Theoretical Computer Science 11 (1980) 71-77

THE COMPLEXITY OF LOGICAL THEORIES

Leonard BERMAN
Computer Sciences, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.

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

Abstract. In this paper we introduce a method of encoding the computation of an alternating TM into a logical theory. The efficiency of the embedding we give together with the decision procedures, using Ehrenfencht games, which have been developed over the past few years, yield precise lower bounds for many decidable theories. In this paper we apply our technique explicitly to the theory of reals with addition; however, it should be clear that the techniques apply directly to other theories as well.

We also outline the proof of a general theorem, motivated by a comment of A. R. Meyer and discovered independently by A. R. Meyer and L. Stockmeyer, which allows us to obtain a recent result of Bruss and Meyer directly from our precise characterization of RA.

1. Introduction

Over the past few years there has been a large amount of work on the problem of establishing the precise resource requirements of decidable mathematical theories. Two classical theories which have received considerable attention are Presburger Arithmetic (PA), i.e. the theory of natural numbers with addition, $<$, and 0; and the theory of reals with addition $<$, and 0 (RA).

Up until now, these theories have resisted efforts to establish their precise resource requirements. The lower bounds on the complexity of these theories are due to the work of Fischer and Rabin [6]. The upper bounds have undergone a series of improvements, culminating with the work of Ferrante and Rackoff [5]. In both PA and RA the differences between the upper and lower bounds have been essentially the difference between time and tape. The main result of this paper is a precise characterization of the complexity of these two theories. These results suggest that these theories (and other complex theories as well) may not have precise complexity characterizations in terms of the usual time and tape measures.

Logicians have long been aware of the importance of the minimum quantifier depth in the definition of a set as a measure of the complexity (in their case the degree of undecidability) of the set. Stockmeyer and Meyer [9] have observed that, at least formally, there is a similar hierarchy of sets expressible by various numbers of polynomial bounded quantifiers in front of a polynomial time predicate.

In this paper we define a complexity measure of three parameters which enables us to generalize the polynomial time hierarchy. We find that in this measure PA and RA are both complete for a 'diagonal class'. This measure is closely related to the notion of alternation in the work of Chandra, Kozen and Stockmeyer [4], and by using their results we are able to see how these 'diagonal classes' relate to the standard time and tape classes. We see that these new classes probably do not correspond to any of the well known time or tape classes.

2. Measure

In this section we introduce a new measure on the complexity of a computation by a single-tape alternating TM. Our measure takes into account the three components of the complexity of such a computation: space, time, and alternation depth.

Since alternation is a new concept we will give a brief, informal definition of an alternating machine, for precise definitions see [4].

The computation of an alternating machine produces a binary computation tree just as a non-deterministic machine does. Associated with an alternating machine there is a function $g:\text{states}\to\{\wedge, \vee\}$. This function labels the computation tree and allows us to define acceptance inductively on the height of the finite subtree $T_{M,x}$ defined here.

$T_{M,x}$ is any minimal, finite subtree of the full computation tree (if one exists) with the following properties:

(1) $T_{M,x}$ contains the root,

(2) if one son of a node of the computation tree is in $T_{M,x}$, then all sons of that node are also,

(3) the root of $T_{M,x}$ is an accepting node when acceptance is defined by the algorithm which follows.

If the tree is of height 0, i.e. a single node, the node is an accepting node iff the state associated with the node is an accepting state. If the tree is of height $n+1$, and the state of the root node is q, there are two cases:

(1) if $g(q)=\vee$, then the root node is an accepting node iff either of its sons are accepting nodes,

(2) if $g(q)=\wedge$, then the root node is an accepting node iff both of its sons are accepting nodes.

A machine M_i accepts an input x iff a subtree $T_{M,x}$, as defined above, exists.

The alternation depth of the computation tree of M on x is defined as follows (where $T_{M,x}$ is as above):

$$\text{Alt-depth}\,(T_{M,x})=\max\{\#(p):p \text{ is a path from root to a leaf of } T_{M,x}\},$$

$$\#(p)=|\{i : \text{ID}_i \text{ is on path } p,\ \text{ID}_{i+1} \text{ follows ID}_i \text{ on path } p,$$

$$g(\text{state}(\text{ID}_i))\neq g(\text{state}(\text{ID}_{i+1}))\}|.$$

The time of the alternating computation of M on x is the height of $T_{M,x}$. The space of the alternating computation of M on x is the maximum size of an ID in $T_{M,x}$. Note time and space are not identical to the usual notations for non-deterministic machines.

We now define the combined space, time, alternation measure STA(· , · , ·). We say that a set A is in the class STA$(s(n), t(n), a(n))$ if there is single-tape, alternating TM, M_i, which accepts A and has the following properties:

(1) $\max\{\text{space}(T_{M_i,x}) \mid \text{length}(x)=n\} \leqslant s(n)$,

(2) $\max\{\text{time}(T_{M_i,x}) \mid \text{length}(x)=n\} \leqslant t(n)$,

(3) $\max\{\text{Alt-depth}(T_{M_i,x}) \mid \text{length}(x)=n\} < a(n)$.

Where, if $a(n)=0$, we interpret condition (3) to be satisfied iff $T_{M_i,x}$ has no nodes of degree 2, i.e. M_i is a deterministic machine.

Although in our work we shall always refer to the roots of computation trees as if they were labeled with an $\vee$, we do not, in general, require this to be the case. This guarantees that appropriately chosen STA complexity classes are closed under complement. It is, of course, possible to consider classes where we require $g(q_{\text{init}})=\vee$.

We use a '$*$' to indicate no limit in the given coordinate. E.g.,

$$\text{STA}(n,*,1)=\bigcup_{c>1}\text{STA}(n,c^n,1).$$

It should be clear that we can alter machines so that they have the following properties:

(1) each machine has two accepting states $q_\vee$, $q_\wedge$ such that $g(q_\vee)\neq g(q_\wedge)$,

(2) each machine has an ε-transition $q_\vee \to q_\wedge$ and one $q_\wedge \to q_\vee$.

We should also point out that the classes defined by our measure, relate to the usual time and space classes in the following straightforward manner:

$$P=\bigcup_{k\geqslant 1}\text{STA}(*,n^k,0),$$

$$\text{PSPACE}=\bigcup_{k>1}\text{STA}(*,n^k,n^k),$$

$$\bigcup_{k\geqslant 1}\text{DSPACE}(T(n)^k)=\bigcup_{k\geqslant 1}\text{STA}(*,T(n)^k,T(n)^k),$$

$$\bigcup_{k\geqslant 1}\text{DTIME}(T(n)^k)=\bigcup_{k\geqslant 1}\text{STA}(*,T(n)^k,0).$$

3. Lower bound

In this section we define the embedding

$$\text{STA}(*,2^n,n)\to \text{RA}.$$

In the embedding we further restrict our machines so that every ID with state(ID) $\neq$ accepting state has the property that there exists an ID$'$ for which:

(1) ID $\to$ ID$'$ in less than 2^n steps, and

(2) $g(\text{state(ID)}) \neq g(\text{state(ID}'))$.

By choosing the machines to satisfy this property, we do not alter the set which we can embed in RA. The machine, M, we refer to is an alternating TM which runs in time 2^n, n alternations, and satisfies the restrictions above. In order to do this we must define a predicate $P_i(x, y, z, w)$ with two properties:

(1) $P_i(x, y, 0, 0)$ iff x and y are 2^n bit integers which encode ID's of a computation of M, there is a sequence $s_1, \ldots, s_j$ of ID's of M such that $s_1 = x$, $s_j = y$, $j < 2^n$, $\forall m$ $|s_m| < 2^n$, $\forall m$ ($s_m \to s_{m+1}$ by one step of M), $\forall m < j$ $g(s_m) = g(s_1)$, and $g(s_j) \neq g(s_1)$,

(2) $P_i(\cdot, \cdot, z, \{^0_1\})$ iff z is a 2^n bit integer which encodes the root of an accepting computation tree of alternation depth less than or equal to i, and $g(\text{state}(z)) = \{^\vee_\wedge\}$.

We will define P_n inductively. P_0 is constructed entirely through the methods of Fischer and Rabin [6]. We construct a predicate $S_n(x, y)$ which is satisfied only if x and y satisfy the first property required by P_i. We can then define P_0 by

$$\begin{aligned} P_0(x, y, z, w) \equiv {} & [z = 0 \wedge w = 0 \wedge S_n(x, y)] \\ & \vee [z \neq 0 \wedge w = 0 \wedge S_n(z, \text{accept}_\vee)] \\ & \vee [z \neq 0 \wedge w = 1 \wedge \forall u\, [\neg S_n(z, u) \vee u = \text{accept}_\wedge]]. \end{aligned}$$

Since our machines are modified to have a unique tape configuration when they enter an accepting state this predicate is easy to construct. For the details of how to construct $S_n(x, y)$ see [6], for an outline of the construction see [1].

We now define P_i by

$$\begin{aligned} P_i(x, y, z, w) \equiv {} & [z = 0 \wedge w = 0 \wedge P_{i-1}(x, y, 0, 0)] \\ & \vee [z \neq 0 \wedge w = 0 \wedge \exists u_1\, [P_{i-1}(z, u_1, 0, 0) \wedge P_{i-1}(0, 0, u_1, 1)]] \\ & \vee [z \neq 0 \wedge w = 1 \\ & \quad \wedge \forall u_2\, [\neg P_{i-1}(z, u_2, 0, 0) \vee P_{i-1}(0, 0, u_2, 0)]]. \end{aligned}$$

From the definition of the time used by an alternating machine we see that if a machine runs in STA($*$, 2^n, n), then although the machine altered to include ε-moves from the accepting states will no longer run in time 2^n the predicate P_n which we construct from it will make $P_n(0, 0, \text{start config. of } M \text{ on } x, 0)$ valid if and only if the original machine accepts x.

P_i as written grows in size as 5^i; however, by using the abbreviation trick of Fischer, Meyer and Stockmeyer we can write it in the following equivalent forms:

First put in prenex normal form

$$\begin{aligned} \exists u_1\, \forall u_2 \{ & [z = 0 \wedge w = 0 \wedge P_{i-1}(x, y, 0, 0)] \\ & \vee [w = 0 \wedge P_{i-1}(z, u_1, 0, 0) \wedge P_{i-1}(0, 0, u_1, 1)] \\ & \vee [w = 1 \wedge (\neg P_{i-1}(z, u_2, 0, 0) \vee P_{i-1}(0, 0, u_2, 0))] \} \end{aligned}$$

and then using the abbreviation trick [8, p. 189–190] write it equivalently as:

$$\begin{aligned}
&\exists u_1 \forall u_2 \exists y_1 \cdots y_5\{(z=0 \wedge w=0 \wedge y_1=1)\\
&\quad \vee (w=0 \wedge y_2=1 \wedge y_3=1) \vee (w=1 \wedge (y_4 \neq 1 \vee y_5=1))\\
&\quad \vee \forall d_1 \cdots d_4 \forall y\, [[(d_1=x \wedge d_2=y \wedge d_3=0 \wedge d_4=0 \wedge y=y_1)\\
&\qquad \vee (d_1=z \wedge d_2=u_1 \wedge d_3=0 \wedge d_4=0 \wedge y=y_2)\\
&\qquad \vee (d_1=0 \wedge d_2=0 \wedge d_3=u_1 \wedge d_4=1 \wedge y=y_3)\\
&\qquad \vee (d_1=z \wedge d_2=u_2 \wedge d_3=0 \wedge d_4=0 \wedge y=y_4)\\
&\qquad \vee (d_1=0 \wedge d_2=0 \wedge d_3=u_2 \wedge d_4=0 \wedge y=y_5)]\\
&\qquad \Rightarrow (y=1) \Leftrightarrow P_{i-1}(d_1, d_2, d_3, d_4)]\}.
\end{aligned}$$

When expanding P_{i-1} we must use new variables for $d_1, \ldots, d_4$; however, in expanding P_{i-2} we are able to reuse $d_1, \ldots, d_4$. This allows us to use merely a bounded number of variables and so we see that

$$\text{size}(P_n) \leq cn + \text{size}(S_n) \leq c_1 n.$$

This completes the embedding of STA($*$, 2^n, n) into RA.

In the construction and size analysis of the formula P_n only two features of RA were used. First, we used the fact that a formula S_n, of linear size could be constructed to satisfy our requirements in defining P_0. Second, we used the fact that a single variable could hold the description of an ID of the computation.

Other theories satisfy these criterion. For example, Presburger arithmetic satisfies both requirements and an identical construction shows

$$\text{STA}(*, 2^{2^{cn}}, n) \rightarrow \text{Presburger arithmetic}$$

via maps computable in polynomial time which increase lengths by a constant factor.

We feel that this result is basically a characterization of the power of unrestricted quantification and we suggest that analogous results will be found to hold for other theories.

The main result of this section may be restated:

Theorem 1. *For every set, A, in* STA($*$, 2^n, n) *there is a map f_A, computable in polynomial time, and a constant c_A such that*

$$A \rightarrow \text{RA} \quad \textit{via } f_A \textit{ and } \forall x\, (|f_A(x)| < c_A \cdot |x|).$$

4. Upper bounds

Ferrante and Rackoff [5] have shown that it is possible to determine the truth of a sentence in RA by merely checking the sentence and limiting the variables to range over particular subsets of the rational numbers whose numerators and denominators are integers of less than 2^{cn} bits for some c. This result implies $\mathrm{RA} \in \mathrm{STA}(*, 2^{cn}, n)$ for some c.

5. Limited alternation

It has been observed by the author and independently by Meyer and Stockmeyer [9] that there is a c such that

$$\mathrm{STA}(2^n, 2^{n^2}, 1) \subseteq \mathrm{STA}(2^{cn}, 2^{cn}, cn) \tag{1}$$

and also

$$\mathrm{STA}(2^n, 2^{n^2}, n) \subseteq \mathrm{STA}(2^{cn}, 2^{cn}, cn). \tag{2}$$

We may see that (1) holds by observing that a computation of a machine in $\mathrm{STA}(2^n, 2^{n^2}, 1)$ may be represented as the frontier of a tree of height n which has branching of degree 2^n at every level. Such a tree can also be thought of as the computation tree of a machine in $\mathrm{STA}(2^{cn}, 2^{cn}, cn)$. We can obtain a proof of Bruss and Meyer's result through a formalization of this argument.

(2) follows from (1) and the methods of Section 3.

We have obtained hierarchy results for this measure [2] and these results together with observation (1) above yield an extension of Bruss and Meyer's result [3] to:

Theorem 2. *There is a $c > 1$ such that any alternating algorithm for* RA *uses either*:

(1) *c^n space,*

(2) *c^{n^2} time,*

(3) *$O(n)$ alternations.*

Bruss and Meyer had previously obtained the above result for non-deterministic algorithms.

Our second observation above yields an interesting result:

Theorem 3. $\bigcup_{c>1} \mathrm{STA}(2^{cn}, 2^{cn^2}, cn) = \bigcup_{c>1} \mathrm{STA}(2^{cn}, 2^{cn}, cn)$.

This is in contrast to the fact that

$$\bigcup_{c>1} \mathrm{DTIME}(2^{cn^2}) \supsetneq \bigcup_{c>1} \mathrm{DTIME}(2^{cn}).$$

One might hope that an increase in any coordinate of the STA measure would yield a new class, the above suggests that this may not be true.

Conclusion

We have introduced a new method for obtaining precise bounds on theories. This method relates the power of quantification in arbitrary theories to the power of alternation in a TM computation.

The power of the method suggests that it may be worthwhile to study the STA measure introduced here in more detail.

References

[1] L. Berman, Precise bounds for Presburger Arithmetic and the reals with addition: Preliminary report, in: *Proc. 18th Annual IEEE Foundations of Computer Science Conference* (1977) 95–99.

[2] L. Berman, Complexity of QBF, to appear.

[3] A. Bruss and A. Meyer, On time-space classes and their relation to the theory of real addition, *Theoret. Comput. Sci.* **11** (1980) 59–69, this journal.

[4] A. Chandra, D. Kozen and L. Stockmeyer, Alternation, *J.ACM*, to appear.

[5] J. Ferrante and C. Rackoff, A decision procedure for the first order theory of real addition with order, *SIAM J. Comput.* **4** (1) (1975) 69–76.

[6] M.J. Fischer and M.O. Rabin, Super exponential complexity of Presburger Arithmetic, in: *SIAM-AMS Proc.* **VII** (AMS, Providence, RI, 1974).

[7] D.C. Oppen, A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger Arithmetic, *J. Comput. System Sci.* **16** (3) (1978) 323–332.

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

[9] L. Stockmeyer and A. Meyer, Word problems requiring exponential time: Preliminary report, in: *Proc. 5th SIGACT* (1973) 1–9.