Theoretical Computer Science 11 (1980) 59–69
© North-Holland Publishing Company

ON TIME-SPACE CLASSES AND THEIR RELATION TO THE THEORY OF REAL ADDITION*

Anna R. BRUSS

The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

Albert R. MEYER

Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

Communicated by M. S. Paterson
Received April 1978
Revised May 1979

Abstract. A new lower bound on the computational complexity of the theory of real addition and several related theories is established: any decision procedure for these theories requires either space $2^{\varepsilon n}$ or nondeterministic time $2^{\varepsilon n^2}$ for some constant $\varepsilon > 0$ and infinitely many n.

The proof is based on the families of languages TISP$(T(n), S(n))$ which can be recognized simultaneously in time $T(n)$ and $S(n)$ and the conditions under which they form a hierarchy.

1. Introduction

We consider the computational complexity of the theory of real addition, Th$\langle \mathbb{R}, +\rangle$, and several related theories. Previous results provide the following bounds on the complexity of Th$\langle \mathbb{R}, +\rangle$:

(1) *Lower bound* [6]. Any decision procedure for Th$\langle \mathbb{R}, +\rangle$ requires nondeterministic *time* $2^{\Omega(n)}$ for infinitely many n.[1]

(2) *Upper bound* [4]. Th$\langle \mathbb{R}, +\rangle$ is decidable within *space* $2^{O(n)}$.

Because the precise relation between computation time and space remains unknown, there is an exponential discrepancy when upper and lower bounds are both expressed in terms of time or space alone. That is, the exponential lower bound (1) for time is only known to imply a linear space lower bound; the exponential upper bound (2) for space is only known to imply a double exponential upper bound for time.

In this paper we improve the lower bound, showing in particular:

* This work was supported by NSF grant 77-19754MCS.

[1] $\Omega(f(n))$ denotes the set of all $g(n)$ such that there exist positive constants c and n_0 with $g(n) \geq cf(n)$ for all $n \geq n_0$[11].

Main Theorem. *There is an $\varepsilon > 0$ such that any decision procedure for* Th$\langle \mathbb{R}, + \rangle$ *requires either more than space $2^{\varepsilon n}$ or more than nondeterministic time $2^{\varepsilon n^2}$ for infinitely many n.*

Let (N)TISP($T(n)$, $S(n)$) be the family of languages recognizable by a (non)deterministic Turing machine which runs in time $T(n)$ and space $S(n)$ simultaneously for almost all n. The Main Theorem is equivalent to the assertion that Th$\langle \mathbb{R}, + \rangle$ is not a member of NTISP($2^{\varepsilon n^2}$, $2^{\varepsilon n}$) for some $\varepsilon > 0$.

We do not interpret the Main Theorem as suggesting the likelihood of an inherent time-space tradeoff among decision algorithms for Th$\langle \mathbb{R}, + \rangle$. The Theorem merely leaves open the possibility of such a tradeoff.

The Main Theorem applies to other theories such as monadic predicate calculus and exponentially bounded concatenation theory, all of which can be shown to be log-linear equivalent [17, 19] Recently Berman has observed that Th$\langle \mathbb{R}, + \rangle$ is an example of a language complete under polynomial time reduction in what is essentially the class Alt(2^n, n) of languages recognizable by alternating Turing machines using time 2^n and n alternations [1, 3, 12]. Our results imply that NTISP(2^{n^2}, 2^n) $\subset$ Alt($2^{O(n)}$, $O(n)$), an observation which we interpret as supporting the conjecture that Berman's alternating machine complexity classes properly contain the languages recognizable in nondeterministic exponential time.

2. Time-space classes

The basic computational model used is a deterministic or nondeterministic multitape Turing machine (DTM or NTM). It has a finite number of worktapes, each with a single read-write head which can move in both directions and a single input tape with a two-way read-only head. An accepting computation of a Turing machine M on input x is a computation of M which starts with the word x written on the input tape and the rest of the tapes blank, and terminates in an accepting state. The time of a computation is the number of steps in it; its space is the number of worktape squares visited during the computation (input tape squares not counted). By the linear speed-up theorem [10, pp. 137, 138], it suffices to specify time and space bounds only to within a constant factor (e.g., it is unnecessary to specify the base of a logarithm). All time and space bounds are assumed to be positive valued functions on the positve integers.

Definition 1. Let T and S be functions from the positive integers to the positive integers. Then a *(n)tisp(T, S) machine* is a (non)deterministic multitape Turing machine such that on every input x of length n, if there is an accepting computation for x, then there is an accepting computation which uses time at most $T(n)$ and space at most $S(n)$.

Remark. A NTM which runs in time $T(n)$ and space $S(n)$ is not necessarily an ntisp(T, S)-machine as Definition 1 requires that time and space bounds are achieved in a single computational path.

Definition 2. Let Σ be a finite alphabet. Then (N)TISP(T, S) is the set of languages $A \subseteq \Sigma^*$ for which there exists a (n)tisp(T, S)-machine M such that for all $x \in \Sigma^*$

(i) if $x \in A$, then there is an accepting computation of M on x,

(ii) if $x \notin A$, then there is no accepting computation of M on x.

We will show that under some familiar 'honesty' conditions [9, 15, 16] upon T and S, (N)TISP defines a hierarchy in the following sense: for small increases in the growth rate of T and S new languages can be accepted which could not be accepted before.

Definition 3 ([15]). A function S is *fully constructible* if there is a DTM M such that for each input of length n M halts in precisely space $S(n)$ with the string $\# \beta^{S(n)-2} \#$ on one of its work tapes (where β denotes the blank tape symbol).

Definition 4 ([16]). A function T with $T(n) \geqslant n$ is a *running time* if there is a DTM M such that for each input of length n, the computation of M has precisely $T(n)$ steps.

Definition 5. Two functions T and S are *compatible* if each of them is computable by a tisp(T, S)-machine.

Remark. It is easy to show that if two functions T and S are compatible and $T(n) \geqslant n$, then T is a running time and S is fully constructible.

Theorem 1 ([9]). *Let T_2 be a running time, $S_2 \geqslant \log(n)$ and let T_1 and S_1, and T_2 and S_2, be pairs of compatible functions respectively. If*

(i) $T_1(n) \log(T_1(n)) = o(T_2(n))$ *and*

(ii) $S_1(n) = o(S_2(n))$,

then

$$\text{TISP}(T_1(n), S_1(n)) \subsetneq \text{TISP}(T_2(n), S_2(n)).$$

Proof. It is a well-known result that condition (i) suffices to show that DTIME$(T_1(n))$ (i.e., the class of languages recognized by a DTM within time $T_1(n)$) is properly contained in DTIME$(T_2(n))$. Likewise condition (ii) suffices to obtain a similar result for deterministic space [10, pp. 149–155]. It is straightforward to combine these proofs to obtain the separation result for TISP. We omit the details, which can be found in [9].

Theorem 2. *Let T_2 be a running time, $S_2(n) \geqslant \log(n)$, and let T_1 and S_1, and T_2 and S_2, be pairs of compatible functions. If*

(i) $T_1(n+1) = o(T_2(n))$ *and*

(ii) $S_1(n+1) = o(S_2(n))$,

then

$$\mathrm{NTISP}(T_1(n), S_1(n)) \subsetneq \mathrm{NTISP}(T_2(n), S_2(n)).$$

Proof. Condition (i) suffices to obtain a separation result for nondeterministic time classes whereas condition (ii) is adequate to get a similar result for nondeterministic space classes [15, 16]. We merely indicate how to combine the proofs of these results—assuming familiarity with the notation of [15, 16]—to obtain a proof of Theorem 2. The conditions for the program code [16, p. 152; 15, pp. 76, 77] are the same as for the time and space theorem ([16, Theorem 4] and [15, Theorem 6]). The universal simulator first lays off $S_2(n)$ squares and then behaves like the clocked version. Only in the case when $k \geq T(|x|)$ and $\log(k) \geq S(|x|)$ does the machine M' behave like the machine M. In all other cases it behaves like U_1.

Basic for proving the Main Theorem is the notion of log-linear reducibility defined in [17, p. 43]:

Definition 6. Let $A \subseteq \Sigma^+$, $B \subseteq \Delta^+$ for some finite alphabets Σ, Δ. Then

$$A \leq_{\text{log-lin}} B \quad \text{via } f$$

iff f is a function, $f: \Sigma^+ \to \Delta^+$, such that

$$x \in A \text{ iff } f(x) \in B \quad \text{for all } x \in \Sigma^+,$$

and f can be computed by a DTM within space $\log(n)$ and f is linear bounded, i.e., there is $c \in N^+$ such that $|f(x)| \leq c|x|$ for all $x \in \Sigma^+$.

Lemma 1. *Let $A \leq_{\text{log-lin}} B$, $T(n)$ and $S(n)$ be monotone nondecreasing functions. Then there is some polynomial p and some constant $c > 0$ such that*

(i)

$$A \notin \begin{cases} \mathrm{DTIME}(T(n)+p(n)) \\ \mathrm{DSPACE}(S(n)+\log(n)) \\ \mathrm{NTIME}(T(n)+p(n)) \\ \mathrm{NSPACE}(S(n)+\log(n)) \end{cases} \Rightarrow \quad B \notin \begin{cases} \mathrm{DTIME}(T(cn)) \\ \mathrm{DSPACE}(S(cn)) \\ \mathrm{NTIME}(T(cn)) \\ \mathrm{NSPACE}(S(cn)) \end{cases}$$

(ii)

$$A \notin \begin{cases} \mathrm{TISP}(T(n)+p(n), S(n)+\log(n)) \\ \mathrm{NTISP}(T(n)+p(n), S(n)+\log(n)) \end{cases} \Rightarrow \quad B \notin \begin{cases} \mathrm{TISP}(T(cn), S(cn)) \\ \mathrm{NTISP}(T(cn), S(cn)). \end{cases}$$

For a proof of part (i) of Lemma 1 see [17, pp. 46, 47]. Part (ii) can be shown similarly.

3. The theory of real addition

Let $\mathfrak{R} = \langle \mathbb{R}, + \rangle$ be the structure consisting of the set of all real numbers with the operation of addition. Let $\mathrm{Th}(\mathfrak{R})$ be the first order theory of $\mathfrak{R}$, i.e., the set of all first order sentences true in $\mathfrak{R}$.

As a technical tool for the proof of the Main Theorem as stated in the introduction we will use the first order theory of string concatenation and what we call t-bounded concatenation theory. Meyer [8] has shown that 2^n-bounded concatenation theory is log-lin reducible to Th($\mathfrak{R}$). We will show that NTISP($2^{n^2}, 2^n$) is log-lin reducible to 2^n-bounded concatenation theory. The Main Theorem then follows immediately from Lemma 1, Theorem 2 and the transitivity of log-lin reducibility.

Definition 7. Let Σ be a finite set and let $L(\Sigma)$ be the first order language with equality, with constants $\boldsymbol{\sigma}$ for each $\sigma \in \Sigma$, and whose only atomic formulae (other than equalities) are of the form $\mathbf{cat}(x, y, z)$. The *elementary theory of concatenation*, CT(Σ), is the set of true sentences in $L(\Sigma)$ under the following interpretation: Σ^* is the underlying domain, the constant symbols denote the elements $\sigma \in \Sigma$, and for $a, b, c \in \Sigma^*$, $\mathbf{cat}(a, b, c)$ is true iff a is the concatenation of b and c.

We assume that one of the standard formats is used for writing well formed formulae in CT(Σ) which are built up with propositional connectives and quantifiers as usual. The *length* of a formula is the number of symbols in the formula where subscripts are written in binary.

By bounding the length of strings in CT(Σ) (in a sense made precise in the following definition), we obtain bounded concatenation theory.

Definition 8. Let Σ be a finite set and let $L(\Sigma)$ be the first order language with equality, with constants $\boldsymbol{\sigma}$ for each $\sigma \in \Sigma$, and whose only atomic formulae (other than equalities) are of the form $\mathbf{bcat}(x, y, z, \boldsymbol{n})$, where $\boldsymbol{n}$ is the *unary* numeral for the nonnegative integer n. Then for any function $t: N \to N$, we define *t-bounded concatenation theory* (t-BCT(Σ)) as the set of true sentences in $L(\Sigma)$ under the following interpretation: Σ^* is the underlying domain, the constant symbols denote the elements $\sigma \in \Sigma$, and for $a, b, c \in \Sigma^*$, $\mathbf{bcat}(a, b, c, \boldsymbol{n})$ is true iff a is the concatenation of b and c and the length of the string a is at most $t(n)$.

Remark. As $\boldsymbol{n}$ is written in unary, the length of the atomic formula $\mathbf{bcat}(x, y, z, \boldsymbol{n})$ is proportional to n plus the size of the variables x, y and z.

In reducing NTISP to bounded concatenation theory it is convenient to restrict the underlying computational model to be a 'simple' one-tape Turing machine (STM) [18, p. 3]. This can be done without loss of generality because an STM can simulate a multitape Turing machine with only a quadratic time loss and no space loss [10, p. 139]. Furthermore, we assume that any move which shifts the head off the left end of the tape causes the STM to halt and reject the input.

In the reduction we will describe the computation of an STM with short formulae in 2^{3n}-*BCT*(Σ). Let M be an STM, let Q denote the set of its states and S its tape alphabet. An instantaneous description (i.d.) of M can always be represented as a word in S^*QS^*. We use the convention that in every i.d. the state symbol q is positioned immediately to the left of the symbol being scanned. As in [18, p. 15] we define the function $\mathrm{Next}_M : S^*QS^* \to 2^{S^*QS^*}$, where $\mathrm{Next}_M(d)$ is the set of i.d.'s that can occur one step after i.d. d. We remark here that Next_M is length preserving. It suffices to make 'local checks' within i.d. d_1 and i.d. d_2 to decide if $d_2 \in \mathrm{Next}_M(d_1)$. The reason for this is that in one step only a few symbols around the state symbol can change.

Lemma 2 ([18, p. 15]). *Let M be an STM, $\$ \notin S \cup Q$, and $\Sigma = S \cup Q \cup \{\$\}$. There is a function $N_M : \Sigma^3 \to 2^{\Sigma^3}$ with the following properties: Let d_1 be any i.d. of M, let k be the length of d_1 and suppose*

$$\$d_1\$ = d_{10}d_{11}d_{12} \cdots d_{1k}d_{1,k+1} \quad \textit{where } d_{1j} \in \Sigma \text{ for } 0 \leq j \leq k+1$$

and

$$\$d_2\$ = d_{20}d_{21}d_{22} \cdots d_{2k}d_{2,k+1} \quad \textit{where } d_{2j} \in \Sigma \text{ for } 0 \leq j \leq k+1$$

then

$$d_2 \in \mathrm{Next}_M(d_1) \textit{ iff } d_{2,j-1}d_{2,j}d_{2,j+1} \in N_M(d_{1,j-1}d_{1,j}d_{1,j+1}) \quad \textit{for all } 1 \leq j \leq k.$$

For a proof of Lemma 2 see [17, pp. 38, 39]. Informally N_M specifies all possibilities of how the symbols of one i.d. can change in one step.

The classes 1-TISP and 1-NTISP are defined for STM's in the same way that TISP and NTISP were given in Definition 2 above for (n)tisp(T, S)-machines. Then the main lemma can be stated as follows:

Lemma 3. $(\forall A \in 1\text{-NTISP}(2^{n^2}, 2^n))(\exists \Sigma) A \leq_{\text{log-lin}} 2^{3n}\text{-BCT}(\Sigma))$.

Proof. Let M be a nondeterministic STM recognizing $A \subseteq \theta^*$ simultaneously within time 2^{n^2} and space 2^n. Let Σ be the alphabet for M given in Lemma 2. For each $x \in \theta^*$ we will describe a sentence S_x in 2^{3n}-BCT(Σ) which asserts that there is an accepting computation of M on input x. Thus $x \in A$ iff S_x is true in 2^{3n}-BCT(Σ). We will then observe that the function mapping x to S_x is computable in deterministic logspace and is linear bounded, viz., the length of S_x is at most proportional to the length of x. This will then complete the proof.

Let $n = |x|$. The computation to be described is 2^{n^2} steps long, thus a word consisting of a representation of the whole computation would be of length $2^{O(n^2)}$ and therefore too long to be expressed in the language of 2^{3n}-BCT(Σ). Instead we shall define the formula S_x in 2^{3n}-BCT(Σ) based on the construction of the formula $P_{k,n}(z)$ in CT(Σ) which, for all integers k and n and for all $z \in \Sigma^*$ is true iff

(P1) z is a string of the form $\$z_0\$z_1\$ \cdots \$z_{2^n}\$$,

(P2) z_j represents an i.d., $0 \leq j \leq 2^n$,

(P3) $|z_j| = 2^n + 1$, $0 \leq j \leq 2^n$,

(P4) in some computation of M which is started in i.d. z_j the i.d. z_{j+1} can be reached in at most 2^{kn} steps using space at most 2^n, $0 \leq j \leq 2^n$.

The formulae $P_{k,n}(z)$ will be defined inductively.

As a notational convenience we will introduce some abbreviations for formulae in concatenation theory. Let $\Delta = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, where $\sigma_i \in \Sigma$ for $1 \leq i \leq k$ and let ε denote the empty string:

Abbreviation	Formula
$p = qr$	$\mathbf{cat}(p, q, r)$
$p = qrs$	$(\exists x)(p = qx \wedge x = rs)$
$p = \varepsilon$	$p = pp$
$p = q$	$(\exists x)(p = qx \wedge x = \varepsilon)$
$p \in \Delta$	$(p = \sigma_1) \vee \cdots \vee (p = \sigma_k)$
$p \in \Delta^*$	$(\forall x, y, z)((p = xyz \wedge y \in \Sigma) \rightarrow y \in \Delta)$
$p \subset q$	$(\exists x, y)(q = xpy)$

We also define for each $k \in N$ a formula $\mathbf{ln}_k(x)$ of concatenation theory which is true iff the length of x is equal to k. We define $\mathbf{ln}_k(x)$ inductively:

- $\mathbf{ln}_1(x) := x \in \Sigma$,
- $\mathbf{ln}_{2k}(x) := (\exists y, z)(x = yz \wedge \mathbf{ln}_k(y) \wedge \mathbf{ln}_k(z))$,
- $\mathbf{ln}_{2k+1}(x) := (\exists y, z)(x = yz \wedge \mathbf{ln}_{2k}(y) \wedge \mathbf{ln}_1(z))$.

The formula $\mathbf{ln}_m$ has proportional to m connectives and variables and so its length is proportional to $m \log(m)$. However there is a standard 'abbreviation trick' [14, 5] which allows n occurrences of subformulae which are the same—except for the names of the variables—to be replaced by single occurrences of n distinct variables and one occurrence of the subformula. Applying the abbreviation trick to $\mathbf{ln}_m$ yields an equivalent formula $\mathbf{l}_m$ whose length is proportional to $\log(m)$, namely:

- $\mathbf{l}_1(x) := x \in \Sigma$,
- $\mathbf{l}_{2k}(x) := (\exists y, z)(x = yz \wedge (\forall w)((w = y \vee w = z) \rightarrow \mathbf{l}_k(w)))$,
- $\mathbf{l}_{2k+1}(x) := (\exists y, z)(x = yz \wedge \mathbf{l}_{2k}(y) \wedge \mathbf{l}_1(z))$.

By this definition, the formula $\mathbf{l}_m$ has proportional to $\log(m)$ connectives and occurrences of variables. But the new variables introduced in constructing $\mathbf{l}_{2k}$ from $\mathbf{l}_k$ need only be distinct from each other and from the free variables of $\mathbf{l}_k$. Thus only a constant number of different variables is needed to construct $\mathbf{l}_m$, and its length is therefore proportional to $\log(m)$ as claimed.

The formula Form(z) will assert conditions (P1)–(P3) above:

$$\text{Form}(z) := (\exists w)(z = \$w\$) \wedge \mathbf{l}_{(2^n+1)(2^n+2)+1}(z) \wedge (\forall z_1)\{(\$z_1\$ \subset z \wedge \$ \not\subset z_1) \rightarrow [\mathbf{l}_{2^n+1}(z_1) \wedge (\exists w_1, w_2, q)(w_1, w_2 \in S^* \wedge q \in Q \wedge w_2 \neq \varepsilon \wedge z_1 = w_1 q w_2)]\}. \quad (1)$$

As the induction base we will construct the formula $P_{0,n}(z)$ which satisfies the conditions (P1)–(P3) above and the conditions that each of the successive i.d.'s are either identical or follow in one step:

$$P_{0,n}(z) := \text{Form}(z) \wedge (\forall z_1, z_2)\{(\$z_1\$z_2\$ \subset z \wedge \$ \not\subset z_1 \wedge \$ \not\subset z_2) \rightarrow [(z_1 = z_2) \vee (\exists w_1, s_1, q, s_2, w_2, u)(s_1, s_2 \in S \cup \{\$\} \wedge q \in Q \wedge \$z_1\$ = w_1 s_1 q s_2 w_2 \wedge \$z_2\$ = w_1 u w_2 \wedge u \in N_M(s_1 q s_2))]\}. \quad (2)$$

For the induction step we will write a formula $P_{k+1,n}(z)$ using $P_{k,n}(z)$ as a subformula. The basic idea is that i.d. z_{j+1} can be reached in $2^{(k+1)n}$ steps from i.d. z_j iff there is a string w which has z_j as a prefix, z_{j+1} as a suffix and for which $P_{k,n}(w)$ holds. Thus $P_{k+1,n}(z)$ can be written as:

$$P_{k+1,n}(z) := \text{Form}(z) \wedge (\forall z_1, z_2)[(\$z_1\$z_2\$ \subset z \wedge \$ \not\subset z_1 \wedge \$ \not\subset z_2)$$

$$\rightarrow (\exists w_1)(P_{k,n}(\$z_1\$w_1\$z_2\$))]. \qquad (3)$$

This completes the inductive construction of $P_{k,n}(z)$.

We remark here that the length of $P_{k+1,n}$ is equal to a constant plus the length of $P_{k,n}$ and the length of the formula Form. The formula Form is of length $O(n)$. Hence $P_{n,n}$ is of length $O(n^2)$ primarily because of the n occurrences of Form. To apply the standard abbreviation trick to $P_{n,n}$ n distinct variables are needed which forces the subscript of each of them to be of size $\log(n)$. Thus an equivalent formula $P'_{n,n}$ obtained in such a way would be of length $O(n \log(n))$.

We wish now to construct a short formula $b\text{-}P_{n,n}(z)$ in the language of 2^{3n}-BCT(Σ) which is true iff conditions (P1)–(P4) as above hold. The straightforward way to obtain such a $b\text{-}P_{n,n}$ is to first rewrite $P'_{n,n}$ so that the formula **bcat** replaces each occurrence of **cat**. Since there are only proportional to n occurrences of **cat** in $P'_{n,n}$ and the length of **bcat** is $O(n)$, one could next apply the standard abbreviation trick on the multiple occurrences of **bcat** to obtain a formula $b\text{-}P_{n,n}$ which is also of length $O(n \log(n))$. This would be enough to prove a version of our Main Theorem with $2^{O(n^2/\log^2(n))}$ instead of $2^{O(n^2)}$ and $2^{O(n/\log(n))}$ instead of $2^{O(n)}$. In the paragraphs below we will give a slightly more complicated construction yielding a formula $b\text{-}P_{n,n}$ which is actually of length $O(n)$. The straightforward construction of $b\text{-}P_{n,n}$ just suggested above, suffered because no effort was made to economize on the number of occurrences of the formula **bcat**. Only in the final step each occurrence of **bcat** was replaced by a distinct variable. It will turn out to be more efficient to define $b\text{-}P_{n,n}$, **bcat** and also the auxiliary formula $\mathbf{l}_m$ by simultaneous induction. In particular the idea of the construction is as follows: the formula $S_{k,n}(a, b, c, d, e, z)$ in the language of 2^{3n}-BCT(Σ) will mean the same as

$$\mathbf{bcat}(a, b, c, \boldsymbol{n}) \wedge \mathbf{l}_{2^n+1}(d) \wedge \mathbf{l}_{(2^n+1)(2^n+2)+1}(e) \wedge P_{k,n}(z). \qquad (4)$$

Thus the formula $S_{k+1,n}(a, b, c, d, e, z)$ will be equivalent to

$$(\exists u)S_{k,n}(a, b, c, d, e, u) \wedge P_{k+1,n}(z).$$

Now note that as in (3), $P_{k+1,n}(z)$ is equivalent to

$$\text{Form}(z) \wedge (\forall z_1, z_2)[(\$z_1\$z_2\$ \subset z \wedge \$ \not\subset z_1 \wedge \$ \not\subset z_2)$$

$$\rightarrow (\exists w_1, f, g)(S_{k,n}(\varepsilon, \varepsilon, \varepsilon, f, g, \$z_1\$w_1\$z_2\$))]. \qquad (5)$$

Similarly the formula Form(z) as in (1) is equivalent to

$$(\exists w)(z = \$w\$) \wedge (\exists f, u)S_{k,n}(\varepsilon, \varepsilon, \varepsilon, f, z, u) \wedge (\forall z_1)\{(\$z_1\$ \subset z \wedge \$ \not\subset z_1)$$
$$\rightarrow [(\exists g, u)S_{k,n}(\varepsilon, \varepsilon, \varepsilon, z_1, g, u)$$
$$\wedge (\forall w_1, w_2, q)(w_1, w_2 \in S^* \wedge q \in Q \wedge w_2 \neq \varepsilon \wedge z_1 = w_1qw_2)]\} \qquad (6)$$

We observe now that the meaning of formulae (5) and (6) does not change if each occurrence of the formula **cat** is replaced by the formula **bcat** as the length of all strings in (5) and (6) is bounded by 2^{3^n}. Thus (5) and (6) can equivalently be written by replacing each occurrence of **cat**(p, q, r) by $(\exists f, g, u)S_{k,n}(p, q, r, f, g, u)$. So we can conclude that a formula $S'_{k+1,n}$ equivalent to $S_{k+1,n}$ can be written using a fixed number (independent of k and n) of copies of $S_{k,n}$ plus a fixed number of additional quantifiers, variables and logical connectives. Applying the abbreviation trick to $S'_{k+1,n}$ yields the desired formula $S_{k+1,n}$ which has only one copy of $S_{k,n}$ as a subformula and a fixed number of quantifiers, variables and logical connectives. Again we note that no difficulty arises if the new variables introduced in constructing $S_{k+1,n}$ coincide with variables bound inside $S_{k,n}$. Thus only a constant number of additional variables are needed to construct $S_{k+1,n}$ from $S_{0,n}$. Therefore the length of $S_{k+1,n}$ is $O(k+1)$ plus the length of $S_{0,n}$.

It remains only to indicate how to construct a formula $S_{0,n}(a, b, c, d, e, z)$. Using the same techniques as illustrated previously, a formula of size $O(m + \log(m))$ equivalent to $\mathbf{l}_m$ can be defined in 2^{3^n}-BCT(Σ). Since $S_{0,n}$ can, according to (1), (2) and (4), be defined using only a fixed number of occurrences of formulae equivalent to $\mathbf{l}_m$ or **bcat**(a, b, c, n), it follows that the length of $S_{0,n}$ can be kept proportional to n.

To complete the construction of S_x we also need a formula $\mathrm{IN}_{x,n}(w)$ which is true iff w is the string x. Again with care in reusing bound variables, we can define $\mathrm{IN}_{x,n}(w)$ such that the length of this formula is proportional to n.

Finally let S_x be the following formula, where q_0 denotes the initial state, q_a the accepting state and β the blank tape symbol.

$$S_x := (\exists w, b, z, u)[\mathrm{IN}_{x,n}(w) \wedge b \in \{\beta\}^*$$
$$\wedge \$q_0wb\$q_au\$ \subset z \wedge S_{n,n}(\varepsilon, \varepsilon, \varepsilon, q_au, z, z)].$$

Clearly $x \in A$ iff S_x is (true) in 2^{3^n}-BCT(Σ). We have already shown that the function mapping x to S_x is linear bounded. The results of [19, 13] may be used to show that the computation of S_x can be carried out within deterministic logspace; we leave the verification of this final claim to the reader. Hence the transformation of x to S_x implies that $A \leq_{\text{log-lin}} 2^{3^n}$-BCT($\Sigma$).

Remark. For any $c > 1$ and any alphabet Σ there exists an alphabet θ such that 2^{cn}-BCT(Σ) is log-lin reducible to 2^n-BCT(θ).

Lemma 3 and the preceding remark, together with the reduction of 2^n-BCT(Σ) to Th$\langle\mathbb{R}, +\rangle$ completes the proof of the Main Theorem.

4. Open problems

In this paper we classified logical theories with respect to both computation time and space. The basic open question remaining is to characterize the complexity of Th⟨ℝ, +⟩ (or equivalently Alt(2^n, n)) more precisely in terms of time and space. Note that the claims that Alt(2^n, n) is equivalent to NTIME(2^n) or equivalent to SPACE(2^n), or both for that matter, remain consistent with our Main Theorem.

A second related open problem is to improve the known lower bounds on the complexity of Presburger Arithmetic. Such improvements do not follow directly by the same method used to bound Th⟨ℝ, +⟩, as can easily be seen by parameterizing our main result. We have shown that for $f(n)=2^n$, the class NTISP($f(n)^n, f(n)$) reduces to Th⟨ℝ, +⟩. The same proof shows only that NTISP($g(n)^n$, $g(n)$) reduces to Presburger Arithmetic where $g(n)=2^{2^n}$, a result which degenerates to the known results [6] that NTIME(2^{2^n}) reduces to Presburger Arithmetic.

Acknowledgment

We would like to thank Jeanne Ferrante, Karl Winklmann and the referees for their careful reading and helpful comments concerning this paper.

References

[1] L. Berman, Precise bounds for Presburger Arithmetic and the reals with addition: Preliminary report, in: *Proc. 18th Annual Symposium on Foundations of Computer Science* (1977) 95–99.
[2] L. Berman, The complexity of logical theories, *Theoret. Comput. Sci.* **11** (1980) 71–77, this journal.
[3] A.K. Chandra and L.J. Stockmeyer, Alternation, in: *Proc. 17th Annual Symposium on Foundations of Computer Science* (1976) 98–108.
[4] J. Ferrante and C.W. Rackoff, A decision procedure for the first order theory of real addition with order, *SIAM J. Comput.* 4(1) (1975) 69–76.
[5] J. Ferrante and C.W. Rackoff, *The Computational Complexity of Logical Theories*, Lecture Notes in Mathematics (Springer, Berlin, 1979).
[6] M.J. Fischer and M.O. Rabin, Super-exponential complexity of Presburger Arithmetic, *SIAM-AMS Proc.* **VII** (AMS, Providence, RI, 1974).
[7] K. Fleischmann, B. Mahr and D. Siefkes, Bounded concatenation theory as a uniform method for proving lower complexity bounds, in: R.O. Gandy and J.M.E. Hyland, Eds, *Logic Colloquium 76* (North-Holland, Amsterdam, 1977) 471–490.
[8] K. Fleischmann, B. Mahr and D. Siefkes, Complexity of decision problems, Technische Universität Berlin, Bericht Nr.76-09 (1976).
[9] E.P. Glinert, On restricting Turing computability, *Math. Systems Theory* **5**(4) (1971) 331–343.
[10] J.E. Hopcroft and J.D. Ullman, *Formal Languages and their Relation to Automata* (Addison-Wesley, Reading, MA, 1969) 137–140, 149–155.
[11] D.E. Knuth, Big omicron and big omega and big theta, *Sigact News* **8** (2) (1976) 19.
[12] D. Kozen, On parallelism in Turing machines, in: *Proc. 17th Annual Symposium on Foundations of Computer Science* (1976) 89–97.
[13] J.C. Lind, Computing in logarithmic space, MIT Project MAC TM 52 (1974).

[14] C.W. Rackoff, The computational complexity of some logical theories, MIT Project MAC TR 144 (1975) 118–127.

[15] J.I. Seiferas, Techniques for separating space complexity classes, *J. Comput. System Sci.* **14**(1) (1977) 73–99.

[16] J.I. Seiferas, M.J. Fischer and A.R. Meyer, Separating nondeterministic time complexity classes, *J. ACM* **25**(1) (1978) 146–167.

[17] L.J. Stockmeyer, The complexity of decision problems in automata theory and logic, MIT Project MAC TR 133 (1974).

[18] L.J. Stockmeyer, The polynomial time hierarchy, *Theoret. Computer Sci.* 3 (1977) 1–22.

[19] L.J. Stockmeyer and A.R. Meyer, Word problems requiring exponential time: Preliminary report, in: *Proc. 5th Annual ACM Symposium on Theory of Computing* (1973) 1–9.