STRUCTURE OF COMPLEXITY IN THE WEAK MONADIC SECOND-ORDER THEORIES OF THE NATURAL NUMBERS

by

Edward L. Robertson

Research Report CS-73-31

Department of Applied Analysis and
Computer Science

University of Waterloo

Waterloo, Ontario, Canada

December, 1973

STRUCTURE OF COMPLEXITY IN THE WEAK MONADIC SECOND-ORDER THEORIES OF THE NATURAL NUMBERS

Edward L. Robertson

ABSTRACT

The complexity of decision procedures for the Weak Monadic Second-Order Theories of the Natural Numbers are considered. If only successor is allowed as a primitive, then every alternation of second-order quantifiers causes an exponential increase in the complexity of deciding the validity of a formula. Thus a heirarchy similar in form to Kleene's arithmetic heirarchy may be shown to correspond to the Ritchie functions. On the other hand, if first-order less-than is allowed as a primitive, one existential quantifier suffices for arbitrarily complex (in the Ritchie heirarchy) decision problems. This leads to a normal form, in which every sentence in the theory is equivalent in polynomial time to a sentence with less-than but only one existential second-order quantifier.

KEY WORDS:

Weak Monadic Second-Order Theories, decision procedures, computational complexity, Ritchie heirarchy.

ACKNOWLEDGEMENTS

The majority of this research was conducted at the University of Wisconsin with the support of the N.S.F. Grant No. GJ-33087. It was written up at the University of Waterloo with the support of the University Research Committee and the N.R.C. Grant No. A8653.

The author wishes to express his gratitude and appreciation to Prof. L. H. Landweber for many stimulating conversations and discerning comments.

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. THE WEAK MONADIC SECOND-ORDER THEORY	5
III. THE SIMULATION OF A MACHINE COMPUTATION	15
IV. MANIPULATION OF FORMULAS INTO SPECIAL FORMS.	20
V. CONCLUSION	41
BIBLIOGRAPHY	49

I. INTRODUCTION

In [6], Meyer presents a method for reducing the decision problem of any set whose characteristic function is elementary to the problem of deciding whether a formula in weak monadic second-order theory of one successor (WMST(')) is valid. The length of the formula in WMST(') is only polynomial in the length of the input, thus providing a reduction of any elementary computation to a sentence which is polynomial in the length of the input.

Notation For any string α, $|\alpha|$ denotes the length of α. This is used both for strings over an input alphabet and for formulas in logical languages.

Definition: Let $\mathcal{T}$ be a logical theory, and $\mathcal{C}$ a class of sets. Then $\mathcal{C}$ is polynomial expressable in $\mathcal{T}$ if, for every $S \in \mathcal{C}$ there is some Turing-machine $\mathcal{M}$ such that

1) S is the set accepted by $\mathcal{M}$.
2) for every input α there is a formula Φ_α (or $\Phi_{\mathcal{M},\alpha}$) such that Φ_α is valid in $\mathcal{T}$ iff $\mathcal{M}$ accepts α.
3) there is a polynomial p such that, given α, Φ_α may be constructed within $p(|\alpha|)$ time, and hence $|\Phi_\alpha| \leq p(|\alpha|)$.

The relevant time and space bounds are the well-know "Ritchie functions" [8] f_k, defined inductively by

$$f_0(n) = n$$
$$f_{i+1}(n) = 2^{f_i(n)} \quad ;$$

so that $f_k(n)$ is the kth exponential of n, that is

$$2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}} \quad \left.\right\} \text{ k times}$$

We use $\mathcal{F}_k$ to denote the class of all sets S, such that S is accepted by some (possibly non-deterministic) Turing machine which operates within space and time bounded by $f_k(c \cdot n)$, where c is an arbitrary constant and n is the length of the input. This definition varies from Ritchie's on several technical details: time as well as space is bounded, the classes are of sets (acceptance problems) rather than functions. However, the basic feature of Ritchie's definition - that $\mathcal{F}_{k+1}$ is a class which is exponentially harder than $\mathcal{F}_k$ - remains unchanged. Also unchanged is Ritchie's result that the union of all the classes $\mathcal{F}_k$ is the class computable in space and time bounded by the elementary functions [8]. It is easy to see that any function computed within time bounded by a polynomial in f_k is in $\mathcal{F}_k$, for $k > 1$.

The proof that WMST is not decidable in elementary time involves showing that each class $\mathcal{F}_k$ is expressable in WMST. If the set of true sentences of WMST were indeed elementary, then by Ritchie's result it would belong to $\mathcal{F}_k$ for some k. But then $\mathcal{F}_{k+1}$ would be reducible to $\mathcal{F}_k$, contradicting the strict heirarchy of the $\mathcal{F}_k$'s.

Meyer's construction employs the extremely elegant but opaque device of "γ-expressions". In reading Meyer's proof, one suspects there is an intimate relation between the structure of WMST formulas and the Ritchie functions.

In this paper we exhibit that relationship directly. Given any set S in $\mathcal{F}_k$ and any input α, we construct the formula Φ_α. Moreover, Φ_α may be seen to be in a standard form, with all second order quantifiers out front, with k-1 alternations of second order quantifiers, and beginning with a second-order existential quantifier - i.e. Φ_α is in the second-order analogue of Σ_k-form of the Kleene heirarchy.

There are three major sections to this paper. Section II formally introduces WMST in several variations. The main result of Section II is an important tool, which essentially allows the expression of congruence modulo $f_k(n)$ in a formula whose length is merely polynomial in n. Section III puts this tool to work, describing the construction the formulas Φ_α which simulate the computation of on input α. The construction of $\Phi_{\mathcal{M},\alpha}$ will be done in a direct manner, using a method similar to that of Cook [2] for describing the computation of on α. This section therfore provides a direct proof of Meyer's result.

Section IV examines the results of the previous two, manipulating the formulas to relate their structure to the computational complexity of the machine described. In particular, a heirarchy similar to the Kleene-heirarchy is shown to exist in WMST with only successor as a primitive. If < is also allowed as a primitive, the heirarchy disappears entirely.

The concluding section briefly mentions the complexity of specific algorithms for deciding various theories. With a slight modification, the heirarchy of section IV is claimed to be one-one. Finally, these and previous results are used to achieve normal form for sentences of the language of WMST plus the primitive relation less-than such that any formula can be transformed into this normal form within time and space bounded by a polynomial in the length of the formula.

II. THE WEAK MONADIC SECOND-ORDER THEORY

A monadic second-order language includes the usual logical connectives, first and second order variables and quantifiers for these variables, and monadic (unary) applications of second-order variables to first-order ones. In addition, the language may be augmented with other non-logical symbols representing first-order constants or fixed functions or predicates on first-order variables.

If the first-order variables are interpreted as members of N (the natural numbers), a second-order variable then clearly corresponds to a subset of N - that particular subset for which the predicate is true. This interpretation of a monadic second-order language is denoted by $MST(N, p_1, \cdots, p_m)$, where the p_i are interpretations for the non-logical symbols. If the interpretation is further restricted so that second-order variables are quantified only over finite subsets of N (predicates true for only finitely many individuals), the resulting theory is the weak monadic second-order theory of natural numbers with primitives $p_1, \cdots, p_m$, denoted $WMST(N, p_1, \cdots, p_m)$.

Conventions: Upper-case Roman letters will be used for second-order variables, and will be called sets or set variables, while lower-case Roman letters will be used for first-order or individual variables. Since individuals will always be interpreted as elements of N and sets as finite subsets of N , we drop "N" from the designation of any particular theory. Also, with this interpretation in mind, the application of a set variable (say X) to an individual variable (y) will be denoted by set-theoretic notation ("$y \in X$"). The distinction between a theory and its language is often ignored.

The additional functions and predicates which we shall consider are the successor function (the successor of x is $x+1$, written "x'"), equality, less-than, and congruence modulo a constant (congruence modulo a variable yields an unsolvable theory). The symbols $'$, $=$, $<$, and $\equiv$ will be used both as non-logical symbols of the language and in denoting particular theories. Informally arbitrary numerical constants will be used, but the constant c will be expressed in the formalism by 0 followed by c successors. Thus, since some designation of successor will always be available, "0" is the only numerical constant formally required.

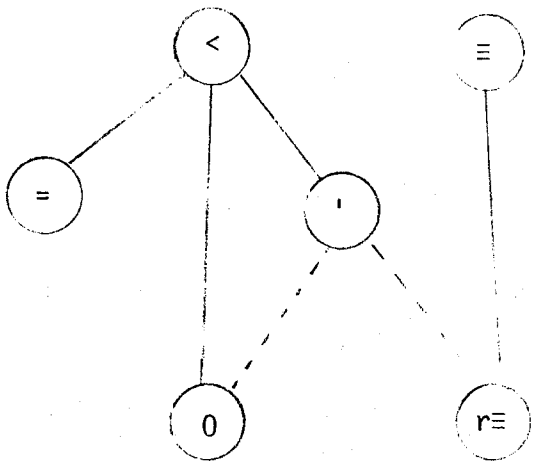
Formulas involving $\equiv$ will often be additionally restricted to require that only one modulus be used (or that there is one modulus divisible by all other moduli). Although this restriction is not on the primitives of the language, we will use "r≡" in denoting theories whose languages obey this restriction (such as "WMST(r≡)") and "restricted congruence" to refer to this restriction.

Previous investigations have concentrated on WMST(') , since any formula true in WMST(0,',=,≡,<) is equivalent to one in WMST('). For example, "$x < y$" may be expressed as

$$(\forall Z)[((\forall z)[z' \in Z \Rightarrow z \in Z] \;\&\; y \in Z) \Rightarrow x \in Z] \; .$$

As may be seen, expressing $<$ in terms of $'$ requires a second-order quantifier, as is also true of $=$ and $\equiv$.

Definition: A primitive β is directly expressible in terms of a primitive γ if every formula in β may be translated into an equivalent formula in γ but not β without introducing additional set variables. Then $'$ and $=$ are directly expressible in terms of $<$.



Direct expressibility relations between non-logical primatives indicated by downward lines (dashed relations discussed in subsequent sections).

Within the language of any particular theory, we will use many informal abbreviations. In particular, we will use as an abbreviation any relation directly expressible in the primatives of the theory, for example using $<$, $=$, and $'$ in WMST($<$), or "+k" and "-k" (k constant) as abbreviations involving successor.+

+ In order to avoid exponential growth in de-abbreviation, we must by cautious and interpret any expression involving $y-1$ as false for $y = 0$ unless it is explicity in the form

$$[y = 0 \;\&\; P(y)] \vee [y > 0 \;\&\; Q(y-1)]$$

as its negation.

For the moment we restrict ourselves to $WMST(<, \equiv)$ and show how to define congruence for very large constants with quite short and structurally simple formulas. Rather than using the functions f_i, we define slight variations g_i such that

$$g_0(n) = n \quad (= f_0(n))$$

$$g_{i+1}(n) = g_i(n)\, 2^{g_i(n)} \quad .$$

As with the functions f_i, a function g_k may be thought of as k applications of g_1. The functions g_k are introduced for notational and conceptual convenience, and obiously delimit the same complexity classes, as is shown by the following.

Lemma 2.1 For $n > 0$, any i, $f_i(n) \leq g_i(n) \leq f_i(4n)$.

Proof: Obviously $f_i \leq g_i$.

The other inequality is shown by induction on i for the stronger condition $4g_i(n) \leq f_i(4 \cdot n)$. For $i = 0$, equality holds by definition. Now assume, for some i, that $4g_i(n) \leq f_i(4\ n)$ and show the same for $i+1$. But

$$4g_{i+1}(n) = 2 \cdot 2 \cdot g_i(n) \cdot 2^{g_i(n)}$$

and each of these factors is less than or equal to $2^{g_i(n)}$. Hence

$$4g_{i+1}(n) \leq (2^{g_i(n)})^4 = 2^{4g_i(n)} \leq 2^{f_i(4n)} = f_{i+1}(4n) \quad .$$

□

Congruence Representation Lemma (2.2): The relation $x \equiv y \mod g_k(n)$ [†] can be expressed in $WMST(<, \equiv)$ by a formula whose length is linear in n but exponential in i.

Proof: The remainder of this section is devoted to the construction of the required formula. This is done by inducting on k for each fixed constant n.

I. Basis

$$"x \equiv y \mod g_0(n)"$$

is simply an instance of primitives.

II. Induction

Given "$x \equiv y \mod m$", define the relation "$x \equiv y \mod m \cdot 2^m$". Repeated application of this inductive step k times clearly gives congruence modulo $g_k(n)$.

First we construct certain important relationships, which will be abbreviated by the indicated expressions.

a) "$x = y + m$":

$$x > y \;\&\; x \equiv y \mod m \;\&\; (\forall t)[x < t < y \Rightarrow t \not\equiv y \mod m].$$

b) from (a) and constants obtain "$x = m$", "$x < m$", etc.

c) "$y = {}_m Lx$", signifying that y is the greatest value $\leq x$ which is congruent to $0 \mod m$.

[†] In this and all future cases, the modulus of a congruence relation is a constant, in this case a constant in n. The other arguments are usually variables.

$$y \leq x \;\&\; y \equiv 0 \mod m \;\&\; (t)[x > t > y \Rightarrow t \not\equiv 0 \mod m]$$

We will sometimes further abbreviate, omitting individual quantifiers and writing "x+n" and "$\lfloor x$" as functions. Similarly, we define $\lceil x = \lfloor x + m$ (the least y great than but not equal to x such that $y \equiv 0 \mod m$).

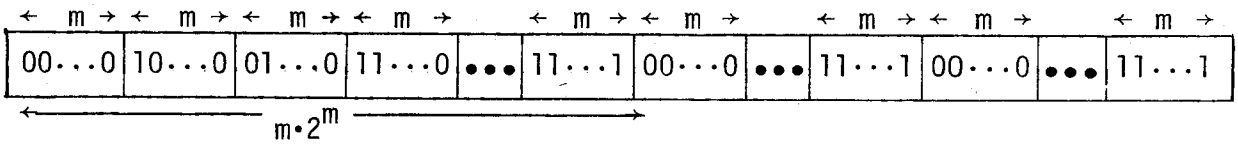
The construction used in the induction step, and also many other constructions in this paper, involve encoding binary sequences in sets such that the set X represents a binary seuqnece which is 1 in the yth position iff $y \in X$. A reference to "sequence" thus means a set interpreted in this manner. Such sequences are technically infinite, but since the sets encoding them are finite, they become only zero after a certain point.

A m-interval is a portion of a sequence which has indicies $cm, cm+1, \ldots, cm+m-1$, for some c - that is, it is m bits long and begins at a position congruent to zero modulo m. An m-interval-sequence is a sequence in which successive intervals encode, in order, the binary representations of the numbers 0 thru 2^m-1, or some number of repetitions of this pattern. Observe that encoding 0 thru 2^m-1 requires $m \cdot 2^m$ bits, whence the definition of the functions g_k.

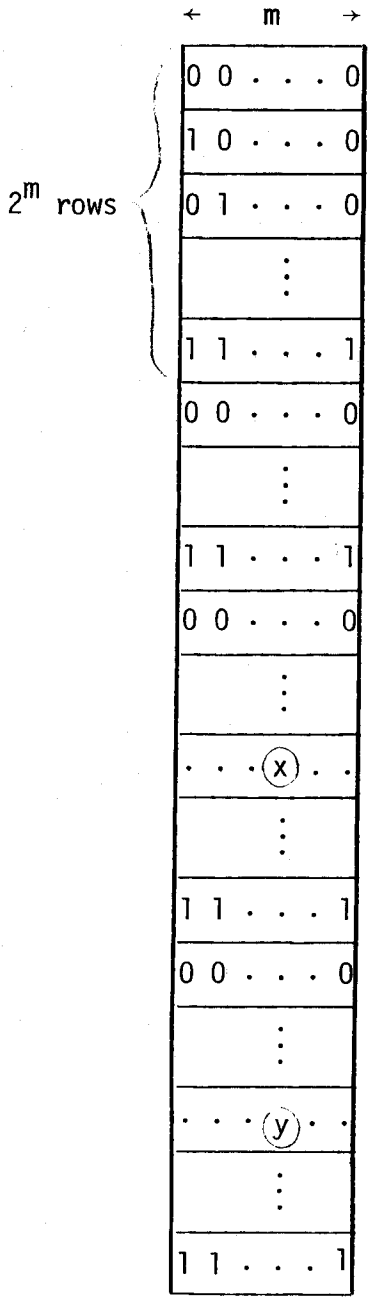
Given arbitrary x and y, it is possible to determine whether $x \equiv y \mod m \cdot 2^m$ by looking at those m-intervals of a sufficiently large m-interval sequence which contains the xth and yth bits. In particular, x is congruent to y modulo $m \cdot 2^m$ if the m-intervals

Figure 1 The use of m-intervals to determine $x \equiv y \bmod m \cdot 2^m$

1a) an m-interval-sequence:



1b) slicing up the m-interval-sequence and laying rows in parallel:



(x) and (y) indicate the x-th y-th positions of the sequence respectively

$x \equiv y \bmod m \cdot 2^m$ iff

i) (x) and (y) are in same columns

ii) rows containing (x) and (y) have same pattern

represent the same binary number and x and y have the same position in their respective m-intervals (i.e. $x \equiv y \bmod m$). Figure 1 indicates the idea of the construction.

It is now necessary to make a short aside in order to clarify notation. Since we will later wish to distinguish between various formal representations of the same relationship, we will designate a relationship with script letters, while the same name in Roman will designate formulas representing the relationship, with subscripts to distinguish different formal representations. For example, the relation m-$\mathcal{IS}_1(X)$, which is true iff X is an m-interval sequence, will be represented by formulas $m\text{-}ISeq_1(X)$ and $m\text{-}ISeq_2(X)$. It should be noted that m is a constant involved in the construction of $m\text{-}ISeq_1$, and hence this notation is more appropriate than listing m as if it were a parameter of the formula in the same way as free variables.

The formulas expressing m-$\mathcal{IS}_1(X)$, denoted $m\text{-}ISeq_0(X)$, is a conjunction of four clauses:

c1) X starts with 0^m:

$$(\forall y)[y < m \Rightarrow y \notin X]$$

c2) 0 and 1 alternate in positions congruent to 0 mod m:

$$(\forall y)[y \equiv 0 \mod m \ \& \ y+m \leq \max(X) \Rightarrow (y \in X \Longleftrightarrow y+m \notin X)]$$

c3) carry propigates correctly:

$$(\forall y)\left\{\begin{array}{l} \lceil y \leq \max(X) \ \& \ (y \in X \Longleftrightarrow y+m \notin X) \\ \Longleftrightarrow (\forall z)\lceil y > z \geq \lfloor y \Rightarrow z \in X] \end{array}\right\}$$

c4) X ends with 1^m :

$$\max(X) \equiv m\text{-}1 \bmod m \;\&\; (\forall y)[\max(X) \geq y \geq \lfloor \max(X) \Rightarrow y \in X]$$

Finally, we construct $\mathcal{M}od(X, x, y)$ which is a predicate such that $x \equiv y \bmod m\cdot 2^m$ iff $(\exists X)[m\text{-}\mathcal{S}eq(X) \;\&\; m\cdot 2^m - \mathcal{M}od(X, x, y)]$. That is, if X is an m-sequence, $m\cdot 2^m - \mathcal{M}od(X, x, y)$ will hold iff x and y are less than $\max(X)$ and $x \equiv y \bmod m\cdot 2^m$. We take this approach rather than directly defining congruence modulo $m\cdot 2^m$ since we will later wish to use the same m-sequence in expressing a number of specific congruence relations. We define $m\cdot 2^m - \mathrm{Mod}_0(X, x, z)$:

$$y \leq \max(X) \;\&\; z \leq \max(X) \;\&\; y \equiv z \bmod m \;\&$$

$$(\forall u)(\forall v)\left[\, u \equiv v \bmod m \;\&\; \lceil y > u \geq \lfloor y \;\&\; \lceil z > v \geq \lfloor z \;\Rightarrow\; (u \in X \Leftrightarrow v \in X) \,\right]$$

The formal definition of $g_i(n)\text{—ISeq}_1$ and $g_{i+1}\text{-Mod}_1$ is done inductively such that $g_0(n)\text{—ISeq}_1$ and $g_1(n)\text{-Mod}_1$ are exactly as given above with congruence as a primitive. For $i > 0$, $g_i(n)\text{-ISeq}_1$ and $g_{i+1}\text{-Mod}_1$ are defined by substituting

$$(\exists X)[g_{i-1}(n)\text{—ISeq}_1(X) \;\&\; g_i(n)\text{—Mod}_1(X, x, z)]$$

for each occurance of $y \equiv z \bmod g_i(n)$ in $g_i(n)\text{-ISeq}_0$ and $g_{i+1}(n)\text{-Mod}_0$ respectively.

Thus we complete defining congruence modulo $g_k(n)$ for any fixed k and n. The above expressions are cumbersome but not difficult, and the reader may easily verify that they express the desired relations. Furthermore, simple substitution for abbreviations shows that 9 uses of congruence modulo m are required to define $m\text{-ISeq}_1$ and 12 to define $m\cdot 2^m\text{-Mod}_1$ (with minor cleverness these may be reduced to 7 and 8). Hence, if congruence modulo n requires $d\cdot n$ symbols to define for fixed n, and $\equiv \bmod m\, 2^m$ is definable from $\equiv \bmod m$ using $c + 2l\{\text{size of defn of} \equiv \bmod m\}$ symbols, and we may define $x \equiv y \bmod g_k(n)$ using

$$(2l^k d)n + c\cdot \sum_{i=0}^{k-1} 2l^i$$

symbols - linear in n but exponential in k.

It is clear that only congruence modulo n is required to express congruence modulo $g_k(n)$, and if the only modulus of a particular sentence is $g_k(n)$ (or $g_{k_1}(n)$, $g_{k_2}(n)$, $\cdots$), then that sentence is expressible with restricted congruence via this lemma.

□

III. THE SIMULATION OF A MACHINE COMPUTATION

As in Cook [2], we use logical expressions to describe the valid computations of Turing Machines, and existential quantifiers to assert that a valid computation exists. Because of the length constraint, we are not able to code each tape-square time-instance by specific expressions, but instead we must encode this information in set variables.

Theorem 3.1 For any $k \geq 0$, $\mathcal{F}_{k+1}$ is polynomial expressible in $WMST(<, \equiv)$.

Proof: Pick any $S \in \mathcal{F}_{k+1}$, and a machine $\mathcal{M}$ which decides $\alpha \in S$ in time $f_{k+1}(c\cdot|\alpha|)$, for some constant c. We fix the input α and construct $\Phi_{m,\alpha}$. Let $n = c\cdot|\alpha|$. For reasons noted in the previous section, we will simulate a time bound of $2^{g_k(n)}$ - without significant change in the situation.

Say $\mathcal{M}$ operates on vocabulary Σ and has states Q. We then write the state of $\mathcal{M}$ on the tape cell which M's head is currently scanning (assuming $\Sigma \cap (Q \times \Sigma) = \emptyset$ and denoting $\Sigma \cup (Q \times \Sigma)$ by $\hat{\Sigma}$). With this well-known modification, we represent the computation of $\mathcal{M}$ as a sequence of transformations on strings over $(\hat{\Sigma})^{2^{g_k(n)}}$. The initial and accept states of $\mathcal{M}$ are q_0 and q_A; and $\mathcal{M}$ is assumed to begin on the left most square of the input, and never to move left of that square. Thus the initial string on input $\alpha = \alpha_1 \cdots \alpha_n$ is

$$\langle q_0, \alpha_1 \rangle \; \alpha_2 \; . \; . \; . \; \alpha_n \;\; \beta \; . \; . \; . \; \beta \; . \; . \; . \; \beta$$
$$\left| \leftarrow 2^{g_n(n)} - n \text{ blanks} \rightarrow \right|$$

For the remainder of this section we fix input α and assume that n is sufficiently large so that $\hat{\Sigma}$ may be represented by $g_k(n)$ bits, that is

$$2^{g_k(n)} \geq |\hat{\Sigma}| \; .$$

We fix $G = g_k(n)$, emphasizing that 2^G is the bound on time and space. The tape cells (equivalently, string positions) are numbered from the left from 0 to $2^G - 1$, knowing that M will never move off these cells.

We conceptualize the construction of the formula Φ with the aid of certain triples

$$\langle \sigma, s, t \rangle \in \hat{\Sigma} \times 2^G \times 2^G \; ,$$

where we take $\langle \sigma, s, t \rangle$ to be an assertion that the value of the s-th symbol at step t is σ. In the obvious manner, define

$$Next: \hat{\Sigma} \times \hat{\Sigma} \times \hat{\Sigma} \rightarrow \hat{\Sigma}$$

from the transition table of $\mathcal{M}$*. An accepting computation of $\mathcal{M}$ on input x is thus described by a set of triples $\mathcal{C}$ such that

1) $\mathcal{C}$ is single valued in the first element of the triples:

$$\langle \sigma, s, t \rangle \in \mathcal{C} \text{ and } \langle \zeta, s, t \rangle \in \mathcal{C} \Rightarrow \sigma = \zeta$$

---

* For any permutation of $\langle \sigma_1, \sigma_2, \langle q_A, \sigma_3 \rangle \rangle$, $\sigma_i \in \Sigma$, $Next$ is the identity - i.e. M remains in q_A forever.

2) $\mathcal{C}$ includes triples describing the initial configuration:

$$\{\langle(a_0, q_0), 0, 0\rangle\} \cup \{\langle a_i, i, 0\rangle \mid 1 \leq i < n\} \cup \{\langle \beta, i, 0\rangle \mid n \leq i < f_k(n)\} \subseteq \mathcal{C}$$

3) $\mathcal{C}$ includes an accept state:

$$\exists \langle \sigma, s, t\rangle \in \mathcal{C} \quad \text{such that} \quad \sigma = (\theta, q_A) \quad \text{for some} \quad \theta \in \Sigma .$$

4) any triples in $\mathcal{C}$ follow from previous (in the "t"-component) triples:

$$\langle \sigma, s, t\rangle \in \mathcal{C} \;\&\; t \neq 0 \;\Rightarrow$$

$$\left[\begin{array}{l} s \neq 0 \;\&\; s \neq 2^G-1 \;\&\; \\ \exists(\langle \zeta_1, s-1, t-1\rangle, \langle \zeta_2, s, t-1\rangle, \langle \zeta_3, s+1, t-1\rangle \in \mathcal{C}) \\ \quad \left[\sigma = \mathit{Next}(\zeta_1, \zeta_2, \zeta_3)\right] \\ \vee \\ s = 0 \;\&\; \exists(\langle \zeta_2, s, t-1\rangle, \langle \zeta_3, s+1, t-1\rangle \in \mathcal{C}) \\ \quad \left[\sigma = \mathit{Next}(\beta, \zeta_2, \zeta_3)\right] \\ \vee \\ s = (2^G-1) \;\&\; \exists(\langle \zeta_1, s-1, t-1\rangle, \langle \zeta_2, s, t-1\rangle \in \mathcal{C}) \\ \quad \left[\sigma = \mathit{Next}(\zeta_1, \zeta_2, \beta)\right] \end{array}\right] ,$$

where the latter two disjuncts are special cases for the left and right ends. These conditions are obviously necessary and sufficient for $\mathcal{C}$ to describe an accepting computation. Observe that $\mathcal{C}$ as defined is unordered, but if it were ordered by the third and then second (time,

then space) component of the triples, then $\mathcal{C}$ would closely resemble the familiar sequence of instantions descriptions used to describe Turing machine computations. The existentials in the fourth clause replace the need for ordering $\mathcal{C}$.

Now it remains to show that the above predicates can be coded into WMST and hence express the existence of $\mathcal{C}$. Three sequences A, S, and T are used to encode $\mathcal{C}$, corresponding to symbol, tape position, and time. A triple $\langle\sigma, s, t\rangle \in \mathcal{C}$ is encoded in A, S, and T by parallel G-intervals in all three sequences. In particular, there will be some $t \equiv 0$ and G such that

$$A(t+G-1)\ A(t+G-2) \cdots A(t+1)\ A(t)$$

is a binary code for the symbol σ and

$$S(t+G-1)\ S(t+G-2) \cdots S(t+1)\ S(t)$$

and

$$T(t+G-1)\ T(t+G-2) \cdots T(t+1)\ T(t)$$

are the binary representations of s and t respectively.

With this representation in mind it is not difficult to construct a formula F_α such that

$$(\exists A)(\exists S)(\exists T)[F_\alpha]$$

is true iff $\mathfrak{M}$ accepts α . F_α is simply a conjuction of clauses F_1, F_2, F_3 and F_4 expressing 1-4 above. Rather than subject the reader to tedious and unenlightening coding of 1-4 at this point, we do

only F_1 as an example:

$$(\forall x)(\forall y)$$

$$\left\{ \left[\begin{array}{l} x \equiv 0 \bmod G \ \ \& \ \ y \equiv 0 \bmod G \ \ \& \ \ x \leq \max(S) \ \ \& \ \ y \leq \max(S) \\ \& \ \ (\forall i)(\forall j) \left[\begin{array}{l} \lceil x > i \geq \underset{G}{L} x \ \ \& \ \ \lceil y > j \geq \underset{G}{L} y \ \ \& \ \ i \equiv j \bmod G \\ \Rightarrow (i \in S \Leftrightarrow j \in S) \ \ \& \ \ (i \in T \Leftrightarrow j \in T) \end{array} \right] \end{array} \right] \right\}$$

$$\Rightarrow \ x = y$$

Note that this states single-valuedness in a somewhat stronger manner, requiring that there be only one code for a given combination of s and t. The length of each of these terms in certainly polynomial (and indeed linear) in n given $\mathcal{M}$ and k, and hence the reduction is polynomial bound.

From Theorem 2.2 we know that $x \equiv y \bmod g_k(n)$ can be expressed in length polynomial in n, and substituting that expression into F1 & F2 & F3 & F4 gives the required formula Φ. Since only congruence modulo n is required as a primitive, Φ is actually in the language of restricted congruence.

□

IV. MANIPULATION OF FORMULAS INTO SPECIAL FORMS

The previous sections provided a polynomial reduction of the problem of deciding acceptance by an f_k-bounded Turing machine to that of deciding the validity of a WMST(<, r≡) sentence. In the course of that reduction, however, there was no concern about the structure of the second-order formulas. In this section we refine the previous constructions in order to investigate the relationships between the structure of second-order formulas and computational complexity.

Definition A second-order formula Φ is in $\Sigma_{i,j}$-form iff

1) Φ is in prenex form, with all set quantifiers followed by all individual quantifiers, and all individual variables bound.
2) the first (outermost) quantifier of Φ is "$\exists$".
3) Φ has $i-1$ alternations of set quantifiers and $j-1$ alternations of first-order quantifiers.†

A formula Φ is in $\Sigma_{i,\omega}$-form iff Φ is in $\Sigma_{i,j}$-form for some j. $\Sigma_{i,j}$ denotes the class of all relationships expressible by formulas in $\Sigma_{i,j}$-form, and similarly

$$\Sigma_{i,\omega} = \bigcup_{j \in N} \Sigma_{i,j} .$$

The forms and classes $\Pi_{i,j}$ and $\Pi_{i,\omega}$ are defined analogously.

---

† In the general case one should also consider the leading individual quantifier, but for monadic second-order formulas this notation is well defined, since we may always find an equivalent formula with the same on fewer individual quantifier alternations where the leading individual quantifier is of different type then the last set quantifier. For example, $\ldots (\forall Z)(\forall a) \ldots M$ is convented to $\ldots(\forall Z)(\forall S_a)(\exists a) \ldots S_a(a) \Rightarrow M$.

With these definitions we may now state the major results of this section

Theorem 4.1 For any $k > 0$, $\mathcal{F}_k$ is polynomial expressible in WMST(<, r≡) by formulas in $\Sigma_{1,\omega}$-form.

Theorem 4.2 For any $k > 0$, $\mathcal{F}_k$ is polynomial expressible in WMST(') by formulas in $\Sigma_{k,\omega}$-form.

The remainder of this section will be devoted to the proofs of these theorems, which involves manipulation of second-order sentences and many intermediate representations. When finding a formula in special form, we must insure that this formula not only expresses the correct relationship, but that it is neither too long nor to difficult to arrive at.

Definition. A class of formulas $\mathcal{A}$ is polynomially expressible by a class of formulas $\mathcal{B}$ if

a) for each $\Phi \in \mathcal{A}$ there is a logically equivalent $\Psi \in \mathcal{B}$.

b) there is some polynomial p such that Φ may be transformed into Ψ in fewer than $p(|\Phi|)$ steps (and hence $|\Psi| \leq p(|\Phi|)$) by some algorithm which works with this bound for all $\Phi \in \mathcal{A}$.

The same notion holds for a class of relationships, except that the relevant bound is polynomial in the constant(s) of a relationship.

The manipulations are based on the following lemmas, which are basic from logic but which provide extremely useful tools. In the

following discussion, it will be unnecessary to distinguish between first- and second-order variables, although the main application will be techniques for moving certain set variables past individual ones.

Definition Let P be a predicate with free variables $x, y_1, y_2, \cdots, y_n$. Then $P(x, y_1, \cdots, y_n)$ is uniquely satisfiable in x for $y_1, \cdots, y_n$ if for each instance of $y_1, \cdots, y_n$ there is a unique instance of x which satisfies P. The phrase "for $y_1, \cdots, y_n$" is omitted unless it is necessary to make all free variables explicit.

Lemma 4.3 Let $P(x)$ be uniquely satisfiable in x. Then the following two expressions are equivalent, for any Q,

$$(\exists x)[P(x) \ \& \ Q(x)]$$
$$(\forall x)[P(x) \Rightarrow Q(x)] \ .$$

Lemma 4.4 Let $P(x)$ be uniquely satisfiable in x and let M be an expression containing $S = (\exists x)[P(x) \ \& \ Q(x)]$, and such that M does not contain x except in S, nor does M contain quantifiers for the other free variables of P. Then M is equivalent to

$$(\exists x)[P(x) \ \& \ \overline{M}(x)] \ \underline{\underline{\text{dfn}}} \ \overline{S}$$

where $\overline{M}(x)$ is obtained from M by replacing S by $Q(x)$.

Proof. Let x_0 be that unique x satisfying P. Clearly S iff $Q(x_0)$ and $\overline{S}$ iff $\overline{M}(x_0)$. Thus, substituting $Q(x_0)$ for S gives $\overline{M}(x_0)$, and this expression is equivalent to M.

□

Lemma 4.5 Let $P(x)$ be uniquely satisfiable, then

$$(\exists x)(\exists y)[P(x) \& P(y) \& Q(x, y)]$$

is equivalent to

$$(\exists x)[P(x) \& Q(x, x)]$$

We now return to second order theories with two lemmas of a technical nature.

Lemma 4.6 Let $\Phi = (\Delta_1\ z_1) \ldots (\Delta_r\ z_r)M(X_1, \ldots, X_p, y_1, \ldots, y_q, z_1, \ldots, z_r)$ be a formula of WMST(', 0, <, r≡), where each Δ_i is $\exists$ or $\forall$. Assume instances have been fixed for $X_1, \ldots, X_p$ and $y_1, \ldots, y_q$. Then the quantification of $z_1, \ldots, z_r$ may be restricted to integers less than or equal to $\ell+r\cdot 2\cdot c$, where $\ell = \max(\max_{i\leq p}(X_i), \max_{j\leq q}(y_j))$ and c is the largest constant in M.

Proof Let m be the unique constant with respect to which congruence is taken (obviously $m \leq c$).

If i and j are such that $z_j > z_i+2\cdot c \geq \ell$, and the instance of no other individual variable falls between z_i and z_j, then the instance of z_j may be decreased by m without changing the truth value of any of the atoms of m. This process may be repeated until there is no "gap" of more than 2c between instances of individual variables. Thus for any instantiation of $z_1, \ldots, z_r$, there is another instantiation from the range bounded by $\ell+r\cdot 2\cdot c$ such that each atomic

components of M has the same truth value for each instantiation.

□

Lemma 4.7 Let Φ be a formula of WMST(', 0, r$\equiv$) in $\Sigma_{k,\omega}$-form, for some $k > 0$. Then Φ is polynomially equivalent to Φ' of WMST('), and Φ' is also in $\Sigma_{k,\omega}$-form.

Proof The expression of equality to a constant is easy, and without loss of generality we consider congruence modulo one fixed constant m.

Assume Φ is $(\Delta_1 X_1) \ldots (\Delta_p X_p)(\Delta_{p+1} y_1) \ldots (\Delta_{P+q} y_q)M$, where each Δ_i is either $\exists$ or $\forall$.

First we translate congruence involving two variables to that only involving one. For example, $u \equiv v \bmod m$ becomes

$$\bigvee_{i<m}[x \equiv i \bmod m \ \& \ y \equiv i \bmod m]$$

Next congruences of the form $y \equiv j \bmod m$, $j \neq 0$, are changed to $y + (m - j) \equiv 0 \bmod m$, so only equivalence to 0 modulo m remains.

By lemma 4.6, we may bound the quantification of individual variables by ℓ. Indeed, we add to M clauses which accomplish this.

Now add an additional set variable W, with quantifier $\Delta_p W$ placed after $\Delta_p X_p$. Also add clauses requiring $\max(W) > \ell$ and $z \in W$ iff $z \equiv 0 \bmod m$. This latter is expressed by

$$0 \in W \;\&\; \&_{0<i<m}\left(i \in W \;\&\; (\forall z)[z + m \in W \Rightarrow z \in W]\right)$$

The construction is completed by replacing "$y \equiv 0 \mod m$" by "$y \in W$".

□

With the purely technical details completed, now we reconstruct the formulas expressing congruence and simulating computations in the required forms. Rather than directly giving the formulas and becoming bogged down in the many details, we will, as much as possible, attempt to give methods for transforming the formulas presented above into the required forms. The goals are not necessarily elimination of second-order quantifiers, but manipulation of them. The first goal is to move all set quantifiers outside of individual quantifiers. Since the expressions constructed for $X \equiv Y \mod g_k(n)$ in Lemma 2.2 required k levels of nesting of set quantifiers within individual quantifiers, the standard procedure [3] for converting these expressions to prenex form would result in many alternations of set quantifiers. We therefore must develop new methods for manipulation, in the following steps:

a) define a uniquely satisfiable form of m-ISeq,

b) substitute the modified form for m-$ISeq_0$, using Lemma 4.6

c) using Lemmas 4.3-4.5, move the modified m-ISeq outside of the scope of individual quantifiers.

d) replace the uniquely satisfiable form of m-USeq by a simpler version. This step is not required for immediate application but it will be convenient later.

a) the predicate m-USeq(ℓ, X) is defined to be true iff X is the least m-interval sequence such that $\max(X) \geq \ell$. This is true for an m-interval sequence X iff there is exactly one m-interval containing all 1's which ends at ℓ or above. Observe that the last m-interval of all 1's need not end precisely at ℓ, nor does ℓ even need to be in that m-interval. m-USeq is represented by

$$\text{m-ISeq}_0(x) \;\&\; (\forall y)\Big[y \geq \ell \;\&\; y+1 \equiv 0 \bmod m \;\&\; (\forall z)[y \geq z \geq {}_{m}y \Rightarrow z \in X] \Rightarrow y = \max(X)\Big],$$

denoted by m-USeq$_0(\ell, X)$. For a fixed value of ℓ, m-USeq is uniquely satisfiable in X; and this X is sufficient to determine the congruence of any values less than ℓ and necessary to verify any congruence involving ℓ.

Steps b) and c) are provided by the following two Lemmas, and Lemma 4.9 provides, in addition, step d).

Lemma 4.8 The relation "$y \equiv z \bmod g_k(n)$" is polynomial expressible in WMST(<, r≡) by the expression

$$(\exists X_k)(\exists X_{k-1}) \ \ldots \ (\exists X_1)$$

$$[g_{k-1}(n)\text{-}USeq_1(\max(y, z), X_k) \ \& \ g_{k-2}(n)\text{-}USeq_1(d, X_{k-1})$$

$$\& \ \ldots \ \& \ g_1(n)\text{-}USeq_1(d, X_2) \ \& \ g_0(n)\text{-}USeq_1(d, X_1)$$

$$\& \ g_k(n)\text{-}Mod_2(X_k, y, z)] \qquad *$$

where $g_i(n)\text{-}USeq_1$ and $g_{i+1}(n)\text{-}Mod_2$ are constructed inductively, for $i > 1$, replacing "$y \equiv z \bmod g_i(n)$" by "$g_i(n)\text{-}Mod_2(X_i, y, z)$" in $g_i(n)\text{-}USeq_0$ and $g_{i+1}(n)\text{-}Mod_0$ respectively, and where $d = \max(X_k) + c$, c a constant similar to $r \cdot 2 \cdot c$ of Lemma 4.6.

Observe that $X_{i-1}, \ldots, X_0$ are also free variables of $USeq_1$ and Mod_2, and that there are no other free variables, either first- or second-order.

Proof The proof is by induction on k, for which no basis is necessary since congruence modulo n is a primative.

We assume the result for some fixed k and express $y \equiv z \bmod g_{k+1}(n)$ as required. From Lemma 2.2 we know that this is represented by

$$(\exists X_{k+1})[g_k(n)\text{-}ISeq_0(X_{k+1}) \ \& \ g_{k+1}(n)\text{-}Mod_1(X_{k+1}, y, z)] \ ,$$

which in turn contains occurrences of congruence modulo $g_k(n)$. These occurrences of congruence modulo $g_k(n)$ are of course replaced by expressions of the form * using the induction assumtpion.

As was observed in the definition of $USeq$,

$\max(X_{k+1}) \geq \max(y, z)$ is necessary and sufficient to verify $y \equiv z \bmod g_{k+1}(n)$, and hence $g_k(n)\text{-}ISeq_o(X_{k+1})$ may be replaced by $g_k(n)\text{-}USeq_o(\max(y, z), X_{k+1})$.

Now observe that in all significant cases X_{k+1} must be greater than any values whose congruence is being considered, and hence we may require $\max(X_{k+1}) + c \geq \max(X_i)$, for all $i \leq k$ and some suitable c. Hence, by Lemma 4.6, all individual variables may be restricted to $\max(X_{k+1}) + c$ (possibly a different c) and every occurrence of $g_i(n)\text{-}USeq_1$, $0 \leq i \leq k\text{-}1$, the first parameter may be replaced by $\max(X_{k+1}) + c$. Each occurrence of $g_i(n)\text{-}USeq_1$ is therefore uniquely satisfiable in X_{i+1} dependent only on X_{k+1}. Using repeated applications of Lemma 4.4, all such occurrences may now be moved out, along with the existential quantifier for their associated X_{i+1}, to just inside of $(\exists X_{k+1})$. This procedure will technically require renaming of many instances of X_{i+1}, and result in many distinct occurrences of $g_i(n)\text{-}USeq_1$; but these multiple occurrences may be unified, by Lemma 4.5, into one occurrence of $g_i(n)\text{-}USeq_1$ and one of $(\exists X_{i+1})$.

With an appropriate ordering of terms, the modified formula expressing "$y \equiv z \bmod g_{k+1}(n)$" is in *-form, completing the induction step.

□

Lemma 4.9 Let Φ be any formula of WMST(<) including, in addition, congruence modulo $g_i(n)$, for any number of $i \leq k$. Assume that the set variables of Φ are $W_1, \ldots, W_q$, all of which are free. Then is equivalent to the following formula of WMST(<, r$\equiv$), where the unique modulus in n:

$$(\exists X_k)(\exists X_{k-1}) \ldots (\exists X_1)[P_k(X_k) \,\&\, \ldots \,\&\, P_0(X_0) \,\&\, \Phi'] \ ,$$

where Φ' is constructed by substituting "$g_i(n)$-$\mathrm{Mod}_2(X_i, y, z)$" for each occurrence of "$y \equiv z \bmod g_i(n)$" and all $P_i(X_i)$ are either $g_{i-1}(n)$-$\mathrm{USeq}_1(\max(\bigcup_{j\leq q} W_j) + c, X_i)$ [call this **-form] or all $g_{i-1}(n)$-$\mathrm{ISeq}_2(X_i)$ [call this ***-form].

Proof A proof similar to that of Lemma 4.8 shows that Φ is equivalent to the **-form expression.

To obtain the equivalence of the **- and ***-forms, first observe that since any set satisfying m-$\mathcal{U}$Seq satisfies m-$\mathcal{J}$Seq, the **-form implies the ***-form. The converse follows since any X_i satisfying $\max(X_i) \geq \max(y, z)$ is sufficient to determine the validity of $y \equiv z \bmod g_i(n)$, and all variables may be limited, by 4.6, to $\max(\bigcup_{j\leq q} W_j) + c$. On the other hand, if particular instances of X_i's satisfy the ***-form expression, any longer instance of X_i which still satisfies $g_{i-1}(n)$-$\mathcal{J}$Seq(X_i) will do as well. Thus any instances of X_i's satisfying the ***-form may be shortened (by the first of the previous two sentences) or lengthened (by the second) to satisfy the restrictions of the **-form.

□

Now we restate and prove:

Theorem 4.1 For any $k > 0$, $\mathcal{F}_k$ is polynomial expressible in WMST(<, r≡) by formulas in $\Sigma_{1,\omega}$-form.

Proof From Theorem 3.1, we know that $\mathcal{F}_k$ is polynomial expressible by formulas

$$(\exists A)(\exists S)(\exists T)[\Phi_\alpha] \ ,$$

where Φ has no other set variables and contains only congruence modulo $g_{k-1}(n)$, where the input α is of length n. But by Lemma 4.9, we may replace these congruences by formulas involving congruence modular, and only existential set quantifiers will be added immediately insode of those three already present. Thus the resulting formula would still be in $\Sigma_{1,\omega}$-form.

□

Having completed the case of WMST(<, r≡), we now eliminate "<" and work with WMST(', 0, r≡). By Lemma 4.7, expressing a relation in terms of ', 0, and r≡ is equivalent to expressing it in terms of ' alone.

We now define certain formulas and classes of formulas, building up to the desired expression of congruence and ultimately

of bounded computations.

$Lt(X, y)$ iff $X = \{z : z \leq y\}$, expressed by $y \in X \;\&\; y' \notin X \;\&\; (\forall z)[z' \in X \Rightarrow z \in X]$. Observe that $Lt(X, y)$ is uniquely satisfiable in X for y.

$x = y$ is expressed by $(\exists X)[Lt(X, x) \;\&\; y \in X \;\&\; y' \notin X]$, or the equivalent for interchanging x and y. We will assume that x is bound external to y, and thus specifically adopt the form given above. Since Lt is uniquely satisfiable in X, the quantifier for X may be moved out to that of x, and the same quantifier form may be used for both.

$x \leq \max(S)$ associate with every S appearing in such a clause another second-order variable $L_S = \{z : z \leq \max(S)\}$, and observe that $x \leq \max(S)$ iff $L_S(x)$. As an example, consider $(\exists S)M$, where M contains $x \leq \max(S)$. This is replaced by

$$(\exists S)(\exists L)\left(\begin{array}{l}(\forall z)[(z \in S \Rightarrow z \in L) \;\&\; (z' \in L \Rightarrow z \in L)] \;\&\\ (\exists z)[z \in S \;\&\; z' \notin S \;\&\; z \in L \;\&\; z' \notin L] \;\&\; M'\end{array}\right)$$

where M' is obtained from M by substituting $x \in L$ for $x \leq \max(S)$.

The predicate defining L_S is uniquely satisfiable. Thus L_S may be quantified with S and causes no additional complexity of set quantifiers, allowing us to think of $x \leq \max(S)$ as an entirely first-order construction.

m-Str(X) which states that X contains one string of exactly m members, and no other members. That is, for some y , $X = \{y, y+1, \ldots, y+(m-1)\}$. m-Str is defined in terms of congruence modulo m , and is expressed by

$$(\exists y)[y \in X] \;\&$$

$$(\forall y)\Big([(y = 0 \;\&\; 0 \in X) \vee (y \in X \;\&\; y-1 \notin X)] \Rightarrow$$

$$(\forall z)\Big[\begin{array}{ll} (z \notin X & \Rightarrow z' \notin X \vee y = z' \;) \;\& \\ (z \in X & \Rightarrow z' \in X \vee y \equiv z \bmod m) \;\& \\ (y \equiv z' \bmod m & \Rightarrow y = z \vee z' \notin X \;) \end{array} \Big] \Big)$$

Observe that equality as well as congruence is used in this statement. As is seen above, equality corresponds to a single set variable, which is bound at the most exterior level with $\forall y$ and may be taken to have a universal quantifier. Since the presence of $\forall y$ will require atleast one second-order quantifier in order to move the quantifiers of mod m to a prenex form, the additional second-order universal generated by equality will not add to the complexity of the resulting expression.

Now assume that mod m is expressible in $\Sigma_{i,\omega} \cap \Pi_{i,\omega}$. Then m-Str is expressible in $\Pi_{i,\omega}$.

m-Int(X) is true iff X is an m-interval. Using the above abbreviation, this is simply m-Str(X) & $\min(X) \equiv 0 \bmod m$. Expanding the abbreviation, this becomes

$$m\text{-}Str(X) \;\&\; (\forall y)[y \in X \;\&\; y\text{-}1 \notin X \Rightarrow y \equiv 0 \bmod m] \quad .$$

As with m-Str , m-Int is in $\Pi_{i,\omega}$ if mod m is in $\Sigma_{i,\omega} \cap \Pi_{i,\omega}$.

m-IntOf(I, x) iff I is an m-interval containing x . This is expressed by m-Int(I) & $x \in I$. Since we know that if mod m is $\Sigma_{i,\omega} \cap \Pi_{i,\omega}$ expressible, m-Str is $\Pi_{i,\omega}$ expressible, then m-IntOf is also $\Pi_{i,\omega}$ expressible. Note also that m-IntOf(I, x) is uniquely satisfiable in I .

Lemma 4.10 $\mathcal{F}_1$ is polynomial expressible in WMST(', 0, r≡) by formulas in $\Sigma_{1,\omega}$-form.

Proof We need only show that the formulas F_1, F_2, F_3, and F_4 of section III may be written, in this special case, so that no universal set quantifiers are required to replace < .

Congruence modulo n , the length of the input, is available as a primitive, and such relations as $x \leq \max(S)$ may be, as noted above, expressed using only existential quantifiers. Other expressions in n may be expressed, using ' , as conjunctions in disjuncts of terms. For example, for any predicate P ,

$$P(\underset{n}{\lfloor} x) \text{ iff } \bigvee_{0 \leq i < n} [P(x - i)] \quad \text{and}$$

$$(\forall z)[y+n > z \geq y \Rightarrow P(z)] \text{ iff } \bigwedge_{0 \leq i < n} [P(x + i)] \quad .$$

The number of terms generated in this way is always polynomial in n .

□

Similar reasoning may be used to show

Lemma 4.11 Congruence modulo $g_1(n)$ is polynomial expressible in $\Sigma_{1,\omega} \cap \Pi_{1,\omega}$ in WMST(', ≡) .

Lemma 4.12 $g_{k+1}(n)$-$\mathcal{M}od$ is polynomial expressible in $\Sigma_{k,\omega} \cap \Pi_{k,\omega}$ in WMST(', r≡) .

Proof The lemma is proved by induction on k . The basis of the induction, that $g_1(n)$-$\mathcal{M}od$ is expressible without any second-order quantifiers, follows in the same manner as Lemma 4.10. The definition of $g_1(n)$-$\mathcal{M}od$ may be carefully re-written from section II, replacing quantified constructions by ones involving n disjuncts or conjuncts, and using constants 0 thru n .

Now assume $g_i(n)$-$\mathcal{M}od$ is expressed by $g_i(n)$-Mod$_3$ in $\Sigma_{i-1,\omega} \cap \Pi_{i-1,\omega}$ and construct $g_{i+1}(n)$-Mod$_3$ in $\Sigma_{i,\omega} \cap \Pi_{i,\omega}$. Define $g_{i+1}(n)$-Mod$_3$ by substituting $g_i(n)$-Mod$_3$ for all occurences of congruence modulo $g_i(n)$ (including those occurrences appearing in $g_i(n)$-IntOf) in the following:

$$y \leq \max(X) \;\&\; z \leq \max(X) \;\&\; y \equiv z \bmod g_i(n) \;\&$$
$$(\exists I_y)(\exists I_z)[g_i(n)\text{-IntOf}(I_y, y) \;\&\; g_i(n)\text{-IntOf}(I_z, z)$$
$$\&\; \text{Match}(X, I_y, I_z)] \;,$$

where Match is true if the binary sequence encoded by X in the interval I_y matches that encoded in the interval I_z. Match is expressed by

$$(\forall u)(\forall v)[u \equiv v \bmod g_i(n) \;\&\; u \in I_y \;\&\; v \in I_z \Rightarrow (u \in X \Leftrightarrow v \in X)]$$

This results in a $\Sigma_{i,\omega}$-form for $g_{i+1}(n)$-Seq. To obtain a $\Pi_{i,\omega}$-form, observe that since $g_i(n)$-IntOf is uniquely satisfiable, we may replace $(\exists I_x)(\exists I_y)$ by $(\forall I_x)(\forall I_y)$ using Lemma 4.3. But this also places $g_i(n)$-IntOf as the antecedent of an implication, effectively converting the $\Pi_{i-1,\omega}$-forms into $\Sigma_{i-1,\omega}$-forms, and resulting in a $\Pi_{i,\omega}$-form for $g_{i+1}(n)$-Mod.

□

Lemma 4.13 $g_k(n)$-Seq is polynomially expressible in WMST(', r≡) by formulas in $\Pi_{k,\omega}$.

Proof We know that $g_k(n)$-Seq is expressible in terms of congruence modulo $g_k(n)$ and that, in the appropriate context (c.f. Lemma 4.8), formulas expressing $g_k(n)$-Mod, in particular $g_k(n)$-Mod_3, may be used to express the congruence relations. By the previous lemma, $g_k(n)$-Mod_3 is in $\Sigma_{k-1,\omega} \cap \Pi_{k-1,\omega}$. If we can

express $g_k(n)$-$\mathcal{J}\mathcal{S}\mathcal{e}\mathcal{q}$ by embedding $g_k(n)$-Mod_3 in only one quantifier alternation, of "$\forall\exists$" form, we will have the desired $\Pi_{k,\omega}$ expression.

The following four clauses express C1 thru C4 of section II. We use m to abbreviate $g_k(n)$.

C1: $(\forall W)[\text{m-Str}(W) \;\&\; 0 \in W \Rightarrow (\forall y)[y \in W \Rightarrow y \notin X]]$

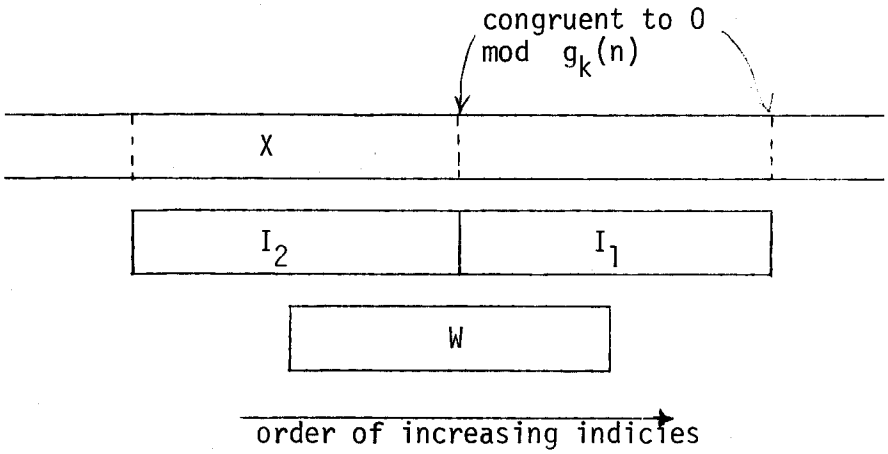
C2: $(\forall I)\left[\begin{array}{l}\text{m-Int}(I) \;\&\; \max(I) \leq \max(X) \;\Rightarrow \\ (\min(I) \in X \Leftrightarrow \max(I) + 1 \notin X\end{array}\right]$

C3: $(\forall I_1)(\forall I_2)(\forall W)$

$$\left[\begin{array}{l}\text{m-Int}(I_1) \;\&\; \text{m-Int}(I_2) \;\&\; \text{m-Str}(W) \;\& \\ \min(I_1) = \max(I_2) + 1 \;\&\; \max(I_1) \leq \max(X) \;\&\; \min(W) \in I_2 \\ \Rightarrow \left(\begin{array}{l}(\max(W) + 1 \in X \Leftrightarrow \min(W) \notin X) \Leftrightarrow \\ (\forall z)[\min(W) > z \geq \min(I_2) \;\Rightarrow\; z \notin X]\end{array}\right)\end{array}\right]$$

C4: $(\forall I)\left[\begin{array}{l}\text{m-Int}(I) \;\&\; \max(X) \in I \;\Rightarrow \\ (\max(X) = \max(I) \;\&\; (\forall y)[y \in I \Rightarrow y \in X])\end{array}\right]$

The situation in C3 may be visualized as



with W "sliding along" over I_1-I_2 measuring off an interval of length $g_k(n)$ to insure the carry propigates correctly.

The expression $g_k(n)$-$ISeq_3$ is the conjunction C1 & C2 & C3 & C4, of course expressing $g_k(n)$-$\mathcal{ISeq}$.

Since the terms involving "max", "min", etc. are abbreviations for certain first-order constructions, the only second-order variables are those universal quantifiers occurring explicitly in C1 thru C4 and any quantifiers occurring in Str and Int. From Lemma 4.8 and the observations made when these relations were defined, it follows that Str and Int are in $\Pi_{k-1,\omega}$. But since Str and Int occur as antecedents of implications, the entire expression is in $\Pi_{k,\omega}$.

□

At long last we restate our goal, replacing k in the original statement by k+1 to simplify notation in the proof.

Theorem 4.2 For any $k \geq 0$, $\mathcal{F}_{k+1}$ is polynomial expressible in WMST(') by formulas in $\Sigma_{k+1,\omega}$-form.

Proof The case for $k = 0$ is handled in Lemma 4.10, hence we consider only $k \geq 1$. Again we abbreviate $G = g_k(n)$.

By Lemma 4.7, it is sufficient to show that $\mathcal{F}_{k+1}$ is polynomial expressible in WMST(', 0, r≡) .

Pick k , $S \in \mathcal{F}_{k+1}$, and any input α , $|\alpha| = n$. In Theorem 3.1, we constructed Φ_α such that $(\exists A)(\exists S)(\exists T)[\Phi_\alpha]$ is true iff α is accepted by some machine $\mathcal{M}$ computing the characteristic function of S . By Lemma 4.9, with substitutions provided by Lemmas 4.12 and 4.13, this is equivalent to

$$\#\quad (\exists A)(\exists S)(\exists T)(\exists X_k)\ldots(\exists X_1)$$
$$[g_{k-1}(n)\text{-}ISeq_3(X_k) \;\&\; \ldots \;\&\; g_1(n)\text{-}ISeq_3(X_1) \;\&\; \hat{\Phi}_\alpha] ,$$

where $\hat{\Phi}_\alpha$ is obtained from Φ_α using $g_k(n)\text{-}Mod_3$ to express congurence modulo $g_k(n)$. From 4.13, $g_{k-1}(n)\text{-}ISeq_3$ is in $\Pi_{k-1,\omega}$, and the other ISeq expressions are in even simple classes. Thus considering only the ISeq terms the formula # is in $\Sigma_{k+1,\omega}$. It remains to show that an expression equivalent to Φ'_α may be found in $\Sigma_{k+1,\omega}$.

Of the clauses F1-F4 (see section III) which make up Φ , F1 and F2 are only universal quantifiers bounding congruence modulo $g_k(n)$, and by Lemma 4.12, these are thus in $\Pi_{k-1,\omega}$. Similarly, F3 uses existential quantifiers outside of the congruence and thus is in $\Sigma_{k-1,\omega}$.

The encoding of F4 requires greater care. As presented in section III, F4 states, ignoring special cases, "for every triple $\langle\sigma, s, t\rangle$, there exist three triples $\langle\zeta_1, s\text{-}1, t\text{-}1\rangle$, $\langle\zeta_2, s, t\text{-}1\rangle$, $\langle\zeta_3, s+1, t\text{-}1\rangle$ ...". This requires an alternation of quantifiers

which is unacceptable if we desire a $\Sigma_{k+1,\omega}$-form. To avoid this difficulty, we will add a fifth clause F5 (presented below). F5 will require that, for every $s,t \leq m$, $\langle \sigma, s, t \rangle \in \mathcal{C}$ for some σ. F1 already insures single-valuedness. Thus F4 may be revised to the purely universal form: "For every triple $\langle \sigma, s, t \rangle$, and all $\langle \zeta_1, s-1, t-1 \rangle, \langle \zeta_2, s, t-1 \rangle, \langle \zeta_3, s+1, t-1 \rangle, \ldots$".

The actual quantification is done, of course, with individual variables. For example, "for all $\langle \sigma, s, t \rangle \ldots$" is stated "$(\forall i)[i \equiv 0 \bmod \Rightarrow \ldots]$" and the binary values $S_{i+G-1}\, S_{i+G-2} \cdots S_{i+1}\, S_i$ and $T_{i+G-1}\, T_{i+G-2} \cdots T_i$ (where $S_x = 1$ if $x \in S$ and 0 otherwise, etc.) correspond to s and t.

"Next" is obviously finitely defined, dependent only on M; and the relation $s = 0$, for example, requires that a certain m-interval be all zero's. In order to express $s-1$, $s+1$, and $t-1$, we require the relation $g_k(n)\text{-Incr}(R, y, z)$, which is true iff y and $z \equiv 0 \bmod G$ and the binary value $R_{y+G-1}\, R_{y+G-2} \cdots R_{y+1}\, R_y$ is one less than the binary value $R_{z+G-1}\, R_{z+G-2} \cdots R_{z+1}\, R_z$. This relation is expressed by

$$y \equiv 0 \bmod G \;\&\; z \equiv 0 \bmod G \;\&\; (y \in R \Leftrightarrow z \notin R) \;\&\; (\forall Y)(\forall Z)$$
$$\Big[\text{G-IntOf}(Y, y) \;\&\; \text{G-IntOf}(Z, z) \Rightarrow$$
$$(\forall u)(\forall v)\Big[u \equiv v \bmod G \;\&\; u \in Y \;\&\; v \in Z \Rightarrow$$
$$((u \in Y \Leftrightarrow v \notin Z) \Leftrightarrow (\forall w)[w \in Y \;\&\; w < u \Rightarrow w \in R])\Big]\Big].$$

By previous remarks, G-IntOf is in $\Pi_{k-1,\omega}$; and since it occurs as the antecedent of an implication, it becomes, in effect, $\Sigma_{k-1,\omega}$ and causes

m-Incr to be at least $\Pi_{k,\omega}$. "$u \equiv v \bmod m$" is in $\Sigma_{k-1,\omega} \cup \Pi_{k-1,\omega}$, thru the substitution of $m\text{-Mod}_3$, and thus does not require the expression to be above $\Pi_{k,\omega}$. "$w < u$" requires a single universal set quantifier associated with $(\forall u)$, and thus m-Incr is indeed in $\Pi_{k,\omega}$. Thus F4 is in $\Pi_{k,\omega}$.

Finally, we consider F5, which will require that $\mathcal{C}$ contain some triple $\langle \sigma, s, t \rangle$ for all possible combinations of s and t. Since there are 2^G possibilities for each s and t, and since both are represented simultaneously by sequences of length, the total sequence representing all combinations will have length $G \cdot (2^G)^2 = G \cdot 2^G$. Since it is easy, by doubling all occurrences of successor, to double the length of all sets, we presume this done (changing F1-F4 into F1'-F4', also ISeq_3 into ISeq'_3, etc.) and require instead

$$\text{F5)} \qquad \max(S) = 2g_k(n)\, 2^{2g_k(n)} - 1 .$$

This, in turn, may be easily expressed if $2g_k(n)$-ISeq is expressible. But 2m-ISeq is very similar to m-ISeq, and in fact comes essentially free if the doubling convention is dropped. Of course, this is in $\Pi_{k,\omega}$, so F5 is.

Hence the required $\hat{\Phi}_\alpha$, in $\Pi_{k,\omega}$, is F1' & F2' & F3' & F4' & F5. Note that the expression # must be modified, in obvious but trivial ways, to double certain occurrences of the successor function as required by the construction of F5.

□

V. CONCLUSION

Thus far we have exhibited a correspondence between classes of the (modified) Ritchie heirarchy and sentences of WMST in certain forms. We will now briefly discuss the reverse correspondence - from formulas to the complexity of their decision algorithms.

First consider WMST(') . It has long been known that the validity of a sentence in this theory may be decided by constructing a finite automata which accepts some input iff the sentence is valid [1]. While the original algorithm was not designed for computational efficiency, it may be carefully re-done avoiding unnecessary steps exponential growth and using great care in choosing intermediate representation. This will allow us to come very close to the desired result and to show that every formula in $\Sigma_{k,\omega}$ may be decided by some procedure in $\mathcal{F}_{k+1}$. Unfortunately, within the above notation, we are unable to reduce the $k+1$ to k and thus obtain an exact one-one correspondence between complexity classes and sentence structures.

The difficulty seems to lie in a rather peculiar anomaly that no uniform definition of complexity class which corresponds to the structural heirarchy in WMST(') . We must define

for $i \leq 2$, $\hat{\mathcal{F}}_i = \mathcal{F}_i$

for $i \geq 3$, $\hat{\mathcal{F}}_i$ = the class of all sets S accepted by come (possibly non-deterministic) Turing machine in space (but not time) bounded by f_i .

Thus for $i \geq 3$, $\hat{\mathcal{F}}_i$ corresponds to Ritchie's original definition.

Claim: For any $k > 0$,

1) $\hat{\mathcal{F}}_k$ is polynomial expressible in WMST(') by formulas in $\Sigma_{k,\omega}$-form.

2) the decision procedure for formulas in WMST(') of $\Sigma_{k,\omega}$-form is in $\hat{\mathcal{F}}_k$.

The second part of this claim follows from a careful implementation of the decision procedure, while the first part follows from an argument similar to those used in sections III and IV. In particular, for $k \geq 3$, the unordered set of triple $\mathcal{C}$ is replaced by an ordered sequence - essentially the sequence of instantaneous description of the computation. This removes the time constraint for the simulation of computation. The set $\mathcal{C}$ is required for $k \leq 2$ since the formula does not allow expression of the required ordering.

The interpretation of the strange fact that time as well as space must be bounded for $k \leq 2$ may only be guessed. This fact may give support to the conjecture that an exponential space bound implies an exponential time bound as well; or in fact the conjecture could be reversed, supposing that $\hat{\mathcal{F}}_k - \mathcal{F}_k$, $k \geq 3$, is indeed empty. On the other hand, many of the sub-recursive heirarchies contain anomalies at their lowest levels, and study of abstract complexity has shown very important properties which may fail for the lower complexity classes of some measures [4, 5]. Perhaps we have simply uncovered another such anomaly.

The constraint $k > 0$ may be replaced by $k \geq 0$ if we interpret $\Sigma_{0,\omega}$ as the propositional calculus and appeal to Cook's result [2].

We thus have constructed a heirarchy of formulas of WMST(') which has the same structure as Kleene's arithmetic heirarchy, except that a "jump" from $\Sigma_{k,\omega}$ to $\Sigma_{k+1,\omega}$ corresponds to exponential growth in computation space. For any k, the class of sentences in $\Sigma_{k,\omega} \cap \Pi_{k,\omega}$ corresponds to those computations which may be performed deterministically in f_k-bounded space. This may be seen from the fact that if S is accepted deterministically in f_k space, then so is $\overline{S}$. Hence membership in S is equivalent to the validity of negations are naturally in $\Pi_{k,\omega}$. It is also possible to show this by a direct construction, constructing for a particular input a formula which says any (hence the unique) terminating computation must end in an accept state.

There is no doubt that it will be possible to find sets which are p-complete in $\Sigma_{k,\omega}$ and $\Pi_{k,\omega}$. As with the polynomial heirarchy [7], it will probably also be possible to construct still further structure similar to that of the arithmetic heirarchy.

Now we consider WMST(<). First observe that any decision procedure for WMST(') provides a decision procedure for WMST(<). This is done in the obvious manner, replacing each expression "x < z" by the expression which was noted in Section II, and which requires only successor. The algorithm for WMST(') is then applied. However, the formula expressing

"$x < z$" introduces an additional second order variable, and theorem 4.1 implies we are unable to place any elementary a priori bound on the decision time of a sentence for WMST(<) by considering the form of its set quantifiers alone. However, such a bound may be obtained by considering the entire structure.

Theorem 5.1 Let Φ be a sentence of WMST(', <, r$\equiv$) which has s set quantifiers, i individual quantifiers, and r occurrences of the relation. Then Φ is polynomially equivalent to Ψ of WMST('), where Ψ is in $\Sigma_{k,\ell}$-form, for $k = s+r+i+2$ and $\ell = 3i+r+2$. Hence the validity of Φ may be decided in space $f_k(|\Phi|)$ and time $f_{k+1}(|\Phi|)$.

Proof Each substitution for $<$ introduces exactly one additional set quantifier, so Φ may be converted into an equivalent formula in WMST(') with $s+r+1$ set quantifiers. This also introduces r additional individual quantifiers.

By standard algorithms, we may convert the resulting formula into normal form, but with no particular order on the quantifiers. At worst, the i individual quantifiers bound all $s+r$ set quantifiers, with the r individual quantifiers introduced during the substitution for $<$ inside. Next, a clause which will be used for expressing equality to 0 is added. Say the formula in prenex form is Φ', then it is converted into $(\exists Z)[(\exists x)[x \in Z] \& (\forall y)[y' \notin Z] \& \Phi']$. Now define $Ltz(X, y)$, which is similar to Lt of section IV except that

it is true for $Y = 0$ if X does not equal $\{z : z \leq x_0\}$, for some x_0. Thus $Ltz(X, y)$ is unquely satisfiable in y for fixed x. Specifically, $Ltz(X, y)$ is

$$((\forall z)[z' \in X \Rightarrow z \in X] \ \& \ y \in X \ \& \ y' \in X) \ \vee$$

$$(((\forall z)[z \notin X] \vee (\exists w)[w' \in X \ \& \ w \notin X]) \ \& \ y \in Z) .$$

Now each individual quantifier which contains set quantifiers in its scope is replaced by a set quantifier as follows. Consider the case of a particular individual quantifier $(\forall y)$ which occurs in the prefix form as

$$(\ell\text{-quant. prefix})(\forall y)(\text{r-quant. prefix})M ,$$

where M is quantifier free (and r-quanti. prefix contains at least one set quantifier to make this step necessary). This formula is equivalent to

$$(\ell\text{-quant. prefix})(\forall L_y)(\exists y)[Ltz(L_y, y) \ \& \ (\text{r.-quant. prefix})M] .$$

Since $Ltz(X, y)$ is uniquely satisfiable in y, this is equivalent, by lemma 4.4, to

$$(\ell\text{-quant. prefix})(\forall L_y)(\text{r.-quant. prefix})(\exists y)[Ltz(L_y, y) \ \& \ M] .$$

The individual quantifiers in $Ltz(L_y, y)$ may trivially be moved out of $[Ltz \ \& \ M]$, obtaining an equivalent prenex form with one fewer individual quantifiers which bound set quantifiers. This step has

introduced one additional set quantifier and two individual quantifiers. An existential individual quantifier is taken care of similarly. In all at most $i+1$ set quantifiers and $2i+2$ individual quantifiers (including Z and the associated individual quantifiers) have been introduced. Thus we have $s+r+i+2$ set quantifiers and $i+r+2i+2$ individual quantifiers, and the number of alternations is of course bounded by the number of quantifiers. By construction $(\exists Z)$ is the leading quantifier.

□

The decision procedure for WMST(<) may be implimented in a uniform way by a Turing machine which applies Theorem 5.1 to bound the space it uses for its computation. This provides a normal form for WMST(<) .

Lemma 5.2 Let Φ be any formula of WMST(P) , for some collection of primitives P , such that Φ is in $\Sigma_{k,\omega}$ or $\Pi_{k,\omega}$ form. Then Φ is polynomially equivalent to a formula Φ' of WMST(P, ') , where Φ' is of the same form as Φ but has exactly k set quantifiers.

Proof: This lemma states that any number of adjacent set quantifiers of the same sign may be collapsed into a single quantifier of the same sign. Rather than a formal proof, we will indicate how, for

example, $(\exists V)(\exists W)(\exists X)$ may be collapsed into $(\exists Y)$ with an encoding such that positions $8i$ thru $(8i + 7)$ of Y (interpreted as a binary sequence) correspond to the sequence $110V_i0W_i0X_i$. This expands individual variables by a factor of 8, and they are synchronized by the regular occurrence of 110 in Y only at positions $8i$, $8i+1$, $8i+2$. The clause "$z \in V$" of the original formula would translate to

$$"z \in V \ \& \ z+1 \in V \ \& \ z+2 \notin V \ \& \ z+3 \in V" \ ,$$

the first three conjuncts providing the synchronization. This construction may be applied to all quantifier alternations in parallel, and can at worst increase the formula length by less than a square.

□

Theorem 5.3 Let Φ be any sentence of WMST(', =, <, r≡). Then Φ is polynomially equivalent to a sentence Φ' of WMST(<), where Φ' is $(\exists Y)\Psi(Y)$, and Ψ has no set variables other than Y.

Proof: Say Φ is as in Theorem 5.1 with s set quantifiers, r occurrences of $<$, and i individual quantifiers. Then the validity of Φ is accepted in space and time $f_{k+1}(|\Phi|)$ by the machine $\mathcal{M}$ deciding WMST(<). Since we can bound the computation time of $\mathcal{M}$ on input $\mathcal{M}$, Φ halts on Φ iff a formula of WMST(<) in

$\Sigma_{1,\omega}$-form is valid, by Theorem 4.1. But the previous lemma states there is an equivalent expression with exactly one set quantifier.

□

BIBLIOGRAPHY

1. Büchi, J. R. Weak second-order arithmetic and finite automata, Zeit Math. logik Grundl. Math. 6 (1960), 66-92.

2. Cook, S. A. Complexity of theorem proving procedures, Proc. Third Ann. ACM Symp. Theory of Computing (May 1971), 151-158.

3. Elgot, C. C. Decision problems of finite automata design and related arithmetics, Trans. AMS, 99 (1961), 21-51.

4. Landweber, L. H. and Robertson, E. L. Recursive properties of abstract complexity classes, JACM 19, 2 (April 1972), 269-308.

5. Lewis, F. D. The enumerability and invariance of complexity classes, JCSS 5, 3 (June 1971), 286-303.

6. Meyer, A. R. Weak monadic second order theory of successor is not elementary recursive, Preliminary Report, MIT (1972).

7. Meyer, A. R. and Stockmeyer, L. J. The equivalence problem for regular expressions with squaring requires exponential space, Proc. 13th Ann. IEEE Symp. on Switching and Automata Theory (Oct. 1972), 125-129.

8. Ritchie, R. W. Classes of predictably computable functions, Trans. AMS, 106 (1963), 139-173.