INFORMATION AND CONTROL 57, 180–204 (1983)

Complexity of the First-Order Theory of Almost All Finite Structures

ETIENNE GRANDJEAN

Département de Mathématiques, Université Lyon 1,
43 Bd du 11 novembre 1918, 69622 Villeurbanne Cedex, France

A first-order sentence of a relational type $\mathscr{S}$ is *true almost everywhere* if the proportion of its models among the structures of type $\mathscr{S}$ and cardinality m tends to 1 when m tends to ∞. It is shown that $\mathrm{Th}(\mathscr{S})$, the set of sentences (of type $\mathscr{S}$) true almost everywhere, is *complete* in PSPACE. Further, various upper and lower bounds of the complexity of this theory are obtained. For example, if the arity of the relation symbols of $\mathscr{S}$ is $d \geqslant 2$ and if $\mathrm{Pr\ Th}(\mathscr{S})$ is the set of prenex sentences of $\mathrm{Th}(\mathscr{S})$, then

$$\mathrm{Pr\ Th}(\mathscr{S}) \in \mathrm{DSPACE}((n/\log n)^d)$$

and

$$\mathrm{Pr\ Th}(\mathscr{S}) \notin \mathrm{NTIME}(o(n/\log n)^d).$$

If R is a binary relation symbol and $\mathscr{S} = \{R\}$, ($\mathrm{Th}(\mathscr{S})$ is the theory of almost all graphs), then

$$\mathrm{Pr\ Th}(\mathscr{S}) \notin \mathrm{NSPACE}(o(n/\log n)).$$

These results are optimal modulo open problems in complexity such as $\mathrm{NTIME}(T) \subsetneqq ?\ \mathrm{DSPACE}(T)$ and $\mathrm{NSPACE}(S) = ?\ \mathrm{DSPACE}(S^2)$.

1. INTRODUCTION

Let $\mathscr{S}$ be a relational type and φ be a first-order sentence of type $\mathscr{S}$. Let $p_m(\varphi)$ denote the proportion of the models of φ among the structures of type $\mathscr{S}$ and of domain $\{1, 2, \ldots, m\}$. We say that φ is *true almost everywhere* (*true a.e.*) if $\lim_{m \to \infty} p_m(\varphi) = 1$.

Fagin (1976) proved that the theory, called $\mathrm{Th}(\mathscr{S})$, of the first-order sentences (of type $\mathscr{S}$) true a.e. coincides with the theory the axioms of which say roughly: "*any finite substructure* (*possibly empty*) *has any possible finite extension.*" This rather natural theory was previously discovered and studied by S. Jaskowski, A. Ehrenfeucht, and C. Ryll-Nardzewski (unpublished results); they proved that this theory has exactly one infinite denumerable model and therefore is *consistent*, *complete*, and *decidable*. (In contrast, in

0019-9958/83 $3.00

case $\mathscr{S}$ contains a non-unary relation symbol, there is no algorithm for deciding if a monadic second-order sentence of type $\mathscr{S}$ is true a.e. Blass and Harary (1979) noted that the existence of such a sentence neither true a.e. nor false a.e. is unknown.)

Gaifman (1964) introduced Th($\mathscr{S}$) in the context of a certain infinite "probability model" and gave a procedure to decide this theory by *quantifier elimination* (see also Blass and Harary, 1979). However, the quantifier elimination is not an efficient procedure.

Graph theorists are interested in graph properties true a.e. Unfortunately Th($\mathscr{S}$) is not a very interesting theory from this point of view because Blass and Harary (1979) proved that important graph properties true a.e. such as rigidity or hamiltonicity are not deducible from first-order properties true a.e. However, we are convinced that Th($\mathscr{S}$) deserves to be studied because it is a natural logical theory.

In this paper, we show that Th($\mathscr{S}$) is $\leqslant_{\log}$-*complete* in PSPACE and then is one of the simplest logical theories. (*Quantified Boolean Formulas* and the first-order theory of *equality* (Stockmeyer, 1974; Stockmeyer and Meyer, 1973) are the best-known examples of logical theories complete in PSPACE.) Moreover we give many upper and lower bounds of the complexity of Th($\mathscr{S}$), the main bounds of which are described in Table 1.

In Section 3 we exhibit a decision algorithm of Th($\mathscr{S}$). The principle of our method is similar to that used by Ferrante and Geiser (1977) in an efficient procedure for the theory of rational order, that is: "*only order*

TABLE 1

Theory arity	Upper bounds	Lower bounds
Pr Th($\mathscr{S}$), $d \geqslant 2$	$\in$ TIME, ALT($(n/\log n)^d$, $n/\log n$) $\subseteq$ DSPACE($(n/\log n)^d$)	$\notin$ NTISP($2^{\varepsilon n/\log n}$, $o(n/\log n)^{d-1}$) for a constant ε $\notin$ NTISP(n^k, $o((n/\log n)^d/\log n)$) for any k $\notin$ NTIME($o(n/\log n)^d$)
Th($\mathscr{S}$), $d \geqslant 2$	$\in$ DSPACE($(n/\log n)^d \log n$) $\in$ DTIME($c^{(n/\log n)^d}$) for a constant c	$\notin$ NTISP($2^{\varepsilon n}$, $o(n/\log n)^{d-1}$) for a constant ε $\notin$ NTIME($o(n/\log n)^d$)
Pr Th($\mathscr{S}$), $d = 2$	$\in$ TIME, ALT($(n/\log n)^2$, $n/\log n$) $\subseteq$ DSPACE($(n/\log n)^2$)	$\notin$ NSPACE($o(n/\log n)$)
Th($\mathscr{S}$), $d = 2$	$\in$ DSPACE($n^2/\log n$)	$\notin$ NSPACE($o(n/(\log n \log\log n)^{1/2})$)

Note. TIME, ALT($T(n), A(n)$) denotes the class of languages accepted by alternating Turing machines in time $T(n)$ with $A(n)$ alternations.

counts" (in the theory of rational order). Similarly, in any model of Th($\mathscr{S}$), a finite set of elements is completely described by the "*internal relations*" in this set. This gives a rather natural game-theoretic algorithm for the decision of Th($\mathscr{S}$), which is formalized by means of the *alternation*, a suitable tool with which to measure the complexity of logical theories (see Berman, 1980b).

In Section 4, we prove various *lower bounds* of the complexity of Th($\mathscr{S}$) by using the now-classical method of *arithmetization* of Turing machines of bounded complexity (see Stockmeyer, 1976, for example). More precisely, we show that some complexity classes that involve time and space together can be reduced to the problem Th($\mathscr{S}$). Each reduction we use is computable on a Turing machine working both in space $\log n$ and in time n^k, for a fixed k, and consequently is *efficient* for time and space together. In addition, using hierarchy results for time–space classes (Bruss and Meyer, 1980), we obtain *time–space* lower bounds for Th($\mathscr{S}$). Our best-time lower bound is a consequence of a very sharp result of Lynch (1982) about *spectra* of first-order sentences.

Many of our upper and lower bounds are *optimal* modulo open problems in complexity such as NSPACE(S) = ? DSPACE(S^2).

Our ideas and methods are inspired of those of Ferrante and Geiser (1977), Stockmeyer (1976), Bruss and Meyer (1980), Berman (1980a,b), Immerman (1982), and Lynch (1982).

2. Definitions and Notation

2.1. *Definitions in Logic and Model Theory*

We use the usual notation and definitions in first-order logic and model theory (see Chang and Keisler, 1973, Chap. 1, for example).

(*Individual*) *variables* are called x, y, x', x'', x_i, y_i,..., where i is any nonnegative integer in binary notation. Sometimes for convenience we use doubly indexed variables $x_i^j, x_i^\sigma, \ldots$. Moreover we shall use *metavariables* $v, v', v_i, \ldots$ to denote variables.

Henceforth we shall write "integer" for "strictly positive integer."

Define a (finite *relational*) *type* to be a finite set of relation symbols $\mathscr{S} = \{R, \ldots, R'\}$ where each symbol is given an integer called its *arity*. The *arity* of $\mathscr{S}$ is the greatest arity of its relation symbols. In the following we assume that we have fixed such a type $\mathscr{S}$.

The *first-order language* of type $\mathscr{S}$ we use has the following symbols: the elements of $\mathscr{S}$, parentheses (,), brackets [,], variables, the equality symbol =, the logical connectives $\vee$ (or), $\wedge$ (and), $\neg$ (not), and the existential and universal quantifiers $\exists$ and $\forall$. The *atomic formulas* of type $\mathscr{S}$ are expressions of the form $v = v'$ or $R(v_1 v_2 \cdots v_m)$, where $R \in \mathscr{S}$ and m is

the arity of R. The *(first-order) formulas* of type $\mathscr{S}$ are then built up out of the atomic formulas in the usual way using $\vee$, $\wedge$, $\neg$, $\exists$, and $\forall$. Sometimes for convenience we will suppress $\wedge$ and $\forall$ or we will add the connective $\Rightarrow$ (imply). It does not matter because these symbols can be easily defined from $\vee$, $\neg$, and $\exists$.

We suppose that distinct metavariables denote distinct variables unless the opposite is specified. Let $\bar{v}_m$ denote the m-tuple $v_1, v_2, \ldots, v_m$ and similarly for $\bar{x}_m, \bar{y}_m, \ldots$. For a quantifier Q, let $Q\bar{v}_m$ denote the string $Qv_1 \cdots Qv_m$. A formula which is a part of another formula φ is called a *subformula* of φ. A variable v is *free* in φ if there is a *free occurrence* of v in φ; i.e., this occurrence does not belong to any subformula $\exists v\psi$ or $\forall v\psi$. We write $\varphi(\bar{v}_m)$ to mean that the free variables of φ are exactly the m (distinct) variables $v_1, \ldots, v_m$. Let $\varphi[v' \mid v]$ denote the formula φ in which each free occurrence of the variable v is replaced by v'. A *sentence* is a formula without free variables. A *quantifier-free* formula is a formula without quantifiers. A formula φ is said to be *prenex* (or in *prenex form*) if it is of the form $Q_1 v_1 \cdots Q_m v_m \psi$, where the Q_i are quantifiers and ψ is a quantifier-free formula. $Q_1 v_1 \cdots Q_m v_m$ is called the *prefix* of φ.

We define an *individual description* of a variable v with respect to a tuple $\bar{v}_m$ to be a formula of type $\mathscr{S}$, called $D(\bar{v}_m, v)$ (or simply D), which is a conjunction of the form

$$\bigwedge_{1 \leqslant i \leqslant m} v \neq v_i \wedge \bigwedge_{1 \leqslant i \leqslant p} \alpha_i'$$

where each α_i' is α_i or $\neg\alpha_i$ and the sequence $\alpha_1, \alpha_2, \ldots, \alpha_p$ is a list (without repetitions) of all the atomic formulas such that

(i) each α_i contains a relation symbol of $\mathscr{S}$;

(ii) v occurs in α_i;

(iii) all the other variables of α_i are among $v_1, \ldots, v_m$.

For example, if R is a binary relation symbol and if $\mathscr{S} = \{R\}$, then

$$\left(\bigwedge_{i=1,2} x_3 \neq x_i \wedge Rx_1x_3 \wedge \neg Rx_3x_1 \wedge Rx_2x_3 \wedge Rx_3x_2 \wedge \neg Rx_3x_3 \right)$$

is an individual description of x_3 with respect to x_1, x_2. Similarly a *complete description*, $\Delta(\bar{v}_m)$ (or simply Δ), of a tuple of variables $\bar{v}_m$ is a conjunction of the form

$$\bigwedge_{1 \leqslant i < j \leqslant m} v_i \neq v_j \wedge \bigwedge_{1 \leqslant i \leqslant q} \alpha_i'$$

where each α_i' is α_i or $\neg\alpha_i$ and $\alpha_1, \alpha_2, \ldots, \alpha_q$ is a list (without repetitions) of the atomic formulas such that:

(i) each α_i contains a relation symbol of $\mathscr{S}$;

(ii) all the variables of α_i are among $v_1,\ldots, v_m$.

Define a *complete description* of a formula $\varphi(\bar{v}_m)$ of type $\mathscr{S}$ to be a complete description of the tuple $\bar{v}_m$. The only complete description of a sentence is the empty conjunction denoted $\varnothing$.

A *structure* $\mathscr{M} = \langle D, R,\ldots, R'\rangle$ of type $\mathscr{S}$ consists of a nonempty set D called the *domain* of $\mathscr{M}$ (also written $\mathscr{M}$) and for each p-ary relation symbol R of $\mathscr{S}$ an *interpretation*, that is, a p-ary relation R on the domain D. In our notation we do not distinguish between a relation symbol and its interpretation. A *substructure* $\mathscr{N}$ of the structure $\mathscr{M}$ is a structure of type $\mathscr{S}$ such that the domain D' of $\mathscr{N}$ is a subset of D and the relations of $\mathscr{N}$ are the restrictions of the relations of $\mathscr{M}$ to the subset D'. Let $\bar{a}_m$ denote an m-tuple of elements $a_1,\ldots, a_m$ of a structure $\mathscr{M}$.

Let $\varphi(\bar{v}_m)$ be a formula of type $\mathscr{S}$ and let $\mathscr{M}$ be a structure of type $\mathscr{S}$ and $\bar{a}_m$ be a tuple of elements of $\mathscr{M}$. Then we write $\langle \mathscr{M}, \bar{a}_m\rangle \models \varphi(\bar{v}_m)$ to mean that $\varphi(\bar{v}_m)$ becomes a true assertion in $\mathscr{M}$ when

(i) $\bar{v}_m$ is replaced by $\bar{a}_m$;

(ii) each logical symbol is given its usual meaning;

(iii) each relation symbol is given its interpretation in $\mathscr{M}$.

If moreover φ is a sentence, then we say that $\mathscr{M}$ is a *model* of φ and we write $\mathscr{M} \models \varphi$. If Φ and $\varphi(\bar{v}_m)$ are respectively a set of sentences of type $\mathscr{S}$ and a formula of type $\mathscr{S}$, then we write $\Phi \models \varphi$ to mean that every model of all the sentences of Φ is a model of the sentence $\forall \bar{v}_m\, \varphi(\bar{v}_m)$ (the "universal closure" of φ).

Let φ be a sentence of type $\mathscr{S}$ and let $\mathscr{S}_m$ denote the set of structures of type $\mathscr{S}$ on the domain $\{1, 2,\ldots, m\}$. Let us define the proportion:

$$p_m(\varphi) = \operatorname{card}\{\mathscr{M} \mid \mathscr{M} \in \mathscr{S}_m \text{ and } \mathscr{M} \models \varphi\}/\operatorname{card} \mathscr{S}_m.$$

We say that the sentence φ is *true* (*false*) *almost everywhere* or briefly *true* (*false*) *a.e.* if $\lim_{m\to\infty} p_m(\varphi) = 1$ ($= 0$ respectively). Let $\operatorname{Th}(\mathscr{S})$ denote the set of the sentences true a.e. More generally, we define similarly a property (even nonexpressible by a first-order sentence) *true a.e.*

We will be interested occasionally in two other theories. Let $\mathbb{Q}$ be the set of rational numbers and let $\langle \mathbb{Q}, <\rangle$ denote the structure of domain $\mathbb{Q}$ with the natural order relation on $\mathbb{Q}$ as the only relation. Let $(\mathbb{Q}, <)$ denote the set of sentences φ such that $\langle \mathbb{Q}, <\rangle \models \varphi$.

Quantified boolean formulas are built up out of propositional variables, with the connectives $\vee$, $\wedge$, $\neg$, parentheses, and the quantifiers $\exists$ and $\forall$. Let QBF denote the set of quantified boolean sentences (i.e., formulas without

free propositional variables) which are true in the trivial boolean algebra of domain $\{0, 1\}$ (see Hopcroft and Ullman, 1979, p. 344).

For a theory, i.e., a set of sentences, $\mathscr{E}$, let Pr $\mathscr{E}$ denote the set of prenex sentences of $\mathscr{E}$.

2.2. *Definitions in Computational Complexity*

A *word* w is a finite string of symbols over a finite alphabet. Let $|w|$ denote the length of the word w. Variables and formulas can be regarded as special words. For example $|x_{101}| = 4$ and if φ is the formula $\neg(\neg Rx_1y_{10} \wedge x_{10} = y_{11})$ then $|\varphi| = 18$.

We use the usual notation and definitions in computational complexity (see Hopcroft and Ullman, 1979; Chandra *et al.*, 1981; Stockmeyer, 1976). Our models of computation are *multitape Turing machines*; more exactly each machine has one read-only tape for input, several read–write tapes (the *worktapes*), and in case it computes a function, one write-only tape for output. For brevity a *deterministic* (*nondeterministic*, *alternating*) Turing machine is called a DTM (NTM, ATM, respectively). We assume that the reader is familiar with DTMs and NTMs.

Let us describe how an ATM *accepts* a language. We suppose that the set of states of an ATM is partitioned, on the one hand, into *branching* states (subdivided into *existential* states and *universal* states) and, on the other hand, into *deterministic* (possibly *accepting* or *rejecting*) states and *negating* states. Thus any *instantaneous description* (ID) of an ATM is either branching, i.e., existential or universal, or deterministic or negating according to its state.

For an ATM M let us define an *accepting* (*rejecting*) ID, by using the *successor* relation (an ID δ' is a successor of an ID δ if δ' follows from δ in one step of M) and the nature of IDs as in Chandra *et al.* (1981):

(i) if δ has an accepting (rejecting) state, then δ is accepting (rejecting);

(ii) if δ is universal (existential) and all the successors of δ are accepting (rejecting), then δ is accepting (rejecting);

(iii) if δ is existential (universal) or deterministic and at least one of the successors of δ is accepting (rejecting), then δ is accepting (rejecting);

(iv) if δ is negating and its successor is accepting (rejecting), then δ is rejecting (accepting);

(v) if no condition among (i)–(iv) holds for δ, then δ is neither accepting nor rejecting.

An input w is *accepted* (*rejected*) by M if the initial ID on input w is accepting (rejecting).

In case M has no negating state, it is equivalent but more convenient to

define acceptance (rejection) by using *accepting* (*rejecting*) *trees* (see Berman, 1980b). An accepting tree of the ATM M is a finite tree the nodes of which are labeled with IDs of M so that:

(i) each terminal node is labeled with an ID of accepting state;

(ii) each internal node labeled with a universal ID δ has a child labeled δ' for each successor ID δ';

(iii) each internal node labeled with an existential or deterministic ID δ has exactly one child labeled δ', where δ' is a successor ID of δ.

An input w is *accepted* by the ATM M if there is an accepting tree of M the root of which is labeled with the initial ID on input w. The *rejection* of an input is defined by duality ("accepting," "existential" are replaced by "rejecting," "universal," respectively, and vice versa). The *language accepted* by an ATM is the set of the inputs accepted by the ATM. We say that an ATM M *accepts* a language A *in time* $T(n)$ (*in space* $S(n)$) if M accepts A and for every w in A of length n, there is an accepting tree of M on input w of depth at most $T(n)$ (where each ID labeling a node uses at most $S(n)$ worktape cells, respectively).

The *complexity class* DTIME($T(n)$) (NTIME($T(n)$), ATIME($T(n)$)) is the class of the languages A with the property: there is a DTM (NTM, ATM) which accepts A in time $T(n)$. The space complexity classes DSPACE($S(n)$), NSPACE($S(n)$), and ASPACE($S(n)$) are defined similarly. TIME, ALT($T(n), A(n)$) will denote the class of languages accepted by alternating Turing machines (without negating state) for which any path of an accepting tree has length $O(T(n))$ and $O(A(n))$ alternations between existential IDs and universal IDs: that is, the class STA($*, T(n), A(n)$) of Berman (1980b). The logarithm of n in base 2 will be denoted by $\log n$. Let J^+ and K^+ denote the sets of nonempty words over the finite alphabets J and K. We say that f is a *reduction* from the language $A \subseteq J^+$ to the language $B \subseteq K^+$ if f is a mapping from J^+ to K^+ such that $w \in A$ iff $f(w) \in B$, for each $w \in J^+$. If moreover there is a DTM which computes f in space $\log(|w|)$ for each $w \in J^+$ and if f is *linearly bounded* (i.e., $|f(w)| \leqslant c\,|w|$ for any $w \in J^+$ and a constant c), then we write $A \leqslant_{\log\text{-lin}} B$ (see Stockmeyer and Meyer, 1973).

3. Upper Bounds of Complexity

For each individual description $D(\bar{x}_{m-1}, x_m)$, let $\theta(D)$ denote the following sentence:

$$\forall \bar{x}_{m-1} \left(\left(\bigwedge_{1 \leqslant i < j < m} x_i \neq x_j \right) \Rightarrow \exists x_m D(\bar{x}_{m-1}, x_m) \right).$$

Let us define the sets

$$\Theta_m = \{\theta(D) \mid D = D(\bar{x}_{m-1}, x_m) \text{ is an individual description of } x_m \text{ with respect to } \bar{x}_{m-1}\}$$

and

$$\Theta = \bigcup_m \Theta_m.$$

LEMMA 3.1 (Fagin, 1976). *Each sentence of Θ is true a.e. In particular for any m, the conjunction of the sentences of Θ_m is true a.e.*

Intuitively it seems clear that in any model of Θ, a complete description Δ of a formula φ determines a truth value for φ which is independent of the model of Θ. This fact (a consequence of Lemmas 3.2 and 3.3 below) helps to clarify the following algorithm, denoted TRUTH(Δ, φ), which is an alternating algorithm (i.e., a program for an ATM).

The Algorithm TRUTH(Δ, φ)

Input: (Δ, φ), where Δ is a complete description of the formula φ.

According as φ is atomic or as its main logical symbol is $\vee$, $\exists$, or $\neg$, go to one of the following subroutines.

($\vee$) $\varphi \equiv \psi_1 \vee \psi_2$. Let Δ_i be the complete description of ψ_i $(i = 1, 2)$, extracted from Δ. Choose $i = 1$ or 2 existentially (i.e., by an existential state) and apply TRUTH(Δ_i, ψ_i).

($\exists$) $\varphi(\bar{v}_k) \equiv \exists v \psi$. Two subcases.

—If v is not a free variable in ψ, apply TRUTH(Δ, ψ).

—If v is a free variable in ψ, choose existentially to replace or not to replace v by a variable among $v_1, v_2, \ldots, v_k$. If we choose "yes," choose existentially an $i \in \{1, 2, \ldots, k\}$ (intuitively v "equals" v_i) and apply TRUTH$(\Delta(\bar{v}_k), \psi[v_i \mid v](\bar{v}_k))$. Otherwise (i.e., v is not replaced), choose existentially an individual description $D(\bar{v}_k, v)$ of v with respect to $\bar{v}_k$ and apply TRUTH$(\Delta(\bar{v}_k) \wedge D(\bar{v}_k, v), \psi(\bar{v}_k, v))$.

($\neg$) $\varphi \equiv \neg\psi$. Apply negatively (i.e., by a negating state) TRUTH(Δ, ψ).

(At) φ is an atomic formula. Three subcases:

—If $\varphi \equiv v = v$, accept.

—If $\varphi \equiv v = v'$, reject (recall: v and v' denote distinct variables).

—Otherwise $\varphi \equiv R\bar{X}_d$, where $R \in \mathscr{S}$ and $\bar{X}_d$ is a d-tuple of distinct or not distinct variables $X_1, \ldots, X_d$; if $R\bar{X}_d$ is a conjunct of the conjunction Δ, accept; else ($\neg R\bar{X}_d$ is a conjunct of Δ), reject.

This ends the algorithm TRUTH.

If we permit the connective $\wedge$ and the quantifier $\forall$ in our formulas, then we only have to add two corresponding instructions $(\wedge)$ and $(\forall)$ in the algorithm TRUTH. These instructions are dual of instructions $(\vee)$ and $(\exists)$, respectively: existential states are replaced by universal states.

If this algorithm accepts (rejects) the input (Δ, φ), we write $\mathrm{TRUTH}(\Delta, \varphi) = 1$ $(= 0$, respectively). We clearly see that for every input, each computation path of this alternating algorithm always stops with acceptance or rejection. As a direct consequence we obtain the following lemma.

LEMMA 3.2. *For any formula φ and any complete description Δ of φ we have* $\mathrm{TRUTH}(\Delta, \varphi) = 1$ *or* 0.

LEMMA 3.3. *For any formula φ with exactly m distinct variables and any complete description Δ of φ, we have*:

(i) $\mathrm{TRUTH}(\Delta, \varphi) = 1$ *implies* Θ_m *(resp.* $\Theta) \models \Delta \Rightarrow \varphi$;

(ii) $\mathrm{TRUTH}(\Delta, \varphi) = 0$ *implies* Θ_m *(resp.* $\Theta) \models \Delta \Rightarrow \neg\varphi$.

Proof of Lemma 3.3 (by induction on the complexity of φ). For the sake of simplicity, we give the proofs of (i) and (ii) for Θ instead of Θ_m.

The cases (At), $(\vee)$, and $(\neg)$ are obvious. Let us prove (i) and (ii) in the case $\varphi \equiv \exists v\psi$. Now suppose v is free in ψ (the subcase when v is not free is obvious). In order to prove (i), suppose $\mathrm{TRUTH}(\Delta(\bar{v}_k), \exists v\psi(\bar{v}_k, v)) = 1$. Then we have either $\mathrm{TRUTH}(\Delta(\bar{v}_k), \psi[v_i \mid v](\bar{v}_k)) = 1$ for some $i \in \{1,..., k\}$ or $\mathrm{TRUTH}(\Delta(\bar{v}_k) \wedge D(\bar{v}_k, v), \psi(\bar{v}_k, v)) = 1$ for some individual description $D(\bar{v}_k, v)$. In the first situation we have

$$\Theta \models \Delta(\bar{v}_k) \Rightarrow \psi[v_i \mid v](\bar{v}_k)$$

and then

$$\Theta \models \Delta(\bar{v}_k) \Rightarrow \exists v\psi(\bar{v}_k, v).$$

In the second situation we have

$$\Theta \models (\Delta(\bar{v}_k) \wedge D(\bar{v}_k, v)) \Rightarrow \psi(\bar{v}_k, v)$$

and from the fact that $\theta(D) \in \Theta$ we deduce

$$\Theta \models \Delta(\bar{v}_k) \Rightarrow \exists v\psi(\bar{v}_k, v).$$

In order to prove (ii), suppose

$$\mathrm{TRUTH}(\Delta(\bar{v}_k), \exists v\psi(\bar{v}_k, v)) = 0.$$

Then we have both $\mathrm{TRUTH}(\Delta(\bar{v}_k), \psi[v_i \mid v](\bar{v}_k)) = 0$ for each $i \in \{1,\ldots,k\}$ and $\mathrm{TRUTH}(\Delta(\bar{v}_k) \wedge D(\bar{v}_k, v), \psi(\bar{v}_k, v)) = 0$ for any individual description $D(\bar{v}_k, v)$. Hence we deduce

$$\Theta \models \Delta(\bar{v}_k) \Rightarrow \forall v \neg\psi(\bar{v}_k, v).$$

By analyzing carefully the proof above, we observe that each assertion of the form $\Theta \models \varphi$ can be replaced by $\Theta_m \models \varphi$, where m is the number of (distinct) variables of the formula φ. This completes the proof. ∎

The following proposition is a direct consequence of Lemmas 3.1, 3.2, and 3.3.

PROPOSITION 3.4. *Let φ be a sentence with exactly m distinct variables. Then* (i), (ii), *and* (iii) *are equivalent*:

(i) $\mathrm{TRUTH}(\varnothing, \varphi) = 1$;

(ii) $\Theta_m \models \varphi$;

(iii) φ *is true a.e. (that is,* $\varphi \in \mathrm{Th}(\mathscr{S})$).

COROLLARY 3.5. *If a sentence φ has exactly m distinct variables, then $\Theta_m \models \varphi$ or $\Theta_m \models \neg\varphi$* (Immerman, 1982). *Every sentence is true a.e. or false a.e.* (Fagin, 1976).

COROLLARY 3.6. *The* ATM *of program* $\mathrm{TRUTH}(\varnothing, \varphi)$ *accepts the set* $\mathrm{Th}(\mathscr{S})$.

In the following we will give upper bounds on complexity of $\mathrm{Th}(\mathscr{S})$ for alternating programs; then we will deduce deterministic upper bounds on $\mathrm{Th}(\mathscr{S})$ by using some simulation results of ATMs by DTMs.

DEFINITION. An ATM M (without negating states) *accepts* a language A *in space $S(n)$ with branching number $B(n)$* if M accepts A and for any $w \in A$ of length n, there is an accepting tree of M on input w, with all the IDs using at most $S(n)$ worktape cells, so that there are at most $B(n)$ branching IDs along any path of the accepting tree.

The following lemma, which generalizes slightly the well-known result $\mathrm{ATIME}(S) \subseteq \mathrm{DSPACE}(S)$ (Chandra *et al.*, 1981, Theorem 3.2), explains our interest in the branching number measure.

LEMMA 3.7. *If a set A is accepted by an* ATM M *in space $S(n)$ with branching number $B(n)$, then A is accepted by a* DTM M' *in space* $\max(S(n), B(n), \log n)$.

Proof. Clearly it is sufficient to prove the lemma in case $S(n) = B(n) \geqslant \log n$. The idea of the simulation is that of Chandra *et al.* (1981, Theorem 3.2). Therefore we emphasize the differences. For each $w \in A$ of length n there is an accepting tree of M using space $\leqslant S(n)$, time $\leqslant c^{S(n)}$ ($c^{S(n)}$ is an upper bound on the number of IDs of M of length $S(n)$) and branching number $\leqslant S(n)$. Each path of the accepting tree can be encoded by a string of $S(n)$ symbols over the alphabet $\{1, 2, ..., b\}$, where b is the maximal outbranching of any ID of M.

First we suppose that the function $S(n)$ is fully space constructible (i.e., there is a DTM which produces $S(n)$ as output using space $S(n)$ for each input of length n).

Now let us describe the simulation. For any input w of length n the DTM M' first constructs $S(n)$, secondly checks one by one, for example from left to right, the paths of the computation tree of M on input w and at the same time uses the output $S(n)$ as a counter for exceeding neither the space $S(n)$, neither the time $c^{S(n)}$, nor the branching number $S(n)$. On one worktape, M' memorizes the string which encodes the visited path and also records the values ("accept" or "reject" or "unknown") given by M to the nodes which are immediately branching to the left of the nodes of this path. On the other tapes, M' works exactly as M does.

In case $S(n)$ is not fully space constructible, M' iterates the above computation for successive values $S(n) = 1, 2, ...$. ∎

We shall also use the following

LEMMA 3.8 (Chandra *et al.*, 1981). *If* $S(n) \geqslant \log n$, *then* $\mathrm{ASPACE}(S(n)) \subseteq \bigcup_{c>0} \mathrm{DTIME}(c^{S(n)})$.

In order that the algorithm TRUTH be as efficient as possible, we shall put sentences into a "standard form."

DEFINITION. A sentence φ is said to be in *standard form* if all the variables of φ are exactly $x_1, x_2, ..., x_m$ for an integer m.

If a sentence φ in standard form has length n, then φ has only $m = O(n/\log n)$ distinct variables and thus the binary representation of the integer m has length $O(\log n)$.

LEMMA 3.9. *There is a* DTM *working in space* $\log n$, *which puts any sentence* φ *in a standard form* φ' *equivalent to* φ. *Moreover* $|\varphi'| \leqslant |\varphi|$.

Proof. We can suppose that the only variables of φ are of the form x_i. The sentence φ' will be obtained by replacing in φ each occurrence of any variable x_i by $x_{i'}$, where i' is the number of all the distinct indices $j \leqslant i$ such that x_j occurs in φ.

We shall exhibit a special DTM M working in space $\log n$ with four heads numbered 1, 2, 3, and 4 (instead of one head) on the input tape. Clearly such a machine can be simulated by an ordinary DTM using additional space $\log n$ to simulate the four input heads (see Savitch, 1973, for more details). Roughly M works on an input φ as follows. Head 1 remains on the index i as long as i' is not found. Afterwards Head 1 will go to the next index immediately to the right of i in φ. Head 2 successively marks the elements of the increasing sequence $j_1, j_2, \ldots, j_{i'} = i$, where the j_k are all the distinct indices $\leqslant i$ of the variables of φ. When Head 2 marks j_k, the integer k is written on the worktape and Heads 3 and 4 do comparisons to find j_{k+1}. ∎

Lemma 3.9 will be useful because of the following (well-known) result.

LEMMA 3.10 (Stockmeyer and Meyer, 1973). *Let A and B be two languages such that $A \leqslant_{\log\text{-lin}} B$. Then we have:*

(i) $B \in \mathrm{DSPACE}(S(n))$ *implies* $A \in \mathrm{DSPACE}(S(cn) + \log n)$ *for some constant c;*

(ii) $B \in \mathrm{DTIME}(T(n))$ *implies* $A \in \mathrm{DTIME}(T(cn) + n^p)$ *for some constants c and p.*

LEMMA 3.11. *Let $d \geqslant 2$ be the arity of $\mathscr{S}$. There is an* ATM *which accepts the set of all the sentences of* $\mathrm{Th}(\mathscr{S})$ *in standard form, in space $S(n) = O((n/\log n)^d)$ with branching number $B(n) = O((n/\log n)^d \log n)$.*

Proof. It is sufficient to prove that for any sentence φ (of type $\mathscr{S}$) in standard form, any path of the alternating algorithm $\mathrm{TRUTH}(\varnothing, \varphi)$ only uses the space $S(n)$ and the branching number $B(n)$ of the lemma. Indeed we can simulate this algorithm with an ATM having no negating states and the simulation only requires the same space and the same branching number (see Chandra *et al.*, 1981, Theorem 2.5, for more details). For the sake of simplicity, suppose $\mathscr{S} = \{R\}$, R a d-ary relation symbol.

Clearly the space used by $\mathrm{TRUTH}(\varnothing, \varphi)$ is at most the maximal length of the encoding of a pair (Δ, ψ), where ψ is a formula obtained from a subformula of φ by eventually replacing several variables by other variables of φ and where Δ is a complete description of ψ. Suppose that the variables of φ are exactly $x_1, x_2, \ldots, x_m$. We have $|\psi| \leqslant |\varphi| \times |x_m| = O(n \log n)$, since $|x_m| = O(\log n)$. We encode Δ by a string $s = s_1 s_2 \cdots s_{m^d}$ of m^d symbols over the alphabet $\{0, 1, 2\}$ in the following manner. If the d-tuple of variables $(X_1, X_2, \ldots, X_d) \in \{x_1, \ldots, x_m\}^d$ is the ith d-tuple of variables (in the order corresponding to lexicographical order of the d-tuples of indices), then $s_i = 1$ iff $RX_1 \cdots X_d$ is a conjunct of Δ, $s_i = 0$ iff $\neg RX_1 \cdots X_d$ is a conjunct of Δ, and $s_i = 2$ iff some variable among $X_1, \ldots, X_d$ is not free in ψ. We have $|s| =$

$m^d = O((n/\log n)^d)$. Therefore we obtain the space bound $O(n \log n) + O((n/\log n)^d) = O((n/\log n)^d)$.

The algorithm TRUTH uses branching steps only in its instructions $(\vee)$, $(\wedge)$, $(\exists)$, and $(\forall)$. For every $(\vee)$ or $(\wedge)$ there is only one branching step. For every $(\exists)$ or $(\forall)$ there are $O(m^{d-1})$ branching steps; indeed we need $(k+1)^d - k^d = O(k^{d-1})$ branching steps to choose an individual description $D(\bar{v}_k, v)$, $k < m$, and in case v is replaced, we need only $O(\log m)$ branching steps to choose which v_i will replace v. For an input φ of length n, any computation path uses $(\vee)$, $(\wedge)$, $(\exists)$, and $(\forall)$ at most n times. Therefore we obtain the branching number $O(n) + O(nm^{d-1}) = O((n/\log n)^d \log n)$. ∎

THEOREM 3.12. *Let $d \geqslant 2$ be the arity of $\mathscr{S}$.*

(i) $\mathrm{Th}(\mathscr{S}) \in \mathrm{DSPACE}((n/\log n)^d \log n)$;

(ii) $\mathrm{Th}(\mathscr{S}) \in \mathrm{DTIME}(c^{(n/\log n)^d})$, *for a constant c.*

Proof. By Lemmas 3.9 and 3.10 it is sufficient to prove the theorem for the subset of all the sentences of $\mathrm{Th}(\mathscr{S})$ in standard form. The theorem is then a consequence of Lemma 3.11 completed by Lemma 3.7 for (i) and by Lemma 3.8 for (ii). ∎

Let us now consider the complexity of theories $(\mathbb{Q}, <)$ and QBF. By the same method as before, we obtain the

THEOREM 3.13. *Let $d \leqslant 1$ be the arity of $\mathscr{S}$;*

(i) $\mathrm{Th}(\mathscr{S})$ *and* $(\mathbb{Q}, <)$ *belong to* $\mathrm{DSPACE}(n \log n)$ (result of Ferrante and Geiser (1977) for $(\mathbb{Q}, <)$) *and belong to* $\mathrm{DTIME}(c^n)$ *for a constant c;*

(ii) $\mathrm{QBF} \in \mathrm{DSPACE}(n)$ (see Stockmeyer, 1976).

Remark. For QBF, a "complete description" of a formula φ is a *truth value assignment* of the free propositional variables of φ. For $(\mathbb{Q}, <)$, a "complete description" of a formula $\varphi(\bar{v}_m)$ is a string of the form

$$v_{\pi(1)} \alpha_1 v_{\pi(2)} \alpha_2 \cdots \alpha_{m-1} v_{\pi(m)},$$

where each α_i is $<$ or $=$ and π is a permutation of the set $\{1, 2, \ldots, m\}$. Such a string has length $O(n)$. For $\mathrm{Th}(\mathscr{S})$ it is easy to obtain a similar notion of "complete description" of length $O(n)$ if the arity of $\mathscr{S}$ is $\leqslant 1$.

In case the sentences are in prenex form, we can improve the above space upper bounds. Indeed a prenex sentence of length n has only $O(n/\log n)$ quantifiers; hence the branching number of any path of the algorithm TRUTH is divided by $\log n$. Clearly we have the

COROLLARY 3.14. *Let d be the arity of $\mathscr{S}$;*

(i) *if* $d \geqslant 2$, *then* $\Pr \mathrm{Th}(\mathscr{S}) \in \mathrm{DSPACE}((n/\log n)^d)$;

(ii) *if* $d \leqslant 1$, *then* $\Pr \mathrm{Th}(\mathscr{S})$ *and* $\Pr(\mathbb{Q}, <) \in \mathrm{DSPACE}(n)$.

Part (i) of the following simple corollary improves the linear bound of Stockmeyer (1976).

COROLLARY 3.15. (i) $\Pr \mathrm{QBF} \in \mathrm{DSPACE}(n/\log n)$;

(ii) $\mathrm{QBF} \in \mathrm{DTIME}(c^{n/\log n})$, *for a constant* c.

Using the alternating classes TIME, ALT(–, –) we can improve Corollary 3.14.

LEMMA 3.16. *If* $A \in \mathrm{DSPACE}(\log n)$ *then* $A \in \mathrm{TIME}, \mathrm{ALT}(n, \log n)$.

Proof. It is exactly the proof of Savitch's (1970) theorem and of Theorem 3.1 of Chandra *et al.* (1981). ∎

THEOREM 3.17. *Let* d *be the arity of* $\mathscr{S}$;

(i) *if* $d \geqslant 2$, *then* $\Pr \mathrm{Th}(\mathscr{S}) \in \mathrm{TIME}, \mathrm{ALT}((n/\log n)^d, n/\log n)$;

(ii) *if* $d \leqslant 1$, *then* $\Pr \mathrm{Th}(\mathscr{S}) \in \mathrm{TIME}, \mathrm{ALT}(n, n/\log n)$.

Remark. Note that $\mathrm{TIME}, \mathrm{ALT}(T, A) \subseteq \mathrm{ATIME}(T) \subseteq \mathrm{DSPACE}(T)$.

Proof of Theorem 3.17. We only give the main ideas and let the reader imagine the technical details by using the proof of Lemma 3.11.

Starting from a prenex sentence, φ, of type $\mathscr{S}$, our algorithm works as follows:

(1) Existentially guess a sentence φ' such that $|\varphi'| \leqslant |\varphi|$.

(2) Universally execute the following subroutines, (a) and (b):

(a) Check if φ' is a standard form of φ. (Moreover we can suppose that φ' is of the form $Q_1 x_1 \cdots Q_m x_m \psi$, where $Q_1 x_1 \cdots Q_m x_m$ is the prefix.)

(b) Check if $\varphi' \in \Pr \mathrm{Th}(\mathscr{S})$ with the following implementation of the algorithm TRUTH:

(b_1) Guess (with existential or universal states according to the quantifiers) a sequence of symbols, Δ, representing the connections between variables $x_1, \ldots, x_m$ (i.e., a "complete description") with respect to the equality relation and the symbols of $\mathscr{S}$.

(b_2) Check if Δ makes ψ true.

Let us analyse this alternating algorithm in case $d \geqslant 2$ (case $d \leqslant 1$ is similar). Part (1) works in time $O(n)$, where $n = |\varphi|$. The subroutine (a) works in DSPACE(log n) by Lemma 3.9 and then in TIME, ALT(n, log n) by Lemma 3.16. (b_1): Our "complete description" is a word of length

$O(m^d) = O(n/\log n)^d$, which can be guessed in TIME, ALT$((n/\log n)^d, n/\log n)$. (b_2): The problem of checking if $\mathcal{A}$ makes ψ true belongs to DSPACE($\log p$), where $p = |\mathcal{A}| + |\psi|$ and then belongs to TIME, ALT$((n/\log n)^d, \log n)$, where $n = |\varphi|$. ∎

4. Lower Bounds of Complexity

Clearly the subtheory of Th($\mathcal{S}$) consisting of all sentences which contain no relation symbol, but only the equality symbol, is the theory of the equality on an infinite domain. Henceforth let $\mathcal{E}$ denote any theory of the equality on a class of domains of unbounded cardinalities. It is implicitly proved by Stockmeyer (1976) that $\mathcal{E}$ is $\leqslant_{\log}$-*hard* for PSPACE and that Pr $\mathcal{E} \notin$ NSPACE($o(n^{1/2})$). In particular Th($\mathcal{S}$) is $\leqslant_{\log}$-*complete* in PSPACE (by Section 3). In this section we will improve the space lower bound of Th($\mathcal{S}$) in case the arity of $\mathcal{S}$ is $d \geqslant 2$. (We will insist on the case $d \geqslant 2$ because we feel that it is the most interesting case.)

We need some new definitions and results about complexity classes.

Definitions. Let $\mathscr{C}$ be a class of languages, for example, a complexity class. We write $\mathscr{C} \leqslant_{\log} B$ *via length order* $l(n)$ (resp. *via time order* $t(n)$) to mean that for each $A \in \mathscr{C}$, $A \subseteq J^+$, there is a reduction f_A from language A to language B, computable by a DTM in space $\log n$, such that $|f_A(w)| \leqslant c_A l(|w|)$, for some constant c_A and any $w \in J^+$ (resp. computable by a DTM both in space $\log n$ and time $c_A t(n)$). In case $\mathscr{C} = \{A\}$ we write $A \leqslant_{\log} B$ *via* ....

We shall use the classes NTIME(T) and NSPACE(S), and also the following class. NTISP(T, S) is the class of the sets A for which there is a NTM M which accepts A, so that for each $w \in A$ of length n, there is an accepting computation of M on w, using time $\leqslant T(n)$ and space $\leqslant S(n)$. Such a machine M is called a NTISP(T, S)-*machine* (Bruss and Meyer, 1980).

An efficient reduction as described above permits us to transfer complexity results because of the following lemma.

Lemma 4.1. *Suppose that S and T are nondecreasing functions from the set of integers to the set of real numbers.*

(i) *If $A \leqslant_{\log} B$ via length order $l(n)$ and $B \in$ NSPACE($S(n)$) then $A \in$ NSPACE($S(cl(n)) + \log n$) for a constant c.*

(ii) *If $A \leqslant_{\log} B$ via time order $t(n)$ and $B \in$ NTIME($T(n)$), then $A \in$ NTIME($T(ct(n)) + ct(n)$) for a constant c.*

(iii) *If $A \leqslant_{\log} B$ via time order $t(n)$ and $B \in$ NTISP($T(n), S(n)$) then $A \in$ NTISP($T(ct(n))\, c_1 t(n), S(ct(n)) + \log n$) for constants c, c_1.*

Proof. Part (ii) is obvious. The proof of (i) and (iii) is similar to the proof of the transitivity of reducibilities $\leqslant_{\log}$ and $\leqslant_{\log\text{-lin}}$ (Stockmeyer and Meyer, 1973, Lemmas 2.1 and 2.2). We explain the factor $c_1 t(n)$ in the time bound of (iii) by the fact that to save space, the NTM M' which accepts A does not write the entire output $f(w)$ of the reduction f from A to B: let M be the NTISP(T, S)-machine which accepts B; at any step of the simulation of M (on input $f(w)$) which concerns the jth symbol of $f(w)$, M' simulates all the computation of f on w till it finds the jth symbol. ∎

DEFINITION. A pair of functions (T, S) is *compatible* if there is a DTM which, on each input of length n, halts with two outputs of lengths $T(n)$ and $S(n)$ exactly, using only a time $O(T(n))$ and space $S(n)$.

Remark. We prefer this notion of compatibility rather than the stronger notion of Bruss and Meyer (1980) because it is sufficient for our results and because we prove more easily that a pair of functions, for example, $(2^n, n^k)$, is compatible in our sense.

For proving lower bounds of complexity, we shall use the following hierarchy theorem.

THEOREM 4.2 (Seiferas, 1977a,b; Seiferas *et al.*, 1978; Bruss and Meyer, 1980). *Let* S_2 *and* T_2 *be functions such that* $\log n = o(S_2(n))$ *and* $n = o(T_2(n))$.

(i) *If* S_2 *is constructible (by a* DTM*) in space* S_2*, then there is a set* $A_1 \in \text{NSPACE}(S_2) - \bigcup\{\text{NSPACE}(S_1) \mid S_1(n+1) = o(S_2(n))\}$.

(ii) *If* T_2 *is constructible (by a* DTM*) in time* $O(T_2)$*, then there is a set* $A_2 \in \text{NTIME}(T_2) - \bigcup\{\text{NTIME}(T_1) \mid T_1(n+1) = o(T_2(n))\}$.

(iii) *If the pair* (T_2, S_2) *is compatible, then there is a set* $A_3 \in \text{NTISP}(T_2, S_2) - \bigcup\{\text{NTISP}(T_1, S_1) \mid T_1(n+1) = o(T_2(n))$ *and* $S_1(n+1) = o(S_2(n))\}$.

Moreover we can take each $A_i \subseteq \{0, 1\}^+$.

Proofs. Parts (i) and (ii) are proved in Seiferas, 1977a,b, and Seiferas *et al.*, 1978, respectively. Part (iii) is proved in a similar (and succinct) manner in Bruss and Meyer, 1980. ∎

Now for proving lower bounds of complexity, we only need (by the results above) to prove that a complexity class can be efficiently reduced to our theory. For example, it is proved by Stockmeyer (1976) that Pr QBF $\notin$ NSPACE$(o((n/\log n)^{1/2}))$ because of the reduction NSPACE$(n) \leqslant_{\log}$ Pr QBF via length order $n^2 \log n$. In fact, in analysing each reduction of Stockmeyer (1976), we see that any "*via length order*" can be replaced by "*via time*

order." The reductions we exhibit in the following still have this property, and therefore they are not only *space efficient* but also *time efficient.*

The following easy lemma is the only model-theoretic argument which is used in this paper for proving lower bounds of $\mathrm{Th}(\mathscr{S})$.

LEMMA 4.3. *Let $\mathscr{N}$ be a finite structure of type $\mathscr{S}$. The property of containing $\mathscr{N}$ as a substructure is true a.e. (In other words, almost all finite structures of type $\mathscr{S}$ contain $\mathscr{N}$ as a substructure.)*

LEMMA 4.4. *Let $d \geqslant 2$ be the arity of $\mathscr{S}$. Then for any integer k, $\mathrm{NTISP}(n^k, (n \log n)^d/\log n) \leqslant_{\log} \mathrm{Pr}\,\mathrm{Th}(\mathscr{S})$ via time order $n(\log n)^2$.*

Proof of Lemma 4.4. The method is similar to that of Stockmeyer (1976, Lemma 6.3). Therefore we emphasize the differences. The proof is divided into two parts. In part 1 we show that for any $A \in \mathrm{NTISP}(n^k, (n \log n)^d/\log n)$, there is a type $\mathscr{S}_1$ of arity d such that $A \leqslant_{\log} \mathrm{Th}(\mathscr{S}_1)$ via time order $n(\log n)^2$. In part 2, we show that $\mathscr{S}_1$ can be replaced by $\mathscr{S}_2 = \{R\}$, where R is a d-ary relation symbol, which proves the lemma for any $\mathscr{S}$ of arity d.

Part 1. Let M be a $\mathrm{NTISP}(T, S)$-machine which accepts a set A, with $T(n) = n^k$ and $S(n) = (n \log n)^d/\log n = (n \log n)^{d-1} \cdot n$. Without loss of generality we can assume that M has only one tape, one-way infinite to the right, that is both an input tape and a worktape, because if $S(n) \geqslant n$, any $\mathrm{NTISP}(T, S)$-machine can be simulated by a one-tape $\mathrm{NTISP}(cT^2, S)$-machine, for a constant c.

Let Γ, Q, and h be respectively the tape alphabet of M, the set of states of M, and a special symbol for the tape head.

We identify an instantaneous description (ID) of length l of M with a string $\delta \in \Sigma^l$, where $\Sigma = \Gamma \cup (\Gamma \times \{h\} \times Q)$. In a natural manner, the string δ describes the tape at a given instant, including the head position and the state.

For each symbol $\sigma \in \Sigma$, we choose a d-ary relation symbol R_σ and take $\mathscr{S}_1 = \{R_\sigma \mid \sigma \in \Sigma\}$. For every input w of M, we will construct a sentence φ'_w of type $\mathscr{S}_1$ so that $w \in A$ iff φ'_w is true a.e. In fact we shall take $\varphi'_w \equiv \exists \bar{x}_m \psi_w(\bar{x}_m)$, where for $|w| = n$, $m = \lceil n \log n \rceil$ and $|\psi_w| = O(n(\log n)^2)$. Now let us give an approximate idea for understanding the construction of the formula ψ_w. (In the following we take $d = 2$ but the proof is similar in the other cases.)

We encode an accepting computation of M on input w in a structure $\mathscr{M}$ of type $\mathscr{S}_1$. Suppose that the value of $\bar{x}_m$ is fixed to the tuple $\bar{a}_m$ of elements in $\mathscr{M}$. An ID δ of length $l = (\lceil n \log n \rceil) \cdot n = m \cdot n$ can be written as a doubly indexed sequence (δ_{ij}) (where $i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, n\}$), ordered by the lexicographical order of pairs (i, j). Thus δ can be encoded by an n-tuple $\bar{b}_n$ of elements in $\mathscr{M}$. More precisely we take $\langle \mathscr{M}, a_i, b_j \rangle \models R_\sigma(x_i, y_j)$ iff

$\delta_{ij} = \sigma$. Therefore an accepting computation of M, using time n^k and space $l = m \cdot n$ can be encoded by a sequence of n^k n-tuples $\bar{b}_n, \ldots, \bar{b}'_n$ of elements in $\mathcal{M}$.

We use two techniques to write "short formulas", i.e., formulas of length $O(n(\log n)^2)$: first the "divide and conquer" argument of Stockmeyer (1976) which uses the idea of the well-known Savitch theorem; secondly a "folding" technique to describe long IDs by short formulas (see Lynch, 1982).

For each $\sigma \in \Sigma$, let $\underline{R}_\sigma(v, v')$ denote the formula $R_\sigma vv' \wedge \bigwedge_{\sigma' \neq \sigma} \neg R_{\sigma'} vv'$.

As in Stockmeyer (1976) it is easy to construct a formula $\mathrm{Init}_w(\bar{y}_n)$ (resp. $\mathrm{Acc}(\bar{y}_n)$) which asserts that $\bar{y}_n$ encodes the initial ID on input w (resp. an accepting ID). (In fact we must write $\mathrm{Init}_w(\bar{x}_m, \bar{y}_n)$ and $\mathrm{Acc}(\bar{x}_m, \bar{y}_n)$, but in all the formulas we omit mentioning the $\bar{x}_m$, which can be considered roughly as "fixed parameters.") Since the initial ID is of the form $\tilde{w}B^{l-n}$, where $|\tilde{w}| = |w| = n$ and B^{l-n} is the blank symbol B repeated $l - n$ times, we take

$$\mathrm{Init}_w(\bar{y}_n) \equiv \bigwedge_{1 \leqslant i \leqslant n} \underline{R}_{\tilde{w}_i}(x_1, y_i) \wedge \forall t \, \forall u \left(\left(\bigvee_{2 \leqslant i \leqslant m} t = x_i \wedge \bigvee_{1 \leqslant i \leqslant n} u = y_i \right) \Rightarrow \underline{R}_B(t, u) \right),$$

where $\tilde{w}_i$ is the ith symbol of $\tilde{w}$. Clearly $|\mathrm{Init}_w| = O(n(\log n)^2)$. The formula $\mathrm{Acc}(\bar{y}_n)$ is constructed in a similar manner.

For two IDs δ and δ', let $\delta \vdash_M^j \delta'$ mean that δ' follows from δ in j steps of computation of M. Let p denote the integer $\lceil k \log n \rceil$, so that $n^k \leqslant 2^p$. We shall give a formula, called $\mathrm{Step}(\bar{y}_n, \bar{y}'_n)$ (resp. $\mathrm{Comp}(\bar{y}^p_n, \bar{z}^p_n)$), which asserts that if $\bar{y}_n$ (resp. $\bar{y}^p_n$) encodes an ID δ of length l, then $\bar{y}'_n$ (resp. $\bar{z}^p_n$) encodes an ID δ' and $\delta \vdash_M^1 \delta'$ (resp. $\delta \vdash_M^{2^p} \delta'$). Suppose that the formula Step is given. Then Comp will be as in Stockmeyer (1976):

$$\mathrm{Comp}(\bar{y}^p_n, \bar{z}^p_n) \equiv \exists \bar{t}^p_n \, \forall \bar{y}^{p-1}_n \, \forall \bar{z}^{p-1}_n \, \exists \bar{t}^{p-1}_n \cdots \exists \bar{t}^1_n \, \forall \bar{y}^0_n \, \forall \bar{z}^0_n$$

$$\left[\bigwedge_{1 \leqslant i \leqslant p} ((\bar{y}^{i-1}_n = \bar{y}^i_n \wedge \bar{z}^{i-1}_n = \bar{t}^i_n) \vee (\bar{y}^{i-1}_n = \bar{t}^i_n \wedge \bar{z}^{i-1}_n = \bar{z}^i_n)) \Rightarrow \mathrm{Step}(\bar{y}^0_n, \bar{z}^0_n) \right],$$

where each equality $\bar{v}_n = \bar{v}'_n$ abbreviates the conjunction $\bigwedge_{1 \leqslant i \leqslant n} v_i = v'_i$. Clearly $|\mathrm{Comp}| = |\mathrm{Step}| + p \cdot O(n \log n) = |\mathrm{Step}| + O(n(\log n)^2)$.

Let ψ_w be the prenex form of the following formula of simple meaning:

$$\exists \bar{y}^p_n \, \exists \bar{z}^p_n \, (\mathrm{Init}_w(\bar{y}^p_n) \wedge \mathrm{Comp}(\bar{y}^p_n, \bar{z}^p_n) \wedge \mathrm{Acc}(\bar{z}^p_n)).$$

It remains to construct the formula Step. As in Stockmeyer (1976) we use the following fact proved by Stockmeyer (1974, pp. 38–39). There is a set $X_M \subseteq \Sigma^3 \times \Sigma^3$ such that for any two strings $\delta = \delta_1 \cdots \delta_l$ and $\delta' = \delta'_1 \cdots \delta'_l$ in Σ^l, if δ is an ID of M then: δ' is an ID of M such that $\delta \vdash_M^1 \delta'$ iff for each

$i \in \{1, 2, ..., l-2\}$ the 6-tuple $(\delta_i \delta_{i+1} \delta_{i+2} \cdot \delta'_i \delta'_{i+1} \delta'_{i+2}) \in X_M$. For two ordered pairs (i,j) and (i',j') of $\{1,...,m\} \times \{1,...,n\}$, let us write $(i,j)\,\mathrm{suc}(i',j')$ to mean that (i',j') is the successor of (i,j) in the lexicographical order. If we take

$$\mathrm{Step}(\bar{y}_n, \bar{y}'_n) \equiv \bigwedge \bigvee_{\tau \in X_M} \bigwedge_{k=1,2,3} (\underline{R}_{\sigma_k}(x_{i_k}, y_{j_k}) \wedge \underline{R}_{\sigma'_k}(x_{i_k}, y'_{j_k}))$$

where τ abbreviates $(\sigma_1\sigma_2\sigma_3 \cdot \sigma'_1\sigma'_2\sigma'_3)$ and the last conjunction is taken for all pairs such that $(i_1,j_1)\,\mathrm{suc}(i_2,j_2)\,\mathrm{suc}(i_3,j_3)$, then the formula Step is too long since there are $mn-2$ possible ordered pairs (i_1,j_1). By a "folding" technique we construct the equivalent formula:

$$\begin{aligned}
\mathrm{Step}(\bar{y}^0_n, \bar{y}^1_n) \equiv \forall \bar{t}_3\, \forall \bar{u}^0_3\, \forall \bar{u}^1_3 \Bigg[\Bigg[\Bigg(& \bigvee_{1 \leqslant i \leqslant m} (t_1 = t_2 = t_3 = x_i) \\
& \wedge \bigvee_{1 \leqslant i \leqslant n-2} \bigwedge_{j=0,1} (u^j_1 = y^j_i \wedge u^j_2 = y^j_{i+1} \wedge u^j_3 = y^j_{i+2}) \Bigg) \\
& \vee \Bigg(\bigvee_{1 \leqslant i \leqslant m-1} (t_1 = t_2 = x_i \wedge t_3 = x_{i+1}) \\
& \wedge \bigwedge_{j=0,1} (u^j_1 = y^j_{n-1} \wedge u^j_2 = y^j_n \wedge u^j_3 = y^j_1) \Bigg) \\
& \vee \Bigg(\bigvee_{1 \leqslant i \leqslant m-1} (t_1 = x_i \wedge t_2 = t_3 = x_{i+1}) \\
& \wedge \bigwedge_{j=0,1} (u^j_1 = y^j_n \wedge u^j_2 = y^j_1 \wedge u^j_3 = y^j_2) \Bigg) \Bigg] \\
\Rightarrow & \bigvee_{\tau \in X_M} \bigwedge_{j=1,2,3} (\underline{R}_{\sigma_j}(t_j, u^0_j) \wedge \underline{R}_{\sigma'_j}(t_j, u^1_j)) \Bigg],
\end{aligned}$$

where y^0_i, y^1_i, u^0_j, u^1_j are written in place of y_i, y'_i, u_j, u'_j, respectively, and τ abbreviates $(\sigma_1\sigma_2\sigma_3 \cdot \sigma'_1\sigma'_2\sigma'_3)$. Clearly $|\mathrm{Step}| = O(n(\log n)^2)$.

The work of the DTM which on input w produces $\varphi'_w \equiv \exists \bar{x}_m \psi_w(\bar{x}_m)$ as output consists essentially in counting from 1 to m (resp. from 1 to n) $O(1)$ times (resp. $O(p)$ times) and writing at the same time these lists of integers on the output tape. The space needed is $\max(|x_m|, |x_n|, |x_p|) = O(\log n)$ and the time needed is

$$O(m \log m) + O(p) \cdot O(n \log n) = O(n(\log n)^2).$$

We can roughly say that a structure $\mathscr{M}$ of type $\mathscr{S}_1$ is a model of the sentence φ'_w iff $\mathscr{M}$ contains a "substructure which encodes" an accepting computation of M on input w. Therefore by Lemma 4.3, M accepts w iff φ'_w is true a.e.

Part 2. Now let us construct a modified version of φ'_w, denoted φ_w, of type $\mathscr{S}_2 = \{R\}$ for R a d-ary relation symbol. Let $\sigma,\ldots,\sigma'$ denote the list of the elements of Σ. The idea is roughly the following. In a structure $\mathscr{M}$ of type $\mathscr{S}_1$ which contains a substructure encoding an accepting computation, we replace each element a by distinct elements $a^{\sigma},\ldots,a^{\sigma'}$ and construct a structure $\mathscr{M}'$ of type $\mathscr{S}_2$ on this new domain, so that for any d-tuple $\bar{a}_d$ of elements of $\mathscr{M}$ and for each $\sigma \in \Sigma$, $\langle \mathscr{M}', \bar{a}_d^{\sigma}\rangle \models R\bar{x}_d^{\sigma}$ iff $\langle \mathscr{M}, \bar{a}_d\rangle \models R_{\sigma}\bar{x}_d$. The sentence φ_w is the sentence φ'_w in which each variable v is replaced by the list of new variables $v^{\sigma},\ldots,v^{\sigma'}$. More precisely each quantified variable Qv of φ'_w is replaced by $Qv^{\sigma}\cdots Qv^{\sigma'}$; subformulas $R_{\sigma}\bar{v}_d$ and $v = v'$ are replaced by $R\bar{v}_d^{\sigma}$ and $\bigwedge_{\sigma\in\Sigma} v^{\sigma} = v'^{\sigma}$, respectively. (Henceforth we do not require that distinct metavariables denote distinct variables.) Clearly M accepts w iff φ_w is true a.e. ∎

In the previous lemma, we have tried to *maximize* the *space bound* of the class NTISP(–, –). In the following lemma, we try to *maximize* the *time bound.*

LEMMA 4.5. *Let $d \geqslant 2$ be the arity of $\mathscr{S}$. Then for any integer c we have*:

(i) $\mathrm{NTISP}(2^{cn}, n^{d-1}) \leqslant_{\log} \mathrm{Pr\ Th}(\mathscr{S})$ *via time order* $n \log n$;

(ii) $\mathrm{NTISP}(2^{cn\log n}, n^{d-1}) \leqslant_{\log} \mathrm{Th}(\mathscr{S})$ *via time order* $n \log n$.

Proof. Similar to the proof of Lemma 4.4. For (i) we only modify the cardinalities: the constant m-tuple $\bar{a}_m$ is replaced by an n-tuple $\bar{a}_n$, i.e., $m = \lceil n \log n\rceil$ is replaced by n; every ID of length n^{d-1} is now encoded by only one element b (instead of an n-tuple $\bar{b}_n$); instead of $p = \lceil k \log n\rceil$, we take $p = cn$.

Part (ii) is proved as (i), except that we take $p = \lceil cn \log n\rceil$ and define a new (non-prenex) formula $\mathrm{Comp}(y, z)$ as follows. Define the formulas $F_j(y, z)$ and $F_j(y', z')$ by induction on the integer j, starting from the formula $F_0(y, z) \equiv \mathrm{Step}(y, z)$ defined as in (i). $F_{j+1}(y, z)$ is the formula

$$\exists t\, \forall y'\, \forall z'\, [((y' = y \wedge z' = t) \vee (y' = t \wedge z' = z)) \Rightarrow F_j(y', z')].$$

$F_{j+1}(y', z')$ is the formula $F_{j+1}(y, z)$ in which y is replaced by y' and vice versa, z is replaced by z' and vice versa; in particular the subformula $F_j(y', z')$ is replaced by $F_j(y, z)$. We take $\mathrm{Comp}(y, z) \equiv F_p(y, z)$. Since each variable y, z, y', z', t occurs $O(p)$ times in F_p, we have $|\mathrm{Comp}| = O(p) = O(n \log n)$. ∎

Remark. In the case $d = 2$, we have $\mathrm{NTISP}(2^{cn\log n}, n^{d-1}) = \mathrm{NSPACE}(n)$ and we shall use the better lemma:

LEMMA 4.6. *If the arity of* $\mathscr{S}$ *is* 2, *then* $\mathrm{NSPACE}(n(\log n/\log\log n)^{1/2}) \leqslant_{\log} \mathrm{Th}(\mathscr{S})$ *via time order* $n \log n$.

Proof. Using a method similar to that of the proof of Lemma 4.5(ii), we want to maximize a function S such that for any integer c, $\mathrm{NTISP}(2^{cS}, S) \leqslant_{\log} \mathrm{Th}(\mathscr{S})$ via time order $n \log n$, which is equivalent to: $\mathrm{NSPACE}(S) \leqslant_{\log} \mathrm{Th}(\mathscr{S})$ via time order $n \log n$. As for Lemma 4.5, we use the fixed n-tuple $\bar{a}_n$; but now any ID is encoded by an $f(n)$-tuple $\bar{b}_{f(n)}$ instead of only one element. (The function $f(n)$ will be defined below.) We now take $p = \lfloor c' n \log n/(f(n)\log f(n))\rfloor$ for a constant integer c'; thus in the sentence φ'_w the sum of the lengths of the encodings of p IDs is $O(n \log n)$. Our formula $\mathrm{Comp}(\bar{y}_{f(n)}, \bar{z}_{f(n)})$ is exactly similar to the formula $\mathrm{Comp}(y, z)$ of Lemma 4.5(ii). Clearly a tuple $\bar{b}_{f(n)}$ can encode an ID of length $n \cdot f(n)$ and the usual "divide and conquer" argument permits us to express a computation of time 2^p with a formula of length $O(n \log n)$. In order to maximize S, choose $f(n)$ so that the numbers $n \cdot f(n)$ and $n \log n/(f(n)\log f(n))$ are of the same order. We obtain $f(n) = \lceil (\log n/\log\log n)^{1/2} \rceil$ by an easy computation and therefore $S(n) = n \cdot f(n)$. ∎

Lemmas 4.4, 4.5, and 4.6 give us lower bounds of complexity, by Lemma 4.1 and Theorem 4.2. Let $d \geqslant 2$ be the arity of $\mathscr{S}$.

THEOREM 4.7. *For any integer* k, $\Pr \mathrm{Th}(\mathscr{S}) \notin \mathrm{NTISP}(n^k, o((n/\log n)^d/\log n))$.

THEOREM 4.8. (i) *There is a constant* $\varepsilon > 0$ *such that* $\Pr \mathrm{Th}(\mathscr{S}) \notin \mathrm{NTISP}(2^{\varepsilon n/\log n}, o(n/\log n)^{d-1})$;

(ii) *if* $d \geqslant 3$, *there is a constant* $\varepsilon > 0$ *such that* $\mathrm{Th}(\mathscr{S}) \notin \mathrm{NTISP}(2^{\varepsilon n}, o(n/\log n)^{d-1})$.

THEOREM 4.9. *If* $d = 2$, *then* $\mathrm{Th}(\mathscr{S}) \notin \mathrm{NSPACE}(o(n/(\log n \log\log n)^{1/2}))$.

Proofs. For proving Theorem 4.8(i), we use a set $A \in \mathrm{NTISP}(T_2, S_2) - \bigcup\{\mathrm{NTISP}(T_1, S_1) \mid T_1(n+1) = o(T_2(n))$ and $S_1(n+1) = o(S_2(n))\}$, where $T_2(n) = 2^n$ and $S_2(n) = n^{d-1}$. By Lemma 4.5(i), there is a reduction from A to $\Pr \mathrm{Th}(\mathscr{S})$, computable by a DTM in space $\log n$ and time $cn \log n$, for a constant c. By Lemma 4.1, we obtain the desired result with $\varepsilon < 1/c$. Proofs of Theorems 4.8(ii), 4.7, and 4.9 are similar. ∎

Remark. Except for Theorem 4.9, our lower bounds of complexity for $\Pr \mathrm{Th}(\mathscr{S})$ (resp. $\mathrm{Th}(\mathscr{S})$) are in terms of classes $\mathrm{NTISP}(T, o(S))$ with $S(n) \cdot \log T(n) \geqslant (n/\log n)^d$ (resp. $\geqslant (n/\log n)^d \log n$). Note that this last function is exactly our upper bound in terms of class DSPACE (Theorems 3.12 and

3.14). It seems difficult to improve our results because the best inclusion that we know between NTISP classes and DSPACE classes is $\mathrm{NTISP}(T, S) \subseteq \mathrm{DSPACE}(S \log T)$ (Savitch, 1970, Theorem 2). In the same manner, Theorem 4.9 is optimal modulo a factor $(\log \log n)^{1/2}$ and modulo the open problem $\mathrm{NSPACE}(S) = ?\ \mathrm{DSPACE}(S^2)$.

Let $d \geqslant 2$ be the arity of $\mathscr{S}$.

COROLLARY 4.10. $\Pr \mathrm{Th}(\mathscr{S}) \notin \mathrm{NTIME}(o((n/\log n)^d/\log n))$.

COROLLARY 4.11. (i) $\Pr \mathrm{Th}(\mathscr{S}) \notin \mathrm{NSPACE}(o(n/\log n))$;

(ii) *if* $d \geqslant 3$, $\mathrm{Th}(\mathscr{S}) \notin \mathrm{NSPACE}(o(n))$.

Proofs. Each of these complexity classes is included in one of the classes of Theorems 4.7 and 4.8. ∎

Remarks. It is implicitly proved by Berman (1980a), Paterson (1972), and Paul *et al.* (1980) that for any function $T(n) \geqslant n$, there is an integer k such that $\mathrm{NTIME}_1(T(n)) \subseteq \mathrm{NTISP}(T(n)^k, T(n)^{1/2})$, where $\mathrm{NTIME}_1(T(n))$ denotes the class of languages accepted by *one-tape* NTMs in time $T(n)$. As a consequence, Theorem 4.7 implies that

$$\Pr \mathrm{Th}(\mathscr{S}) \notin \mathrm{NTIME}_1(o((n/\log n)^{2d}/(\log n)^2)).$$

A similar result appears in Berman (1980a) for QBF.

In proofs of Lemma 4.4–4.6 as in Stockmeyer (1976) we divide the time of a computation into two parts and then successively into 4, 8,..., 2^p parts. We can slightly improve the lower space bound of $\Pr \mathrm{Th}(\mathscr{S})$ by dividing the time into n parts and then successively into $n^2, n^3, \ldots, n^{cn}$ parts (for a constant c). Then we obtain

$$\text{if} \quad d \geqslant 4, \Pr \mathrm{Th}(\mathscr{S}) \notin \mathrm{NSPACE}(o(n))$$

and

$$\text{if} \quad d = 3, \Pr \mathrm{Th}(\mathscr{S}) \notin \mathrm{NSPACE}(o(n/(\log n)^{1/2})).$$

THEOREM 4.12. *For any integer* k, *we have*

(i) $\Pr \mathrm{QBF} \notin \mathrm{NTISP}(n^k, o(n/(\log n)^2))$;

(ii) $\Pr \mathscr{E} \notin \mathrm{NTISP}(n^k, o(n/\log n))$.

Proof. Similar to the proof of Stockmeyer (1976, Corollaries 6.6 and 6.7). ∎

Remarks. A result similar to (i) appears in Berman (1980a). By Theorem 4.12 any improvement of the upper bounds obtained in Section 3

for Pr QBF, $\mathrm{Pr}(\mathbb{Q}, <)$ and Pr Th($\mathscr{S}$) (for an arity $d \leqslant 1$) would imply an improvement of the inclusion $\mathrm{NTISP}(T, S) \subseteq \mathrm{DSPACE}(S \log T)$.

DEFINITION (Lynch, 1982). Let φ be a sentence of relational type $\mathscr{S}_1 \cup \{\mathbf{U}, \mathbf{Suc}\}$, where **U** and **Suc** are respectively unary and binary relation symbols not in $\mathscr{S}_1$. The *spectrum* of φ, denoted $\mathrm{Sp}(\varphi)$, is the language $\subseteq \{0, 1\}^+$ defined as follows. For every word $w \in \{0, 1\}^+$, $w \in \mathrm{Sp}(\varphi)$ if and only if there is a model of φ on the domain $\{0, 1, \ldots, n-1\}$ such that

$$\langle \mathscr{M}, i \rangle \models \mathbf{U}(x) \qquad \text{iff} \quad w_i = 1$$

and

$$\langle \mathscr{M}, i, j \rangle \models \mathbf{Suc}(x, y) \qquad \text{iff} \quad i + 1 = j, \text{ for all } i, j \in \{0, \ldots, n-1\}.$$

Let **Spectra(arity** d**)** denote the class of spectra of sentences having a type of arity d.

We can improve Corollary 4.10 if we admit the nice and difficult result of J. Lynch (1982; 1980, Theorem 1.10) that follows.

THEOREM 4.13. *Any language* $A \subseteq \{0, 1\}^+$ *of* $\mathrm{NTIME}(n^d)$, *for an integer* $d \geqslant 2$, *belongs to* **Spectra(arity** d**)**.

COROLLARY 4.14. *If the arity of* $\mathscr{S}$ *is* $d \geqslant 2$, *then* $\mathrm{Pr\ Th}(\mathscr{S}) \notin \mathrm{NTIME}(o(n/\log n)^d)$.

Proof. By Theorem 4.13, Lemma 4.1, and Theorem 4.2, it is sufficient to prove that **Spectra(arity** d**)** $\leqslant_{\log}$ Pr Th($\mathscr{S}$) via time order $n \log n$.

Let $A \in$ **Spectra(arity** d**)**, i.e., $A = \mathrm{Sp}(\varphi)$, where φ is a sentence of type $\mathscr{S}_1 \cup \{\mathbf{Suc}, \mathbf{U}\}$ and the arity of $\mathscr{S}_1$ is d. From φ we will construct a reduction $w \mapsto \varphi_w$ from A to Pr Th($\mathscr{S}$). Let $w = w_0 w_1 \cdots w_{n-1}$ be a word of $\{0, 1\}^n$ and for each $i \in \{0, \ldots, n-1\}$ let x_i be a variable which does not occur in φ (x_i intuitively represents the integer i). φ_w is defined as follows.

Each variable of φ is "*relativized*" to $\{x_0, \ldots, x_{n-1}\}$: for example, $\exists v$ is replaced by $\exists v \, (\bigvee_{i<n} v = x_i \wedge \cdots)$. Each subformula $\mathbf{Suc}(v, v')$ is replaced by

$$\bigvee_{i<n-1} (v = x_i \wedge v' = x_{i+1}).$$

Let φ''_w be the prenex form of the conjunction of the formula so modified, of the conjunction $\bigwedge_{w_i=1} \mathbf{U}(x_i) \wedge \bigwedge_{w_i=0} \neg \mathbf{U}(x_i)$ and of the following formula of length $O(n \log n)$ which asserts that $x_0, \ldots, x_{n-1}$ are all distinct:

$$\exists y_0 \cdots \exists y_{p-1} \left[\bigwedge_{i<j<p} y_i \neq y_j \wedge \forall x \, \forall y \left[\left(\bigvee_{i<n} x = x_i \wedge \bigvee_{j<p} y = y_j \right) \Rightarrow x \neq y \right] \right.$$

$$\left. \wedge \text{“the formula } (Rxy \wedge Ry'x) \text{ defines a bijection between the elements } x \in \{x_0, \ldots, x_{n-1}\} \text{ and the couples } (y, y') \in \{y_0, \ldots, y_{p-1}\}^{2\text{”}} \right].$$

(For the sake of brevity, we assume that $n = p^2$ and that $\mathscr{S}_1$ contains a binary relation symbol, R, and we do not express explicitly the formula between commas.) The sentence φ_w is obtained from the sentence $\varphi'_w \equiv \exists x_0 \cdots \exists x_{n-1} \varphi''_w$ exactly as in part 2 of the proof of Lemma 4.4. ∎

AN OPEN PROBLEM. We conjecture that if a type $\mathscr{S}$ has a *smaller arity* than another type $\mathscr{S}'$, then $\mathrm{Th}(\mathscr{S})$ has a *strictly weaker* complexity than $\mathrm{Th}(\mathscr{S}')$. Unfortunately the results of the present paper are not sufficient for proving it, although they give us strong evidence of such a fact. We feel that this problem is closely related to the following open question (see Fagin, 1975): Is there a proper hierarchy of first-order spectra which rests on the arity of the types of sentences?

ACKNOWLEDGMENTS

I am thankful to K. Mac Aloon for making me aware of Fagin's work (1976). Thanks to Peter Clote and to the working group on complexity of University Paris 7. Warm thanks to Pascal Michel for helpful discussions, without which this paper would not have been possible.

RECEIVED: June 11, 1982

REFERENCES

BERMAN, L. C. (1980a), The complexity of QBF [An extended abstract], IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y. 10598.

BERMAN, L. C. (1980b), The complexity of logical theories, *Theoret. Comput. Sci.* **11**, 71–77.

BLASS, A., AND HARARY, F. (1979), Properties of almost all graphs and complexes, *J. Graph Theory* **3**, 225–240.

BRUSS, A. R., AND MEYER, A. R. (1980), On time-space classes and their relation to the theory of real addition, *Theoret. Comput. Sci.* **11**, 59–69.

CHANDRA, A. K., KOZEN, D. C., AND STOCKMEYER, L. J. (1981), Alternation, *J. Assoc. Comput. Mach.* **28**, No. 1, 114–133.

CHANG, C. C., AND KEISLER, H. J. (1973), “Model Theory,” North-Holland, Amsterdam.

FAGIN, R. (1975), A spectrum hierarchy, *Z. Math. Logik Grundlag. Math.* **21**, 123–134.

FAGIN, R. (1976), Probabilities on finite models, *J. Symbolic Logic* **41**, No. 1, 50–58.

FERRANTE, J., AND GEISER, J. R. (1977), An efficient decision procedure for the theory of rational order, *Theoret. Comput. Sci.* **4**, 227–233.

GAIFMAN, H. (1964), Concerning measures in first-order calculi, *Israel J. Math.* **2**, 1–18.

HOPCROFT, J. E., AND ULLMAN, J. D. (1979), "Introduction to Automata Theory, Languages and Computation," Addison–Wesley, Reading, Mass.

IMMERMAN, N. (1982), Upper and lower bounds for first-order expressibility, *J. Comput. System Sci.* **25**, 76–98.

LYNCH, J. F. (1980), Almost sure theories, *Ann. Math. Logic* **18**, 91–135.

LYNCH, J. F. (1982), Complexity classes and theories of finite models, *Math. Systems Theory* **15**, 127–144.

PATERSON, M. S. (1972), Tape bounds for time-bounded Turing machines, *J. Comput. System Sci.* **6**, 116–124.

PAUL, W. J., PRAUSS, E. J., AND REISCHUK, R. (1980), On alternation, *Acta Inform.* **14**, 243–255.

RUZZO, W. L. (1980), Tree-size bounded alternation, *J. Comput. System Sci.* **21**, 218–235.

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

SAVITCH, W. J. (1973), A note on multihead automata and context-sensitive languages, *Acta Inform.* **2**, 249–252.

SEIFERAS, J. I. (1977a), Techniques for separating space complexity classes, *J. Comput. System Sci.* **14**, 73–99.

SEIFERAS, J. I. (1977b), Relating refined space complexity classes, *J. Comput. System Sci.* **14**, 100–129.

SEIFERAS, J. I., FISCHER, M. J., AND MEYER, A. R. (1978), Separating nondeterministic time complexity classes, *J. Assoc. Comput. Mach.* **25**, 146–167.

STOCKMEYER, L. J. (1974), The complexity of decision problems in automata theory and logic, Doctoral thesis, Report TR-133, MIT, Project MAC, Cambridge, Mass.

STOCKMEYER, L. J. (1976), The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3**, 1–22.

STOCKMEYER, L. J., AND MEYER, A. R. (1973), Word problems requiring exponential time: Preliminary report, *in* "Proceedings, Fifth Annual Symposium on Theory of Computing," pp. 1–9.