(v_1 \cdots v_n).$$

Since, for $1 \leqslant k \leqslant n$, $|u_k| \leqslant p-1$, and $|v_k| \leqslant p-1$, we have $|u_k + v_k| \leqslant 2p-2$, so it is possible to write $u_k + v_k = px_k + y_k$ with $|x_k| \leqslant 1$ and $|y_k| \leqslant p-2$. (Here we use the fact that $p \geqslant 3$.) If, for $1 \leqslant k \leqslant n+1$, we set $w_k = x_{k-1} + y_k$, where $x_0 = 0$ and $y_{n+1} = 0$, then $|w_k| \leqslant p-1$ and the condition of the preceding paragraph is fulfilled. Since w_k depends only on u_k, u_{k-1}, v_k, and v_{k-1}, w can be computed from u and v in NC^0.

To see that negation (with the bound function $\phi(n) = n$) has an implementation in NC^0, observe that each symbol in $\{-(p-1), \ldots, -1, 0, 1, \ldots, (p-1)\}$ can be negated separately, since

$$\rho_n((-u_1) \cdots (-u_n)) = -\rho_n(u_1 \cdots u_n).$$

To see that multiplication (with the bound function $\phi(n) = 2n + \lceil \log_2(2n) \rceil$) has an implementation in NC^1, suppose that we are given $u_1,..., u_n$ and $v_1,..., v_n$ in $\{-(p-1),..., -1, 0, 1,..., (p-1)\}$ and that we wish to compute $w_1,..., w_{\phi(n)}$ in $\{-(p-1),..., -1, 0, 1,..., (p-1)\}$ such that

$$\rho_{\phi(n)}(w,..., w_{\phi(n)}) = \rho_n(u_1 \cdots u_n)\, \rho_n(v_1 \cdots v_n).$$

Since, for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$, $|u_i| \leqslant p-1$ and $|v_j| \leqslant p-1$, we have $|u_i v_j| \leqslant (p-1)^2 \leqslant p^2 - 1$. Thus it is possible to write $u_i v_j = p x_{i,j} + y_{i,j}$ with $|x_{i,j}| \leqslant p-1$ and $|y_{i,j}| \leqslant p-1$. If we set

$$\begin{aligned} z_{i,j} &= x_{i,j-i} && \text{if} \quad i+1 \leqslant j \leqslant n+i, \\ &= 0 && \text{otherwise,} \end{aligned}$$

and

$$\begin{aligned} z_{n+i,j} &= y_{i,j-i+1} && \text{if} \quad i+1 \leqslant j \leqslant n+i-1, \\ &= 0 && \text{otherwise,} \end{aligned}$$

for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant 2n$, then

$$\sum_{1 \leqslant i \leqslant 2n} \rho_{2n}(z_{i,1} \cdots z_{i,2n}) = \rho_n(u_1 \cdots u_n)\, \rho_n(v_1 \cdots v_n).$$

Since $z_{i,j}$ depends only on u_i and v_{j-i}, and $z_{n+i,j}$ depends only on u_i and v_{j-i+1}, z can be computed from x and y in NC^0. If we compute $w_1,..., w_{\phi(n)}$ in $\{-(p-1),..., -1, 0, 1,..., (p-1)\}$ such that

$$\rho_{\phi(n)}(w_1 \cdots w_{\phi(n)}) = \sum_{1 \leqslant i \leqslant 2n} \rho_{2n}(z_{i,1} \cdots z_{i,2n}),$$

the condition of the preceding paragraph will be fulfilled. By iterated addition (see Corollary 3.7), we can compute w from z in NC^1.

We have proved

PROPOSITION 2.1. *The ring of integers is well endowed, using balanced p-ary representations for* $p \geqslant 3$.

In computing integer functions, one might prefer standard p-ary representations to balanced p-ary representations. The former notion can be formalized as:

For $n \geqslant 1$ in $\mathbb{N}$, define $\sigma_n: \{-1, 0, 1\} \times \{0,..., p-1\}^n \to \mathbb{Z}$ by

$$\sigma_n(u_0 u_1 \cdots u_n) = u_0 \sum_{1 \leqslant k \leqslant n} u_k\, p^{k-1}$$

for u_0 in $\{-1, 0, 1\}$ and $u_1,..., u_n$ in $\{0,..., p-1\}$. Clearly $\mathbb{Z}_n$ is included in $\sigma_n(\{-1, 0, 1\} \times \{0,..., p-1\}^n)$ for all $n \geqslant 1$ in $\mathbb{N}$. By encoding each symbol of $\{-1, 0, 1\}$ as a string of two symbols from $\{0, 1\}$, and each symbol of $\{0,..., p-1\}$ as a string of $\lceil \log_2 p \rceil$ symbols from $\{0, 1\}$, we obtain from $\sigma = (\sigma_1, \sigma_2,...)$ a representation (m, s) with $m(n) = 2 + \lceil \log_2 p \rceil n$. This representation is clearly succinct, and will be called the standard p-ary representation for $(\mathbb{Z}, \zeta)$.

We shall prove

PROPOSITION 2.2. *For $p \geqslant 3$, conversion from standard to balanced p-ary representations has an implementation in NC^0, and conversion from balanced to standard p-ary representations has an implementation in NC^1.*

It is easy to see that conversion from standard p-ary to balanced p-ary has an implementation in NC^0. Simply observe that the sign can be affixed to each symbol in $\{0,..., p-1\}$ separately, since

$$\rho_n((u_0 u_1) \cdots (u_0 u_n)) = \sigma_n(u_0 u_1 \cdots u_n).$$

To see that conversion from balanced p-ary to standard p-ary has an implementation in NC^1, suppose that we are given $u_1,..., u_n$ in $\{-(p-1),..., -1, 0, 1,..., (p-1)\}$ and that we wish to compute v_0 in $\{-1, 0, 1\}$ and $v_1,..., v_n$ in $\{0,..., p-1\}$ such that

$$\sigma_n(v_0 v_1 \cdots v_n) = \rho_n(u_1 \cdots u_n).$$

The sign v_0 is the sign of the most significant nonzero digit of $u_1 \cdots u_n$ (zero if all the digits of $u_1 \cdots u_n$ are zero). Let $a_0 = 0$ and for $1 \leqslant k \leqslant n$, define a_k in $\{-1, 0, 1\}$ by $a_k = -1$ if $u_k < 0$, $a_k = 1$ if $u_k > 0$ and $a_k = a_{k-1}$ otherwise. Then $v_0 = a_n$. Thus v_0 can be computed as the final state of a finite-state machine that makes a single pass over the string u.

The digits of $v_1 \cdots v_n$ are obtained from those of $u_1 \cdots u_n$ by converting digits of the "wrong" sign to those of the "right" sign, which is done by "borrowing" from more significant digits. Let $b_0 = 0$ and for $1 \leqslant k \leqslant n$ define b_k in $\{-1, 0, 1\}$ and v_k in $\{0,..., p-1\}$ by $b_k = -1$ if $v_0 = -1$ and $u_k + b_{k-1} > 0$, $b_k = 1$ if $v_0 = 1$ and $u_k + b_{k-1} < 0$, and $b_k = 0$ otherwise. Let $v_k = v_0(u_k + b_{k-1} - p b_k)$. Then the condition of the preceding paragraph is fulfilled, and the string $v_1 \cdots v_n$ can be computed by a finite-state machine with initial state v_0 that makes a single pass over the string u.

Since functions computed by finite-state machines have implementations in NC^1 (see Ofman, 1963 or Ladner and Fischer, 1980), conversion from balanced p-ary to standard p-ary has an implementation in NC^1.

We have defined standard p-ary for $p \geqslant 2$ and balanced p-ary for $p \geqslant 3$.

For $p = 2$, balanced p-ary does not yield efficient implementations. It is easy, however, to convert from standard 2-ary to standard 4-ary and back in NC^0 (by combining successive pairs of 2-ary digits).

3. Constructing New Well-Endowed Rings from Old Ones

If the ring $\mathscr{A}$ has a length function α, then the direct product $\mathscr{A}^k$ of k copies of $\mathscr{A}$ has a natural length function α^k given by

$$\alpha^k(x_1,..., x_k) = \max\{\alpha(x_1),..., \alpha(x_k)\}.$$

We shall write $(\mathscr{A}, \alpha)^k$ for the ring $\mathscr{A}^k$ with the length function α^k.

More generally, if $\mathscr{B}$ is an algebra of dimension k over $\mathscr{A}$ with structure coefficients $t = \{t_{h,i,j}\}_{1 \leqslant h \leqslant k,\, 1 \leqslant i \leqslant k,\, 1 \leqslant j \leqslant k}$ in $\mathscr{A}$ so that

$$[x + y]_h = x_h + y_h,$$

$$[-x]_h = -x_h,$$

and

$$[xy]_h = \sum_{1 \leqslant i \leqslant k,\, 1 \leqslant j \leqslant k} t_{h,i,j} x_i y_j,$$

for all $x = (x_1,..., x_k)$ and $y = (y_1,..., y_k)$ in $\mathscr{B}$, and $1 \leqslant h \leqslant k$, and if α is a length function for $\mathscr{A}$, then β, given by

$$\beta(x_1,..., x_k) = \max\{\alpha(x_1),..., \alpha(x_k)\}$$

is a natural length function for $\mathscr{B}$. We shall write $(\mathscr{A}, \alpha)[\xi]/(A(\beta))$ for the ring of polynomials in the indeterminate ξ with coefficients in $\mathscr{A}$ modulo the polynomial $A(\xi)$ (assuming the leading coefficient of $A(\xi)$ is a unit in $\mathscr{A}$), and $(\mathscr{A}, \alpha)^{k \times k}$ for the ring of k by k matrices with entries in $\mathscr{A}$, when they are endowed with length functions in this way.

Just as a length function α for $\mathscr{A}$ extends in a natural way to a length function for any finite-dimensional algebra over $\mathscr{A}$, a representation (l, r) for $(\mathscr{A}, \alpha)$ extends in a natural way to a representation for any finite-dimensional algebra over $(\mathscr{A}, \alpha)$ (by representing each component separately) and implementations of addition, negation, and multiplication for $(\mathscr{A}, \alpha, l, r)$ extend in a natural way to implementations for any finite-dimensional algebra over $(\mathscr{A}, \alpha, l, r)$ (by computing each component separately, using the formulae in the preceding paragraph). Furthermore, succinct uniform representations extend to succinct uniform representations and efficient implementations extend to efficient implementations. We summarize these results as

PROPOSITION 3.1. *If* $(\mathscr{A}, \alpha)$ is a well-endowed ring and $(\mathscr{B}, \beta)$ *is a finite-dimensional algebra over* $(\mathscr{A}, \alpha)$, *then* $(\mathscr{B}, \beta)$ *is a naturally well endowed.*

COROLLARY 3.2. *If* $(\mathscr{A}, \alpha)$ *is well endowed, then* $(\mathscr{A}, \alpha)^k$ *is naturally well endowed.*

COROLLARY 3.3. *If* $(\mathscr{A}, \alpha)$ *is well endowed, then* $(\mathscr{A}, \alpha)[\xi]/(A(\xi))$ *is naturally well endowed.*

COROLLARY 3.4. *If* $(\mathscr{A}, \alpha)$ *is well endowed, then* $(\mathscr{A}, \alpha)^{k \times k}$ *is naturally well endowed.*

We now turn our attention to infinite-dimensional algebras.

Let $\mathscr{A}$ be a ring. Let $\mathscr{A}^\infty$ denote the weak direct product of a countably infinite sequence of copies of $\mathscr{A}$; that is, the ring of countably infinite sequences of elements of $\mathscr{A}$ in which all but finitely many components vanish, with componentwise addition and multiplication. By the *order* $\operatorname{ord}(x)$ of an element $x = (x_1, x_2, \ldots)$ in $\mathscr{A}^\infty$ we shall mean the largest n such that x_n does not vanish. If $\mathscr{A}$ has a length function α, then $\mathscr{A}^\infty$ has a natural length function α^∞, given by

$$\alpha^\infty(x) = \max\{\operatorname{ord}(x), \max_{1 \leqslant n < \infty} \alpha(x_n)\}.$$

We shall write $(\mathscr{A}, \alpha)^\infty$ for $(\mathscr{A}^\infty, \alpha^\infty)$. If $(\mathscr{A}, \alpha)$ has a representation (l, r), then $(\mathscr{A}, \alpha)^\infty$ has a natural representation (l^∞, r^∞), given by $l^\infty(n) = nl(n)$ and

$$r_n^\infty(u_1 \cdots u_n) = (r_n(u_1), \ldots, r_n(u_n), 0, \ldots)$$

for all n in $\mathbb{N}$ and $u_1, \ldots, u_n$ in $\{0, 1\}^{l(n)}$. We shall write $(l, r)^\infty$ for (l^∞, r^∞). If (l, r) is succinct and uniform, then so is $(l, r)^\infty$. Furthermore, if $(\mathscr{A}, \alpha)$ has, with respect to (l, r), implementations for addition and negation in NC^0 and an implementation for multiplication in NC^1, then so does $(\mathscr{A}, \alpha)^\infty$ with respect to $(l, r)^\infty$.

We summarize these results as

PROPOSITION 3.5. *If the ring* $\mathscr{A}$ *is well endowed, then the ring* $\mathscr{A}^\infty$ *is naturally well endowed.*

We shall say that a map $f: \mathscr{A} \times \mathscr{A} \to \mathscr{A}$ is *associative* if $f(x, f(y, z)) = f(f(x, y), z)$ for all x, y, z in $\mathscr{A}$, and e is a *neutral element* for f if $f(x, e) = x$

for all x in $\mathscr{A}$. Let $\hat{\mathscr{A}}$ denote the strong direct product product of a countably infinite sequence of copies of $\mathscr{A}$; that is, the ring of countably infinite sequences of elements of $\mathscr{A}$ with componentwise addition and multiplication. Then $f^*: \mathscr{A}^\infty \to \hat{\mathscr{A}}$ (*iterated* f) is defined by $f^*(x_1, x_2, x_3, \ldots) = (x_1, f(x_1, x_2), f(f(x_1, x_2), x_3), \ldots)$. The ring $\hat{\mathscr{A}}$ is not, in general, well endowed, because it does not inherit a length function from $\mathscr{A}$ in a natural way. Nevertheless, we can speak of an implementation of f^* as follows.

Let $\mathscr{A}$ have length function α and representation (l, r). Suppose f is associative and has bound function ϕ, where $\phi(n) \geqslant n$ for all n. Let $\phi^*(n) = \phi^{\lceil \log_2 n \rceil}(n)$; that is, ϕ composed with itself $\lceil \log_2 n \rceil$ times and evaluated at n. Now suppose $x \in \mathscr{A}^\infty$, $1 \leqslant k \leqslant n$, and $\alpha^\infty(x) \leqslant n$. Then the element $[f^*(x)]_k$ can be constructed by a balanced binary tree of applications of f, so that $\alpha([f^*(x)]_k) \leqslant \phi^{\lceil \log_2 k \rceil}(n) \leqslant \phi^*(n)$.

An *implementation* for f^* is a sequence $F^* = \{F_1^*, F_2^*, \ldots\}$ of functions such that $F_n^*: \{0,1\}^{nl(n)} \to \{0,1\}^{nl(\phi^*(n))}$ and for all u in $\{0,1\}^{nl(n)}$, $F_n^*(u) = w_1 w_2 \cdots w_n$, where $w_k \in \{0,1\}^{l(\phi^*(n))}$ and $r_{\phi^*(n)}(w_k) = [f^*(r_n^\infty(u))]_k$, for $1 \leqslant k \leqslant n$.

PROPOSITION 3.6. *If $\mathscr{A}$ is a well endowed ring, and $f: \mathscr{A} \times \mathscr{A} \to \mathscr{A}$ is an associative map with a neutral element and with an implementation in NC^k for some k in $\mathbb{N}$, then $f^*: \mathscr{A}^\infty \to \hat{\mathscr{A}}$ has an implementation in NC^{k+1}.*

The proof is a tree construction (the "parallel prefix algorithm") in the same spirit as Ofman (1963) and Ladner and Fischer (1980). The only additional complication here is the necessity of "padding" (increasing the lengths of) the representations of certain elements computed in the implementing networks so that they will be right for the outputs, and right for inputs to the subnetworks implementing f. The padding of the representation of x is accomplished by computing $f(x, e)$, where e is the neutral element. The representations for e are log-space uniform since the representation (l, r) is uniform.

Since addition is associative, has neutral element 0, and has the bound function $\phi(n) = n + O(1)$, we have

COROLLARY 3.7. *In a well-endowed ring, iterated addition has an implementation in NC^1 and a bound function $\phi^*(n) = n + O(\log n)$.*

Since multiplication is associative, has neutral element 1, and has the bound function $\phi(n) = 2n + O(\log n)$, we have

COROLLARY 3.8. *In a well-endowed ring, iterated multiplication has an implementation in NC^2 and a bound function $\phi^*(n) = 2n^2 + O(n \log n)$.*

Let $\mathscr{A}$ be a ring. Let $\mathscr{A}[\xi]$ denote the ring of polynomials in the indeterminate ξ over $\mathscr{A}$, that is, the ring of formal power series

$$A(\xi) = \sum_{0 \leqslant k < \infty} A_k \xi^k$$

in which all but finitely many of the coefficients A_k vanish. The *degree* $\deg(A(\xi))$ of such a polynomial is the largest k such that A_k does not vanish. If $\mathscr{A}$ has a length function α, then $\mathscr{A}[\xi]$ has a natural length function α' given by

$$\alpha'(A(\xi)) = \max\{\deg(A(\xi)), \max_{0 \leqslant k < \infty} \alpha(A_k)\}.$$

We shall write $(\mathscr{A}, \alpha)[\xi]$ for $(\mathscr{A}[\xi], \alpha')$. If $(\mathscr{A}, \alpha)$ has representation (l, r), then $(\mathscr{A}, \alpha)[\xi]$ has a natural representation (l', r'), given by $l'(n) = (n+1)\, l(n)$ and

$$\begin{aligned} [r'_n(u_0 \cdots u_n)]_k &= r_n(u_k) \qquad \text{if} \quad 0 \leqslant k \leqslant n, \\ &= 0 \qquad\qquad\quad \text{otherwise,} \end{aligned}$$

for all $n \geqslant 1$ in $\mathbb{N}$, $u_0, \ldots, u_n$ in $\{0,1\}^{l(n)}$ and k in $\mathbb{N}$. We shall write $(l, r)[\xi]$ for (l', r'). If (l, r) is succinct, then so is $(l, r)[\xi]$. Furthermore, if $(\mathscr{A}, \alpha)$ has, with respect to (l, r), implementations for addition and negation in NC^0 and an implementation for multiplication in NC^1, then so does $(\mathscr{A}, \alpha)[\xi]$ with respect to $(l, r)[\xi]$. (For multiplication, observe that the sum

$$[A(\xi)\, B(\xi)]_k = \sum_{0 \leqslant j \leqslant k} A_j B_{k-j}$$

can be computed in NC^1 by iterated addition.)

We summarize these results as

PROPOSITION 3.9. *If the ring $\mathscr{A}$ is well endowed, then the ring $\mathscr{A}[\xi]$ is naturally well endowed.*

We close this section with a result which will be useful in computing stochastic closures (Section 5).

Let $\mathscr{A}$ be a commutative ring with unity. Let the map $V: \mathscr{A}[\xi] \to \mathscr{A}^\infty$ be defined by

$$[V(A(\xi))]_k = (d^k A(\xi)/d\xi^k)\,|_{\xi = 1}$$

for all $A(\xi)$ in $\mathscr{A}[\xi]$ and k in $\mathbb{N}$.

PROPOSITION 3.10. *If $\mathscr{A}$ is a well-endowed commutative ring with unity,*

then the map $V: \mathscr{A}[\xi] \to \mathscr{A}^{\infty}$ *has an implementation in* NC^2 *and a bound function* $\phi(n) = O(n \log n)$.

The proof depends upon the obvious formula

$$[V(A(\xi))]_k = \sum_{k \leqslant l \leqslant m} A_l l!/(l-k)!,$$

which holds for all k and m in $\mathbb{N}$ and all $A(\xi)$ of degree at most m in $\mathscr{A}[\xi]$.

The array

$$\begin{aligned} F_{k,l} &= 1 \qquad \text{if} \quad k \leqslant l, \\ &= 0 \qquad \text{otherwise,} \end{aligned}$$

for $k \geqslant 1$ and $l \geqslant 1$ can be computed (as elements of $\mathscr{A}$) in NC^0. The array

$$\begin{aligned} G_{k,l} &= l - k \qquad \text{if} \quad k \leqslant l, \\ &= 0 \qquad \text{otherwise,} \end{aligned}$$

for $k \geqslant 1$ and $l \geqslant 0$ can be computed from F in NC^1 by iterated addition on the rows. The array

$$\begin{aligned} H_{k,l} &= l!/(l-k)! \qquad \text{if} \quad k \leqslant l, \\ &= 0 \qquad \text{otherwise,} \end{aligned}$$

for $k \geqslant 0$ and $l \geqslant 0$ can be computed from G in NC^2 by iterated multiplication on the columns. The sum in the formula can be computed from $A(\xi)$ and H in NC^1. The verification of the bound function is straightforward and completes the proof of Proposition 3.10.

4. Determinants

Let $\mathscr{A}$ be a ring. Let $\mathscr{A}^{\infty \times \infty}$ denote the ring of infinite matrices $\{A_{i,j}\}_{1 \leqslant i < \infty,\, 1 \leqslant j < \infty}$ for which all but finitely many entries $A_{i,j}$ vanish. The *order*, $\operatorname{ord}(A)$, of such a matrix is the largest k such that at least one of the entries $A_{1,k}, \ldots, A_{k,k}, \ldots, A_{k,1}$ does not vanish. If $\mathscr{A}$ has a length function α, then $\mathscr{A}^{\infty \times \infty}$ has a natural length function $\alpha^{\infty \times \infty}$, given by

$$\alpha^{\infty \times \infty}(A) = \max\{\operatorname{ord}(A), \max_{1 \leqslant i < \infty,\, 1 \leqslant j < \infty} \alpha(A_{i,j})\}.$$

We shall write $(\mathscr{A}, \alpha)^{\infty \times \infty}$ for $(\mathscr{A}^{\infty \times \infty}, \alpha^{\infty \times \infty})$. If $(\mathscr{A}, \alpha)$ has a represen-

tation (l, r), then $(\mathscr{A}, \alpha)^{\infty\times\infty}$ has a natural representation $(l^{\infty\times\infty}, r^{\infty\times\infty})$, given by $l^{\infty\times\infty}(n) = n^2 l(n)$ and

$$\begin{aligned}[r_n^{\infty\times\infty}(u_{1,1}\cdots u_{1,n}\cdots u_{n,1}\cdots u_{n,n})]_{i,j} \\ = r_n(u_{i,j}) \qquad &\text{if} \quad 1 \leqslant i \leqslant n \quad \text{and} \quad 1 \leqslant j \leqslant n, \\ = 0 \qquad &\text{otherwise,}\end{aligned}$$

for all $n \geqslant 1$ in $\mathbb{N}$, all $u_{1,1},\ldots,u_{n,n}$ in $\{0,1\}^{l(n)}$ and all $i \geqslant 1$ and $j \geqslant 1$ in $\mathbb{N}$. We shall write $(l, r)^{\infty\times\infty}$ for $(l^{\infty\times\infty}, r^{\infty\times\infty})$. If (l, r) is succinct and uniform, then so is $(l, r)^{\infty\times\infty}$. Furthermore, if $(\mathscr{A}, \alpha)$ has, with respect to (l, r), implementations for addition and negation in NC^0 and an implementation for multiplication in NC^1, then so does $(\mathscr{A}, \alpha)^{\infty\times\infty}$ with respect to $(l, r)^{\infty\times\infty}$. (For multiplication, observe that the sum

$$[AB]_{i,k} = \sum_{1\leqslant j<\infty} A_{i,j}B_{j,k}$$

can be computed in NC^1 by iterated addition—see Corollary 3.7.)

We summarize these results as

PROPOSITION 4.1. *If the ring $\mathscr{A}$ is well endowed, then the ring $\mathscr{A}^{\infty\times\infty}$ is naturally well endowed.*

Let $\mathscr{A}$ be a commutative ring with unity. Let the map $D: \mathscr{A}^{\infty\times\infty} \to \mathscr{A}^{\infty}$ be defined by

$$[D(A)]_k = \Delta(\{A_{i,j}\}_{1\leqslant i\leqslant k,\, 1\leqslant j\leqslant k})$$

for all A in $\mathscr{A}^{\infty\times\infty}$ and $k \geqslant 1$ in $\mathbb{N}$, where Δ denotes the determinant.

PROPOSITION 4.2. *If $\mathscr{A}$ is a well-endowed commutative ring with unity, then the map $D: \mathscr{A}^{\infty\times\infty} \to \mathscr{A}^{\infty}$ has an implementation in NC^2 with a bound function $\phi(n) = n^{O(1)}$.*

The proof depends on the existence of an appropriate log-space uniform sequence $\{\delta_1, \delta_2,\ldots\}$ of arithmetic circuits such that δ_n computes the determinant of an $n \times n$ integer matrix, $n = 1, 2,\ldots$. Such circuits are described in (Borodin *et al.*, 1982), but a simpler and more direct construction is given by Berkowitz in (1982). (Csanky's method (1976) is not general enough for our purposes since it requires division by integer constants.)

To describe the method in (Berkowitz, 1982), let B be an arbitrary $n \times n$ matrix over $\mathscr{A}$. For $1 \leqslant t \leqslant n-1$ let M_t be the $t \times t$ lower right submatrix of

B, let R_t be the $1 \times t$ row submatrix immediately above M_t, and let S_t be the $t \times 1$ column submatrix immediately to the left of M_t. That is,

$$\begin{aligned} [M_t]_{ij} &= B_{n-t+i,n-t+j}, & 1 \leqslant i,j \leqslant t \\ [R_t]_{1,j} &= B_{n-t,n-t+j}, & 1 \leqslant j \leqslant t \\ [S_t]_{i,1} &= B_{n-t+i,n-t}, & 1 \leqslant i \leqslant t. \end{aligned}$$

For $0 \leqslant t \leqslant n-1$ let C_t be the $(t+2) \times (t+1)$ matrix over $\mathscr{A}$ defined as follows:

$$\begin{aligned} [C_t]_{i,j} &= -R_t M_t^{i-j-2} S_t, & 1 \leqslant j \leqslant i-2, \\ &= B_{n-t,n-t}, & j = i-1, \\ &= -1, & j = i, \\ &= 0, & i+1 \leqslant j \leqslant t+1. \end{aligned}$$

In particular, $C_0 = [\,B_{nn}^{-1}\,]$. Then for $1 \leqslant t \leqslant n$, the matrix product $\prod_{k=1}^{t} C_{t-k}$ is a $(t+1) \times 1$ column matrix comprising the coefficients of the characteristic polynomial $\Delta(M_t - \lambda I)$ of M_t, where we define $M_n = B$. Therefore,

$$\Delta(B) = [C_{n-1} C_{n-2} \cdots C_0]_{n,1}.$$

The above formula for the determinant can be implemented using iterated matrix multiplication (Corollary 3.8) and padding (see the proof of Proposition 3.6). This completes the proof of Proposition 4.2.

In Section 6 we shall need the fact that determinants of matrices whose entries are rational functions can be computed in NC^2. This does not follow immediately from Proposition 4.2, because the field of rational functions does not seem to be a well-endowed ring. However, that proposition can be generalized to apply to the field of fractions of any well-endowed integral domain.

Let $(\mathscr{A}, \alpha, l, r)$ be a well-endowed integral domain. Let $\mathscr{A}'$ be the field of fractions of $\mathscr{A}$. Then $\mathscr{A}'$ inherits a natural representation (l', r') from (l, r), where $l'(n) = 2l(n)$ and

$$\begin{aligned} r_n(uv) &= r_n(u)/r_n(v) & \text{if} \quad r_n(v) \neq 0, \\ &= \text{undefined} & \text{if} \quad r_n(v) = 0, \end{aligned}$$

for all $u, v \in \{0,1\}^{l(n)}$. Notice that (l', r') is not strictly a representation in the sense of Section 2, because $\mathscr{A}'$ does not inherit a length function α' from α satisfying conditions (1) and (2) in any obvious way. Nevertheless (l', r') is appropriate for implementing the determinant function D.

COROLLARY 4.3. *Let $\mathscr{A}$ be a well-endowed integral domain and let $\mathscr{A}'$ be its field of fractions. Then the map $D: (\mathscr{A}')^{\infty\times\infty} \to (\mathscr{A}')^{\infty}$ has an implementation in NC^2.*

To prove the corollary, note that the input to the nth network implementing D represents an $n \times n$ matrix A with each element A_{ij} presented as a fraction x_{ij}/y_{ij}, with x_{ij}, y_{ij} in $\mathscr{A}$. (We assume $y_{ij} \neq 0$.) The network computes the matrix B, where $B_{ij} = x_{ij} \prod_{k \neq j} y_{ik}$ by iterated multiplication, in NC^2. Then $\Delta(B)$ is computed in NC^2 by Proposition 4.2, and the output is $\Delta(A) = \Delta(B)/Y$, where $Y = \prod_{i,j} y_{ij}$ is computed by iterated multiplication.

COROLLARY 4.4. *The determinant of an $n \times n$ matrix of rational functions can be computed in NC^2, where the entries are presented as pairs of polynomials of (formal) degree n in some constant number of variables with n-digit integer coefficients.*

5. COMPLETION OF A STOCHASTIC MATRIX

Let A be $n \times n$ Boolean matrix. By the *transitive closure* of A we shall mean the $n \times n$ matrix A^* given by the formula

$$A^* = \sum_{0 \leqslant t < \infty} A^t,$$

where A^t denotes the tth Boolean power of A (A^0 denotes the $n \times n$ Boolean identity matrix) and addition and multiplication of Boolean values are interpreted as disjuction and conjuction, respectively. A simple argument shows that A^* is given by the finite sum

$$A^* = \sum_{0 \leqslant t < n} A^t,$$

and thus by the formula

$$A^* = (I + A)^{n-1}.$$

Since the Boolean product of two $n \times n$ matrices can be computed in Boolean parallel time $O(\log n)$, it follows that A^* can be computed from A in Boolean parallel time $O((\log n)^2)$.

This well-known result can be used to prove the result, due to Savitch (1970), that a nondeterministic machine accepting a language in space S can be simulated by a deterministic machine recognizing the same language in space S^2. We shall establish an analogue of Savitch's theorem for

probabilistic rather than nondeterministic machines. This leads at once to the following problem.

Let A be an $n \times n$ stochastic matrix (an $n \times n$ matrix of nonnegative real numbers in which the entries in each row sum to 1). By the *completion* of A we shall mean the $n \times n$ matrix A^* given by the formula

$$A^* = \sum_{0 \leqslant t < \infty} A^t.$$

Some remarks about the interpretation of this formula are in order. If A is stochastic, then so is A^t for any t in $\mathbb{N}$. Thus, in each term of the sum, the sum of all matrix entries is n, and in the ring of $n \times n$ matrices of real numbers with its usual topology, the sum defining A^* always diverges. To make sense of this formula, we shall interpret it in the following alternative way. For each $1 \leqslant r \leqslant n$ and $1 \leqslant s \leqslant n$, the partial sum

$$\sum_{0 \leqslant t \leqslant u} [A^t]_{r,s}$$

is nonnegative and nondecreasing in u. If it is unbounded, it diverges to ∞, and we shall take $[A^*]_{r,s}$ to be ∞. If it is bounded, it converges to a nonnegative real number x, and we shall take $[A^*]_{r,s}$ to be x. Thus, the entries of A^* are nonnegative extended real numbers.

It happens that if the entries of A are rational, then so are the finite entries of A^* (this will in fact be proved below). Simon (1981a) showed that if the entries of A are drawn from $\{0, \frac{1}{2}, 1\}$, then A^* can be computed in Boolean space $O((\log n)^6)$. He used this result to show that a probabilistic machine accepting a language in space S can be simulated by a deterministic machine recognizing the same language in space S^6. Gill, Hunt, and Simon (1980) used this result in turn to show that a probabilistic machine computing a function in space S can be simulated by a deterministic machine computing the same function in space S^{36}, and Hunt (1978) subsequently improved this result from S^{36} to S^6.

In this section, we shall show that if the entries of an $n \times n$ stochastic matrix A are rational, then A^* can be computed in NC^2 (and thus in Boolean space $O((\log n)^2)$). We shall use this result in the next section to improve the simulations of probabilistic machines by deterministic machines just cited from S^6 to S^2. This improvement constitutes a generalization of Savitch's theorem, since the languages accepted by probabilistic machines in space S include those accepted by nondeterministic machines in space S.

We shall prove

PROPOSITION 5.1. *If A is an $n \times n$ rational stochastic matrix, whose entries are given as pairs of integers (numerator, denominator) having*

standard radix representations, then its completion closure A^* can be computed in the same form in NC^2.

Let A be an $n \times n$ rational stochastic matrix. We wish to compute the (r, s)th entry of its completion:

$$[A^*]_{r,s} = \lim_{u \to \infty} \sum_{0 \leqslant t \leqslant u} [A^t]_{r,s}.$$

Now consider the ring $\mathbb{Q}[[\xi]]$ of formal power series in indeterminate ξ with rational coefficients. Since A is stochastic, the series $\sum_{0 \leqslant t < \infty} [A^t]_{r,s} \xi^t$ converges for each real ξ in the interval $0 \leqslant \xi < 1$. Further, the sum $\sum_{0 \leqslant t \leqslant u} [A^t]_{r,s} \xi^t$ is monotone increasing in both u and ξ, so that $\lim_{\xi \uparrow 1} \sum_{0 \leqslant t < \infty} [A^t]_{r,s} \xi^t = \lim_{u \to \infty} \sum_{0 \leqslant t \leqslant u} [A^t]_{r,s}$, where $\lim_{\xi \uparrow 1}$ indicates that ξ is restricted to values less than 1. Therefore,

$$[A^*]_{r,s} = \lim_{\xi \uparrow 1} \sum_{0 \leqslant t < \infty} [A^t]_{r,s} \xi^t.$$

Let $\mathbb{Q}^{n \times n}[[\xi]]$ denote the ring of formal power series in indeterminate ξ with $n \times n$ rational matrices as coefficients. The formal power series $I - A\xi$ has a two-sided inverse in this ring, namely

$$(I - A\xi)^{-1} = \sum_{0 \leqslant t < \infty} A^t \xi^t,$$

as can be verified directly by multiplication. Now observe that $\mathbb{Q}^{n \times n}[[\xi]]$ is isomorphic to $\mathbb{Q}[[\xi]]^{n \times n}$, the ring of matrices with formal power series as elements, by the obvious bijection. Hence we may write

$$[(I - A\xi)^{-1}]_{r,s} = \sum_{0 \leqslant t < \infty} [A^t]_{r,s} \xi^t.$$

Therefore

$$[A^*]_{r,s} = \lim_{\xi \uparrow 1} [(I - A\xi)^{-1}]_{r,s}.$$

For the purpose of evaluating the right-hand side of this equation, $(I - A\xi)^{-1}$ may be regarded as an element of $\mathbb{Q}(\xi)^{n \times n}$, since the subring of $\mathbb{Q}(\xi)$ consisting of those rational functions with no pole at $\xi = 0$ is naturally a subring of $\mathbb{Q}[[\xi]]$. Hence by Corollary 4.4 and Cramer's rule, the function of A with value $[(I - A\xi)^{-1}]_{r,s}$, which can be expressed and computed as a quotient of integer polynomials $P(\xi)/Q(\xi)$, is in NC^2.

It remains to consider the limiting process in the formula

$$[A^*]_{r,s} = \lim_{\xi \uparrow 1} P(\xi)/Q(\xi).$$

The question, of course, is whether or not the rational function represented by $P(\xi)/Q(\xi)$ has a pole at $\xi = 1$: if so, then $[A^*]_{r,s} = \infty$. If not, then it assumes a finite value x and $[A^*]_{r,s} = x$. It does not suffice to inquire as to whether or not not $Q(1) = 0$, for $P(\xi)$ and $Q(\xi)$ might possess some number of common zeros at $\xi = 1$ without the rational function represented by $P(\xi)/Q(\xi)$ having a pole at $\xi = 1$. One solution to this problem is to eliminate common factors of $(1 - \xi)$ from $P(\xi)$ and $Q(\xi)$ by division, using Newton's method to perform division in NC^2 (see, e.g., Borodin and Munro (1975, Theorem 4.4.1)). It is possible to avoid division, however, in the following way.

Let p denote the smallest natural number m such that $[V(P(\xi))]_m = (d^m P(\xi)/d\xi^m)|_{\xi=1}$ does not vanish, and let q denote the smallest natural number m such that $[V(Q(\xi))]_m = (d^m Q(\xi)/d\xi^m)|_{\xi=1}$ does not vanish. By l'Hôpital's rule, if $p < q$, then

$$\lim_{\xi \uparrow 1} P(\xi)/Q(\xi) = \infty,$$

and if $p \geqslant q$, then

$$\lim_{\xi \uparrow 1} P(\xi)/Q(\xi) = [V(P(\xi))]_q / [V(Q(\xi))]_q.$$

These computations can be performed in NC^2 by Proposition 3.10. This completes the proof of Proposition 5.1.

6. Space-Bounded Probabilistic Machines

In this section we shall prove that a probabilistic machine running in space S can be simulated by a deterministic machine running in space S^2. First, however, we shall need a lemma concerning the computation of probabilities.

An $n \times n$ stochastic matrix A can be regarded as a Markov process with states $\{1,..., n\}$, with $A_{r,s}$ being the probability of transition from state r to state s in one step. By elementary arguments, $[A^t]_{r,s}$ is the probability of transition from state r to state s in t steps, and $[A^*]_{r,s}$ is the expected number of steps that A spends in state s when started in state r. If state s cannot return to itself in one or more steps, then $[A^*]_{r,s}$ is the probability that A ever enters state s when started in state r.

Let $\{i\}$, J and K be mutually disjoint subsets of $\{1,..., n\}$. We shall denote by $P(A: i \to J; K)$ the probability that the process A, started in state i, enters a state in J without previously entering a state in K. We shall abbreviate $P(A: i \to J; \phi)$ by $P(A: i \to J)$.

We shall assume that a subset H of $\{1,...,n\}$ is represented by its characteristic vector

$$\chi_H(h) = 1 \quad \text{if} \quad h \in H,$$
$$= 0 \quad \text{otherwise,}$$

for $1 \leqslant h \leqslant n$.

LEMMA 6.1. *Let A be an $n \times n$ stochastic matrix whose entries are drawn from $\{0, \frac{1}{2}, 1\}$, and let $\{i\}$, J, and K be mutually disjoint subsets of $\{1,...,n\}$. The probability $P(A: i \to J; K)$ can be computed from A, i, J, and K in NC^2.*

Define the $(n+2) \times (n+2)$ stochastic matrix B with entries drawn from $\{0, \frac{1}{2}, 1\}$ as follows. For $1 \leqslant r \leqslant n$ and $1 \leqslant s \leqslant n$, let $B_{r,s} = A_{r,s}$ if s is not in J or K and $B_{r,s} = 0$ if s is in J or K. For $1 \leqslant r \leqslant n$, let

$$B_{r,n+1} = \sum_{s \in J} A_{r,s}$$

and

$$B_{r,n+2} = \sum_{s \in K} A_{r,s}.$$

For $n+1 \leqslant r \leqslant n+2$, let $B_{r,s} = 0$ for $1 \leqslant s \leqslant n+1$ and let $B_{r,n+2} = 1$. The process B mimics A until A enters a state of J or K. When A enters a state of J, B enters the state $n+1$, remains there for one step, then enters the absorbing state $n+2$. When A enters a state of K, B enters the absorbing state $n+2$. The probability $P(A: i \to J; K)$ is the probability that B ever enters state $n+1$ when started in state i. Since state $n+1$ cannot return to itself in one or more steps, this equals the expected number of steps that B spends in state $n+1$ when started in state i, which is $[B^*]_{i,n+1}$. The matrix B can be computed from A in NC^1, and by Proposition 5.1, $[B^*]_{i,n+1}$ can be computed from B in NC^2. This completes the proof of Lemma 6.1.

We shall define a probabilistic machine to be a machine funished with two alternative deterministic transition functions. At each step, the machine moves in accordance with one or the other of its transition functions, depending upon the outcome of an independent unbiased coin flip.

We shall assume that a machine has a finite control, an input tape bearing an input string X of length k which is accessed through a two-way read-only input head and a work tape of length $S(k)$ (where S is a constructable function satisfying $S(k) \geqslant \log_2 k$) which is accessed through a two-way read/write work head.

Let us consider the acceptance of languages. For this we shall assume that the finite control has certain designated accepting states. We shall say that a

probabilistic machine accepts that language comprising the input strings for which the probability that the machine ever enters an accepting state strictly exceeds $\frac{1}{2}$. (This definition assigns a language to each probabilistic machine.)

This is not the only possible definition of a probabilistic machine accepting a language, but it seems to be the broadest, and for our purposes, the broader the definition of acceptance the stronger our results. In any case, it possesses the following two attractive properties: (1) if a language can be accepted by a nondeterministic machine in space S, then it can be accepted by a probabilistic machine in space S (this is trivial); and (2) if a language can be accepted by a probabilistic machine in space S, then so can its complement (see Simon, 1981b).

Let M be a probabilistic machine that accepts a language L in space S. By a *configuration* of M for input X of length k we shall mean a natural number in $\{1,..., k\}$, representing the position of the input head, a symbol of the input alphabet, representing the symbol scanned by the input head, a natural number in $\{1,..., S(k)\}$, representing the position of the work head, a string of length $S(k)$ of symbols of the work alphabet, representing the condition of the work tape, and, finally, a symbol representing the state of the finite control. For a given input string X of length k, there are at most $V(k) = 2^{O(S(k))}$ configurations of M for X, and these configurations can be encoded as elements of $\{1,..., V(k)\}$ in a natural way. Furthermore, the $V(k) \times V(k)$ one-step transition matrix A is a stochastic matrix with entries drawn from $\{0, \frac{1}{2}, 1\}$ which can be computed from X in $NC^0(V) = NC^0(2^S)$. The initial configuration i and the set of accepting configurations J (the set of configurations containing an accepting state) can also be computed from X in $NC^0(V) = NC^0(2^S)$.

We can now construct a deterministic machine M' that recognizes the language L in space S^2. The machine M' simply computes $P(A: i \to J)$, accepts if it strictly exceeds $\frac{1}{2}$, and rejects otherwise. By Lemma 6.1 and the observations of the preceding paragraph, the computation of M' can be performed in $NC^2(V) = NC^2(2^S)$ and thus in space S^2.

We have proved

PROPOSITION 6.2. *If a language can be accepted by a probabilistic machine in space S, it can be recognized in $NC^2(2^S)$.*

COROLLARY 6.3. *If a language can be accepted by a probabilistic machine in logarithmic space, it can be recognized in NC^2.*

Now let us consider the computation of functions. For this we shall assume that the machine has an output tape accessed through a one-way write-only output head. We shall say that a probabilistic machine M computes that partial function whose domain comprises those input strings X

for which there exists an output string Y such that with probability strictly exceeding $\frac{1}{2}$, M halts after writing Y as output, and whose value for the string X in its domain is the string Y. (For each input X there can be at most one output Y satisfying this definition, so the definition assigns a partial function to each probabilistic machine.)

Let M be a probabilistic machine computing a partial function f. By a theorem of Gill (1977), if for some input string X of length k the machine M halts after writing a string Y of length l as output with probability strictly exceeding $\frac{1}{2}$, then $l \leqslant V(k)$. By an *output situation* for an input X of length k, we shall mean (1) the element 0, representing the fact that M has not yet written an output symbol, (2) a pair (m, σ), where m is a natural number in $\{1,..., V(k)\}$ and σ is an element of the output alphabet, representing the fact that M has written m output symbols, the last of which was σ, or (3) the element ω, representing the fact that M has written more than $V(k)$ output symbols. By a *superconfiguration* for an input string X we shall mean a configuration for X together with an output situation for X. For a given input X of length k, there are at most $W(k) = O(V(k)^2) = 2^{O(S(k))}$ superconfigurations for X, and these superconfigurations can be encoded as elements of $\{1,..., W(k)\}$ in a natural way. Furthermore, the $W(k) \times W(k)$ one-step transition matrix A is a stochastic matrix with entries drawn from $\{0, \frac{1}{2}, 1\}$ which can be computed from X in $NC^0(W) = NC^0(2^S)$.

We shall now construct a deterministic machine that computes the partial function f in space S^2. We shall do this in two steps. First we shall construct a deterministic machine M'' that computes a total function g that is an extension of f in space S^2. (This construction could be reduced to Proposition 6.2 rather than the more basic Lemma 6.1.) Then we shall construct a deterministic machine M''' that recognizes the domain of f in space S^2. Combining these two constructions completes the proof.

We shall say that an output situation (m, σ) is *appropriate* for an input X if M enters a superconfiguration having output situation (m, σ) with probability strictly exceeding $\frac{1}{2}$. Let i be the initial superconfiguration and let J be the set of superconfigurations having output situation (m, σ). Then (m, σ) is appropriate if and only if $P(A: i \to J) > \frac{1}{2}$. Thus, by Lemma 6.1, whether or not a given output situation is appropriate for X can be computed from X in $NC^2(W) = NC^2(2^S)$ and thus in space S^2.

The machine M'' with input of length k behaves as follows. For each natural number m from 1 to $V(k)$ in turn, it determines whether or not there exists an output symbol σ such that (m, σ) is appropriate for X. If not, M'' halts. If so, then this σ is unique (since at most one symbol can be written as the mth output with probability strictly exceeding $\frac{1}{2}$); M'' writes σ as the mth output and proceeds to the next value of m.

The machine M'' computes a total function g that is an extension of f, for if M halts after writing a string Y of length l as output with probability

strictly exceeding $\frac{1}{2}$, then for each $1 \leqslant m \leqslant l$, if σ is the mth symbol of Y, then certainly M enters a superconfigurtion having output situation (m, σ) with probability strictly exceeding $\frac{1}{2}$. Furthermore, by the observations of the preceding paragraphs, the computation of M'' can be performed in $NC^2(W) = NC^2(2^S)$ and thus in space S^2, which completes the first part of the proof.

The machine M''' with input X of length k behaves as follows. First, it computes the length l and the last symbol ρ of $g(X)$. This can be done in $NC^2(W) = NC^2(2^S)$ and thus in space S^2. Let i be the initial superconfiguration, let J be the set of halting superconfigurations having output situation (l, ρ) and let K be the set of superconfigurations having an output situation (m, σ) that is not appropriate for X. These can also be computed from X in $NC^2(W) = NC^2(2^S)$ and thus in space S^2. If $P(A: i \to J; K) > \frac{1}{2}$ then M''' accepts, otherwise it rejects.

The probability $P(A: i \to J; K)$ is the probability that M halts after writing $g(X)$ as output. Thus, since g is an extension of f, M''' recognizes the domain of f. Furthermore, by Lemma 6.1, M''' can perform its computation in $NC^2(W) = NC^2(2^S)$ and thus in space S^2.

We have shown

PROPOSITION 6.4. *If a partial function can be computed by a probabilistic machine in space S, it can be computed in $NC^2(2^S)$.*

COROLLARY 6.5. *If a partial function can be computed by a probabilistic machine in logarithmic space, it can be computed in NC^2.*

7. CONCLUSION AND SOME OPEN PROBLEMS

Propositions 6.2 and 6.4, the simulation of an S space bounded probabilistic Turing machine by an S^2 space bounded deterministic machine, provided the original motivation for this paper. It seems that the "technical machinery" needed to establish this result is more than one might anticipate given that so much of the general approach was already established. On the other hand, results such as Corollary 4.3 and Proposition 5.1 (concerning the Boolean parallel complexity of the determinant and stochastic completion problems) should be of independent interest. Moreover, since Boolean circuits are so basic to complexity theory, we feel that the formal concept of well-endowed rings (with some of the specific examples developed here) will prove useful in other applications.

One obvious question which we have not pursued is whether or not one could prove that certain rings are *not* well endowed. Of course, it would be a breakthrough for complexity theory if one could show that a specific

function is not in NC^1 (even fixing the representation), but perhaps the simultaneous requirements on implementations for addition and negation (NC^0) and multiplication (NC^1) are sufficiently strong to induce a negative result (even over all succinct and uniform representations).

That the requirement on multiplication is not as severe as that of addition and negation is justified by the applications. However, it is interesting to consider the consequence of also insisting that multiplication have an implementation in NC^0. Would there be any interesting rings satisfying such a strong condition? For the ring of integers, with a standard $p$ary digit length function, we can obviously have a representation for which multiplication is implementable in NC^0, but at a terrible cost with respect to addition. Namely, represent an integer in terms of its prime factorization and then multiplication is reduced to the addition of exponents. Assuming a succinct and uniform representation (like the one just mentioned) for which multiplication can be implemented in NC^0, what is the best implementation for the addition of integers? In particular, can we prove that addition and multiplication cannot both be implemented in NC^0?

Sections 4 and 5 establish that the determinant and stochastic completion problems are in the Boolean class NC^2. As indicated before, any improvement in the Boolean depth required for either of these problems would imply a corresponding improvement for Savitch's (1970) well-known simulation. A number of perhaps more tractable problems are also worth pursuing. Our proof of Proposition 5.1 actually shows that the stochastic completion problem is NC^1 reducible to the determinant problems (the function V of Proposition 3.10 is clearly NC^1 reducible to integer determinants). What about the converse reduction? Section 6 shows that the stochastic completion problem is NC^1 hard for probabilistic log-space. Is the problem computable within this class, thus making it a complete problem for the class? Is it possible that the determinant problem is NC^1 hard (and hence complete) for NC^2?

Our model of a probabilistic machine is equivalent to one in which an infinite sequence of outcomes of independent unbiased coin flips is written on an orcle tape which is accessed by the machine through a one-way read-only oracle head. Is it possible to extend our simulation results to the case of a two-way read-only oracle head? What about the case of a two-way write-only output head?

RECEIVED APRIL 19, 1983; ACCEPTED November 29, 1983

REFERENCES

AVIZIENIS, A (1961), Signed-digit number representations for fast parallel arithmetic, *Inst. Radio Engr. Trans. Electron. Comput.* **10** 389–400.

BERKOWITZ, S. J. (1982), On computing the determinant in small parallel time using a small number of processors, preprint.

BORODIN, A. (1977), On relating time and space to size and depth, *SIAM J. Comput.* **6** 733–744.

BORODIN, A., VON ZUR GATHEN, J., AND HOPCROFT, J. (1982), Fast parallel matrix and GCD computations, *IEEE Sympos. Found. Comput. Sci.* **23**, 65–71.

BORODIN, A., AND MUNRO, I. (1975), "The Computational Complexity of Algebraic and Numeric Problems," Amer. Elsevier, New York.

COOK, S. A. (1981), Towards a complexity theory of synchronous parallel computation, *L'Enseignement mathématique* **XXVII**, 99–124.

CSANKY, L. (1976), Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **5**, 618–623.

GILL, J. (1977), Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6**, 675–695.

GILL, J., HUNT, J., AND SIMON, J. (1980), Deterministic simulation of tape-bounded probabilistic Turing machine transducers, *Theoret. Comput. Sci.* **12** 333–338.

HOPCROFT, J. E., AND ULLMAN, J. D. (1969), "Formal Languages and Their relation to Automata," Addison–Wesley, Reading, Mass.

HUNT, J. W. (1978), "Topics in Probabilistic Complexity," Ph.D. dissertation, Department of Electrical Engineering, Stanford University, 1978.

JUNG, H. (1981), Relationships between probabilistic and deterministic tape complexity, *in* "10th Sympos. on Mathematical Foundations of Computer Science, 1981," Lecture Notes in Computer Science No. 118, pp. 339–346, Springer-Verlag, Berlin/New York.

LADNER, R. E., AND FISCHER, M. J. (1980), Parallel prefix computation, *J. Assoc. Comput. Mach.* **27** 831–838.

OFMAN, YU. P. (1963), On the algorithmic complexity of discrete functions, *Soviet. Phys. Dokl.* **7**, 589–591.

PIPPENGER, N. (1979), On simultaneous resource bounds, *IEEE Sympos. Found. Comput. Sci.* **20** 307–311.

RUZZO, W. L. (1981), On uniform circuit complexity, *J. Comput. System Sci.* **22**, 365–383.

SAVAGE, J. E. (1976), "The Complexity of Computing," Wiley, New York.

SAVITCH, W. J. (1970), Relationships between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* **4**, 177–192.

SIMON, J. (1981a), On tape-bounded probabilistic Turing machine acceptors, *Theoret. Comput. Sci.* **16**, 75–91.

SIMON, J. (1981b), Space-bounded probabilistic Turing machine complexity classes are closed under complement, *ACM Sympos. Theory of Comput.* **13**, 158–167.

SIMON, J., GILL, J., AND HUNT, J. (1978), On tape-bounded probabilistic Turing machine transducers (extended abstract), *IEEE Sympos. Found. Comput. Sci.* **19** 107–112.

WINOGRAD, S. (1965), On the time required to perform addition, *J. Assoc. Comput. Mach.* **12**, 277–285.

WINOGRAD, S. (1967), On the time required to perform multiplication, *J. Assoc. Comput. Mach.* **14**, 793–802.