THE COMPLEXITY OF THE MAX WORD PROBLEM AND THE POWER OF ONE-WAY INTERACTIVE PROOF SYSTEMS

by

Anne Condon

Computer Sciences Technical Report #952

July 1990

The Complexity of the Max Word Problem

and the Power of One-way Interactive Proof Systems

Anne Condon *
Computer Science Department
University of Wisconsin-Madison

July 25, 1990

Abstract

We study the complexity of the *max word problem* for matrices, a variation of the well-known word problem for matrices. We show that the problem is *NP*-complete, and cannot be approximated within any constant factor, unless $P = NP$. We describe applications of this result to probabilistic finite state automata, rational series and k-regular sequences. Our proof is novel in that it employs the theory of interactive proof systems, rather than a standard reduction argument. As another consequence of our results, we characterize *NP* exactly in terms of *one-way* interactive proof systems.

1 Introduction

We study the *max word problem* for matrices, defined as follows. Given a finite set of $m \times m$ matrices, two m-vectors, v and w, a bound c and an integer k, is there a way to select a sequence of k matrices $M_1, \ldots, M_k$ (not necessarily distinct) from the set in such a way that the product $vM_1 \ldots M_k w^T > c$? We assume that all entries of the matrices and the vectors, as well as the bound c, are rational numbers expressed in binary and that k is an integer, expressed in unary notation. We describe applications of the max word problem in the theory of probabilistic finite state automata, rational series and k-regular sequences. We show that the max word problem for matrices is *NP*-complete and furthermore, that the corresponding optimization problem cannot be approximated within any constant factor, unless $P = NP$.

The max word problem is easily seen to be in *NP*. Our proof that it is *NP*-complete is based on properties of a special class of interactive proof systems that we call *one-way interactive proof systems*. These are protocols between a prover and a probabilistic verifier, in which all communication is in one direction, from the prover to the verifier. The verifier is computationally limited and the prover's goal is to convince the verifier that a common input is in some language. We denote by *oneway-IP(log,poly)* the class of languages accepted by one-way interactive proof systems where the verifier uses only log space and polynomial time.

We prove that the max word problem for matrices is *NP*-complete by showing that (i) any language in *NP* is in the class *oneway-IP(log,poly)* and (ii) any language in the class *oneway-IP(log,poly)* is polynomial time many-one reducible to the max word problem. An additional consequence of these

*Work supported by NSF grant number DCR-8402565

results is that NP = *oneway-IP(log,poly)*, giving a nice characterization of the class NP in terms of interactive proof systems.

In the rest of this section, we describe applications of the max word problem and outline the necessary background on interactive proof systems needed for the proof of the main result. We prove that the max word problem is NP-complete in Section 2 and also that NP = *oneway-IP(log,poly)*. In Section 3 we show that an optimization version of the max word problem cannot be approximated within any constant factor, unless $P = NP$.

1.1 Applications and Related Work

An instance of the max word problem for matrices is a tuple (S, v, w, k, c), where S is a set of $m \times m$ matrices, v and w are m-vectors, k is an integer and c is a constant. We assume that all entries of the matrices and the vectors, as well as the bound c, are rational numbers expressed in binary and that k is an integer, expressed in unary notation. The instance is a "yes-instance" if there a way to select a sequence of k matrices $M_1, \ldots, M_k$ (not necessarily distinct) from S in such a way that the product $vM_1 \ldots M_k w^T$ is greater than c.

The max word problem for matrices is a variation of the well-studied *word problem* for matrices, which is to determine if the product of a given list of matrices equals the identity matrix. Lipton and Zalcstein [6] showed that the word problem for matrices is in log space. They point out that the word problem for groups – the problem of deciding whether or not a product of group elements equals the identity element – is undecidable, a result proved by Novikov and Boone (see [9], Chapter 12). The max word problem is also a restriction of the following problem: given a finite set of matrices, two vectors, v and w and a bound c, is there some number k and matrices $M_1, \ldots, M_k$ in the set, so that the product $vM_1 \ldots M_k w^T$ is greater than c? This problem is actually undecidable; see Salomaa and Soittola ([10], Theorem II.12.1), for one proof.

We describe three applications of the NP-completeness of the max word problem for matrices in the theory of probabilistic finite state automata, rational series and k-regular sequences.

We consider probabilistic finite state automata (*pfa*'s) with rational transition probabilities, as defined in Paz [7] - we describe the model in detail in Section 2. Suppose we define the *k-emptiness problem* for pfa's as follows. Given a pfa and a number k, expressed in unary notation, does the pfa reject every string of length $\leq k$? By a simple reduction from the max word problem, we prove that the k-emptiness problem for pfa's is *co-NP*-complete.

This result also has implications for the complexity of the emptiness problem for pfa's with *bounded error*. Bounded error pfa's are those for which the probability that any string is accepted is bounded away from $1/2$ by some constant. Rabin [8] showed that bounded error pfa's accept exactly the regular languages. He proved this by showing that for any bounded error pfa with q states, there is an equivalent deterministic finite state automaton with $k = 2^{O(q)}$ states. Thus the pfa accepts some input if and only if it accepts an input of length at most k, and hence the emptiness problem for pfa's with bounded error can be decided by reduction to the k-emptiness problem for pfa's. We know of no more efficient decision procedure for the emptiness problem for pfa's with bounded error.

The second area in which the max word problem arises is in the study of *rational series*, a special class of formal power series. Salomaa and Soittola develop the theory in [10] and describe many applications to problems in automata theory, such as finding the density of regular languages or closure properties

of pfa's. For an alphabet Σ and a semi-ring A, a formal power series is a map σ from Σ^* into A. It is written as $\sigma = \sum_{x \in \Sigma^*} \sigma(x)x$. For concreteness, we suppose that A is the set of integers in this paper. The family of rational power series over the integers is a subclass of the class of formal power series and can be characterized in a number of ways, one in terms of matrices. The power series σ is *rational*, or equivalently, *admits a matrix representation* if there is an integer $m \geq 1$ and a map μ from Σ into the family of $m \times m$ matrices with integer entries, such that for all non-empty strings $x = x_1 \ldots x_n$ where each $x_i \in \Sigma$, $\sigma(x)$ is the $(1, n)$th entry of the matrix product $\mu(x_1) \ldots \mu(x_n)$.

There is a close relationship between rational series and regular languages. In fact, a language L is regular if and only if $L = \{x \in \Sigma^* \mid \sigma(x) \neq 0\}$, where σ is a rational power series that admits a matrix representation where the matrix entries are non-negative integers. Rational series generalize regular languages in a natural way: a language assigns a 0-1 value to each string in Σ^*, depending on whether the string is in L, whereas a rational series σ assigns a number, or *multiplicity*, $\sigma(x)$ to a string x. Thus rational series can be thought of as *regular languages with multiplicities*. Again it follows easily from the *NP*-completeness of the max word problem that the problem of finding strings of a rational series with high multiplicities is *NP*-complete. That is, it is *NP*-complete to decide, given the matrix representation of a rational series σ for alphabet Σ, and integers k and c, if there is a string in Σ^* of length $\leq k$ with multiplicity $\geq c$.

Allouche and Shallit [1] studied sequences closely related to rational series, which they call *k-regular sequences*. Think of the base k representation of a number n as a string over alphabet $\{x_0, x_1, \ldots, x_{k-1}\}$, with no leading zeroes and denote it by $base_k(n)$. Then a sequence $S(n)$ is k-regular if and only if $\sum S(n) base_k(n)$ is a rational series. Allouche and Shallit [1] list many natural examples of k-regular sequences. For example, the numerators of the left endpoints of the Cantor set is 2-regular, as is the sequence of numbers represented by the binary Gray code.

Consider the problem of finding *local maxima* of a k-regular sequence, that is, given a k-regular sequence $S(n)$, an integer j and an integer c, is $\max_{2^j \leq n < 2^{j+1}} S(n) > c$? Again, using the max word problem, we can show that the problem of finding local maxima of k-regular sequences, given their matrix representation, is *NP*-complete.

1.2 Background on one-way interactive proof systems

In this section we introduce some background on interactive proof systems that we will need in the proofs of Section 2. The model of an interactive proof system (*IPS*) was introduced by Goldwasser et al. [5]. In this paper, we describe a restricted class, called *one-way* interactive proof systems.

A one-way *IPS* consists of a verifier and a prover. The verifier is a probabilistic Turing machine with a 2-way, read-only input tape, a read-write work tape and a source of random bits (a coin). The states of the verifier are partitioned into reading and communication states. In addition, the Turing machine is augmented with a special communication cell that allows the prover to send information to the verifier. A transition function describes the one-step transitions of the verifier in the usual way when the verifier is in a reading state; there are two possible transitions each equally likely. Whenever the verifier is in a communication state, the next configuration is determined as follows. The prover writes a symbol in the communication cell, and, based on the state and the symbol written by the prover, the verifier's transition function defines the next state of the verifier.

The prover P is specified by a *prover transition function*, which is a function from inputs to strings over the prover's *alphabet*. For a fixed input x, the ith symbol in the string is the symbol written by

the prover the ith time the verifier enters a communication state on input x. Informally, we say that the prover *sends* this string to the verifier. This definition guarantees that the ith symbol the prover writes in the communication cell does not depend on the verifier's computation, but only on the input and i. The IPS is *one-way* because the verifier never writes in the communication cell, and so never communicates with the prover.

The pair (P, V) is a one-way interactive proof system for L with error probability $\epsilon < 1/2$ if

- for all $x \in L$, the probability that (P, V) accepts x is $> 1 - \epsilon$,
- for all $x \notin L$, and all provers P^*, the probability that (P^*, V) rejects x is $> 1 - \epsilon$.

An interactive proof system (P, V) is $s(n)$ space bounded if for all provers P^*, the number of work tape cells read or written by the verifier is $O(s(n))$, on any input of length n. Similarly, (P, V) is $t(n)$ time bounded if for all provers P^*, the number of transitions of the verifier is $O(t(n))$, on any input of length n. We denote by *oneway-IP(log,poly)* the class of languages accepted by one-way interactive proof systems that are simultaneously $\log n$ space bounded and *poly*(n) time bounded, for some polynomial *poly*. If a language L is in the class *oneway-IP(log,poly)*, then for *any* constant ϵ, $0 < \epsilon < 1/2$, there is a one-way IPS that accepts L with error probability ϵ, which is $\log n$ space bounded and *poly*(n) time bounded.

2 The max word problem and one-way interactive proof systems

In this section we show that any language in *oneway-IP(log,poly)* is polynomial time many-one reducible to the max word problem for matrices. We also show that $NP \subseteq$ *oneway-IP(log,poly)*. The NP-completeness of the max word problem follows from these two results.

Theorem 2.1 *Let L be any language in oneway-IP(log,poly). Then L is polynomial time many-one reducible to the max word problem for matrices.*

Proof: Suppose L is accepted by (P, V) that is $\log n$ space bounded and polynomial time bounded and has error probability ϵ. We use the following notation and assumptions in the proof. Let $t(n)$ be a polynomial bounding the running time of (P, V). Just as for Turing machines, a configuration of an interactive proof system for a fixed input is a tuple containing an encoding of the work tape, the positions of the tape heads on the input and work tapes, the state and the contents of the communication cell. We call a configuration of (P, V) that contains a communication state or reading state a communication configuration or reading configuration, respectively. Without loss of generality, we assume that the number of configurations of (P, V) on x is $2m$ for some m which is polynomial in n, where m are communication configurations and m are reading configurations. Assume that the initial configuration and all accepting configurations are communication configurations. Let the prover's alphabet be $\{a, b\}$.

Given any input x, we construct an instance $(\{A, B\}, v, w, k, c)$ of the max word problem as follows. We first define the two $m \times m$ matrices A and B. For $1 \leq i, j \leq m$, let p_{ija} be the probability of reaching communication configuration j from communication configuration i of V when the symbol a has just been written by a prover in the communication cell. Note that this probability is completely determined by x, i, j, a and the transition function of V. Let $A = [p_{ija}]$. Define $B = [p_{ijb}]$ similarly,

replacing a everywhere by b. The vector v has all 0 entries except for the entry corresponding to the initial configuration, which is 1, and the vector w has all 0 entries except for the positions corresponding to accepting configurations, which have the entry 1. Finally, let $k = t(|x|)$ and let $c = 1 - \epsilon$.

This reduction can be computed in polynomial time; in particular, all entries in A and B can be written as rational numbers of the form p/q where $p \leq q \leq 2^{m+1}$. The proof of this is very similar to a proof of Gill ([4], Theorem 6.4) on the transition probabilities of $\log n$ space bounded probabilistic Turing machines. The entry p_{ija} can be computed in the following way. Let C be an ordered set, consisting of the reading configurations, plus the ith communication configuration, and let Q be the $(m+1) \times (m+1)$ probability transition matrix between these configurations, defined as follows. The transition probabilities between reading configurations are given by the transition function of V. The transition probabilities from i to the reading configurations are given by the transition function of V from configuration i, when the prover writes an a in the communication cell. Define the transition probabilities from reading configurations to i to be 0.

Let $\bar{v}$ be the $(m+1)$-vector whose entries are the probabilities that j is the first communication configuration reached from the configurations in C. In particular, the entry of $\bar{v}$ for configuration i is p_{ija}. Then $\bar{v}$ satisfies the equation $\bar{v} = Q\bar{v} + \bar{b}$, or equivalently, $2(I - Q)\bar{v} = 2\bar{b}$, where $\bar{b}$ is the vector whose entries are the probabilities of reaching communication configuration j in 1 step from the configurations of C. Moreover, $2(I - Q)$ is invertible, so by Cramer's rule, each entry e of $\bar{v}$ can be expressed as the quotient of two integers N_e/D where D is the determinant of $2(I - Q)$ and $N_e \leq D$. Also, it is straightforward to see from the definition of Q that each row of $2(I - Q)$ has a constant number of non-zero entries, which are integers whose absolute values sum to at most 4. Using this fact and expansion by minors, it can be shown by induction that the determinant of $2(I - Q)$ is at most 2^{m+1}. Thus p_{ija} can be written in the form p/q where $p \leq q \leq 2^{m+1}$, and these entries can be computed from Q and $\bar{b}$ in polynomial time.

To complete the proof, we show that (P, V) accepts x if and only if $(\{A, B\}, v, w, k, c)$ is a yes-instance of the max word problem. Fix any prover P^*, and suppose $P^*(x) = \alpha_1 \ldots \alpha_k \in \{a, b\}^*$. Let $X_i = A$ or B if $\alpha_i = a$ or b respectively, for $1 \leq i \leq t$. Then the probability that the lth communication configuration entered by (P^*, V) is j is the $(1, j)$th entry of the product $X_1 \ldots X_l$. This can be proved by induction on l. Hence the probability that (P^*, V) accepts x is $vX_1 \ldots X_k w$. If $x \in L$, then (P, V) accepts x with probability $> 1 - \epsilon$. Hence $(\{A, B\}, v, w, k, c)$ is a yes-instance of the max word problem. However, if $x \notin L$, then for all provers P^*, the probability that (P^*, V) accepts x is $< \epsilon$. Hence $(\{A, B\}, v, w, k, c)$ is not a yes-instance of the max word problem, as required. □

In the next theorem, we show that *NP* $\subseteq$ *oneway-IP(log,poly)*. The proof is a refinement of a proof of Condon ([2], Theorem 2) that *IP(poly)* $\subseteq$ *IP(log,poly)*. Here *IP(poly)* and *IP(log,poly)* are the classes of languages accepted by interactive proof systems, not necessarily one-way, that are polynomially time bounded, and simultaneously polynomially time bounded and log space bounded, respectively. We give a brief summary of the proof of the theorem; the details are similar to those of [2].

Theorem 2.2 *NP* $\subseteq$ *oneway-IP(log,poly)*

Proof: Let L be in *NP*, and suppose that L is accepted by a nondeterministic Turing machine M, with one worktape, which runs in time $t(n)$. We construct a one-way IPS (P, V) that accepts L. The idea of the construction is that on a fixed input x, the prover P repeatedly sends to V a computation of M on x. V checks that on every repetition, the computation sent by the prover is a valid, accepting computation, and in that case accepts x. Using only $O(\log n)$ space, the verifier cannot store the complete computation in order to check that it is valid, but instead randomly chooses symbols to check.

Let x be an input of length n. A computation of M on x is a string $m_0 a_1 m_1 \ldots a_i m_i \ldots m_{t(n)}$, where each m_i is a configuration and m_0 is the initial configuration. Each $a_i \in \{1,2\}$ and the $a_i th$ possible next configuration from m_{i-1} is m_i, according to the transition function of M. Since M has one worktape, each configuration m_i can be represented as a string $c_1 \ldots c_{k-1}\ q\ c_k \ldots c_{t(n)}$, where q is a state of M, $c_1 \ldots c_{t(n)}$ represents the contents of the worktape and the tape head is positioned on the kth tape cell. Each c_i is either an input symbol, a worktape symbol or a special blank symbol.

The verifier V checks that a string sent by the prover is the concatenation of $dt(n)$ accepting computations of M, where d is a constant. V can easily verify in $O(\log n)$ space that the string sent by the prover is *syntactically* correct. That is, each computation is composed of $t(n)+1$ configurations m_i separated by strings $a_i \in \{1,2\}$, that each m_i has length $t(n)+1$, that the first configuration is the initial configuration of M on x and the last is an accepting configuration.

The verifier must also check that the $a_i th$ possible next configuration from m_{i-1} is m_i. To do this, V checks *one symbol* of each configuration. We say the kth symbol of m_i is *valid* if it is consistent with the symbols at positions $k-1, k, k+1$ and $k+2$ of configuration m_{i-1} and a_i, and the transition function of M. To check that the kth symbol of m_i is valid, where $i > 0$, V stores on a tape the four symbols numbered $k-1, k, k+1, k+2$ of configuration m_{i-1}, together with k and a_i. In doing this, the verifier uses $O(\log t(n))$ space to store k and constant space to store a_i and the four symbols. For each of the computations sent by the prover, V randomly chooses a number k in the range from 1 to $t(n)+1$, and checks the kth symbol in each configuration of that computation.

It is easy to see that if M accepts x, then there is a prover P such that (P,V) accepts x with probability 1. Suppose that x is not accepted by M, and let P^* be any prover. We show that (P^*,V) accepts x with probability at most $1/4$. If the string P^* sends to V is not syntactically correct, then (P^*,V) rejects x with probability 1. Otherwise, on each of the $dt(n)$ computations, some configuration contains an invalid symbol. The probability that V detects it is at least $1/(t(n)+1)$. Hence the probability that V accepts x is at most $(1-1/(t(n)+1))^{dt(n)}$. Choose d so that this quantity is at most $1/4$. □

Corollaries 2.1 and 2.2 below follow immediately from the last two theorems.

Corollary 2.1 *NP= oneway-IP(log,poly).*

Proof: The direction *NP* $\subseteq$ *oneway-IP(log,poly)* is proved in Theorem 2.2. To see that the other direction holds, note that for any pair of languages L_1 and L_2, if L_1 is many-one reducible to L_2 and $L_2 \in NP$, then $L_1 \in NP$. From Theorem 2.1 every language in *oneway-IP(log,poly)* is many-one reducible to the max word problem, which is in *NP*. □

Corollary 2.2 *The max word problem is NP-complete.*

In fact, the theorems show that the max word problem is *NP*-complete even when the set of matrices is restricted to 2 stochastic matrices, that is, the entries in each row of the matrices are nonnegative and sum to 1.

To illustrate one application of the *NP*-completeness of the max word problem for matrices, we include the proof that the k-emptiness problem for probabilistic finite state automata (pfa's) is *co-NP*-complete.

Corollary 2.3 *The k-emptiness problem for probabilistic finite state automata is co-NP-complete.*

Proof: We first describe our model of a probabilistic finite state automaton (pfa). A *pfa* is a finite state automaton with a 1-way input tape with probabilistic transitions between states. That is, each

letter of the alphabet of the pfa corresponds to a probabilistic transition matrix of dimension $q \times q$, where q is the number of states of the pfa. The probability that a string $x = x_1 \ldots x_n$ is accepted is $vM_{x_1} \ldots M_{x_n} w^T$, where M_{x_i} is a matrix corresponding to the symbol x_i, v is the vector with a 1 in the position corresponding to the start vertex and 0's everywhere else, and w is the vector with a 1 in each position corresponding to a final state and 0's everywhere else. Associated with a pfa is a *cut point*, which is a rational number between 0 and 1, that determines the language accepted by the pfa as follows. A string is in the language if and only if the probability that the pfa accepts that string is greater than the cut point of the pfa.

To prove our result, we show how the max word problem can be reduced to the complement of the k-emptiness problem. Let an instance of the max word problem be given by a finite set S of $m \times m$ matrices, m-vectors v and w, an integer k expressed in unary notation and a bound c. From the proof of Theorem 2.1, we can assume that the set S consists of two stochastic matrices $\{A, B\}$, that $0 < c < 1$ and that v has only one non-zero entry. Then the states of the corresponding pfa are $\{1, \ldots, m\}$, the alphabet is $\{a, b\}$ and the matrices A and B define the probability transitions of the pfa on a and b, respectively. The cut point of the pfa is c. For any $i, 1 \leq i \leq m$, i is the initial state if the ith entry of v is non-zero and i is a final state if the ith entry of w is non-zero. The resulting pfa accepts a string of length $\leq k$ if and only if the instance is a yes-instance of the max word problem. □

3 Approximation

The optimization (maximization) version of the max word problem for matrices is: given a finite set S of $m \times m$ matrices, two m-vectors, v and w, and an integer k, output the maximum of $vM_1 \ldots M_k w^T$, for all choices of M_i from S. We call this maximum, for an instance I of the max word optimization problem, the *solution* of I and denote it by $soln(I)$. In this section, we show that this optimization problem cannot be approximated within any constant factor in polynomial time, unless $P = NP$. By this we mean that for any constant $C > 1$, there is no polynomial time algorithm that, given any instance I of the max word optimization problem, outputs a value in the interval $[soln(I)/C, Csoln(I)]$.

Theorem 3.1 *The max word optimization problem for matrices cannot be approximated within any constant factor in polynomial time, unless $P = NP$.*

Proof: Suppose that for some constant $C > 1$ there is a polynomial time approximation algorithm for the max word optimization problem. To prove the theorem, we show that under this assumption, if L is any language in NP then $L \in P$.

Let M be a one-way IPS for L with error probability ϵ which is less than $1/(1 + C^2)$. From Theorem 2.1, given any instance x of L, there is a polynomial time computable reduction (constructed using M) that maps x to an instance I_x of the max word optimization problem. This reduction has the property that if $x \in L$, then $1 - \epsilon < soln(I_x) \leq 1$, whereas if $x \notin L$ then $0 \leq soln(I_x) < \epsilon$. Hence the approximation algorithm A outputs a number in the interval $[(1 - \epsilon)/C, 1]$ if $x \in L$ and in the interval $[0, C\epsilon]$, if $x \notin L$. These intervals do not intersect, by our choice of ϵ. Hence the output of the approximation algorithm determines conclusively whether $x \in L$. Also the algorithm runs in polynomial time. Hence $L \in P$. □

4 Acknowledgement

Jeff Shallit introduced me to the max word problem and its applications. Thanks also to Jeff and to Eric Bach for a careful review of an earlier draft of this paper. Their comments much improved the presentation.

References

[1] J.-P. Allouche and J. Shallit, The Ring of k-regular Sequences, Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag Lecture Notes in Computer Science, Number 415 February 1990, pp 12-23.

[2] A. Condon, Space Bounded Probabilistic Game Automata, Proceedings of the Third Annual conference on Structure in Complexity Theory, June 1988, pp 162-174. To appear in the *Journal of the Association for Computing Machinery.*

[3] A. Condon and R. J. Lipton, Upper Bounds on the Complexity of Space Bounded Interactive Proofs, Technical Report 841, Computer Sciences Department, University of Wisconsin-Madison, April 1989.

[4] J. Gill, Computational Complexity of Probabilistic Turing Machines, *SIAM Journal on Computing*, **6, No. 4**, 1977, pps 675-695.

[5] S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive proof systems, Proceedings of 17th Symposium of the Theory of Computing (STOC), 1985, pp 291-304.

[6] R. J. Lipton and Y. Zalcstein, Word Problems Solvable in Logspace, *Journal of the Association for Computing Machinery*, **24, No. 3**, July 1977, pp 522-526.

[7] A. Paz, *Introduction to Probabilistic Automata*, Academic Press, 1971.

[8] M. O. Rabin, Probabilistic Automata, *Information and Control* **6**, 1963, pp 230-245.

[9] J. Rotman, *The Theory of Groups: An Introduction.* Allyn and Bacon, Boston, second ed., 1973.

[10] A. Salomaa and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*, Springer-Verlag, 1978.