N 73- 12181

# COMPARING COMPLEXITY CLASSES OF FORMAL LANGUAGES[†]

by

Ronald V. Book

19-72

Center for Research in Computing Technology

Harvard University
Cambridge, Massachusetts 02138

# COMPARING COMPLEXITY CLASSES OF FORMAL LANGUAGES[†]

by

Ronald V. Book

19-72

Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts 02138

[†]This research was supported in part by the National Aeronautics and Space Administration under Grant NGR-22-007-176 and by the National Science Foundation under Grant NSF-GJ-30409.

## Abstract

A property of polynomial complete languages is extended in order to better compare various classes of formal languages. We consider pairs $(\mathscr{L}_1, \mathscr{L}_2)$ of classes of languages such that there is a language $L \in \mathscr{L}_1$ with the property that $L \in \mathscr{L}_2$ if and only if $\mathscr{L}_1 \subseteq \mathscr{L}_2$.

Certain longstanding open questions in automata-based complexity have resurfaced recently due to the work on efficient reducibilities among combinatorial problems [5,8]. In particular, questions regarding time-tape tradeoffs and the deterministic simulation of nondeterministic machines have received renewed attention. The purpose of this paper is to extend a technique used in [5,8] in order to better compare various classes of formal languages defined by time- or tape-bounded Turing machines.

Let P (NP) be the class of languages accepted by deterministic (nondeterministic) Turing machines which operate in polynomial time. A language $L$ is polynomial complete if $L \in NP$ and if the question of the satisfiability of a statement in conjunctive normal form is polynomially reducible to $L$ [8]. It is shown in [8] that for any polynomial complete language $L$, $L \in P$ if and only if $P = NP$. What is shown here is that for many pairs $(\mathscr{L}_1, \mathscr{L}_2)$ of classes of formal languages, there exist languages $L \in \mathscr{L}_1$ such that $L \in \mathscr{L}_2$ if and only if $\mathscr{L}_1 \subseteq \mathscr{L}_2$.

Let T (NT) be the class of languages accepted by deterministic (nondeterministic) Turing machines which operate in polynomial space. By results in [9], $NT = T$ so we refer to this class simply as T. Let DLBA (CS) be the class of languages accepted by deterministic (non-deterministic) linear bounded automata. Let DEXP (NEXP) be the class of languages accepted by deterministic (nondeterministic) Turing machines which operate in exponential time, i.e., time bound $k^n$ for any $k > 1$. (In each case the bound is a function on the length of the input.)

In [4] DEXP was characterized as the class of languages accepted by deterministic or nondeterministic auxiliary pushdown machines which operate in space bound $f(n) = n$. In [7] NEXP was characterized as the class of spectra of formulae of first-order logic (the spectrum of a formula is the set of cardinalities of its models).

<u>Theorem</u>.

A. There exists a language $L \in$ NEXP such that:

(i) $L \in$ DLBA if and only if NEXP $\subseteq$ DLBA so that NEXP = DEXP = CS = DLBA and NP $\subsetneq$ DLBA;

(ii) $L \in$ CS if and only if NEXP $\subseteq$ CS so that NEXP = DEXP = CS and NP $\subsetneq$ CS;

(iii) $L \in$ T if and only if NEXP $\subseteq$ T so that NEXP $\subsetneq$ T;

(iv) $L \in$ DEXP if and only if NEXP $\subseteq$ DEXP so that NEXP = DEXP and NP $\subsetneq$ DEXP.

B. There is a language $L \in$ DEXP such that:

(v) $L \in$ DLBA if and only if DEXP $\subseteq$ DLBA so that DEXP = CS = DLBA and P $\subsetneq$ DLBA;

(vi) $L \in$ CS if and only if DEXP $\subseteq$ CS so that DEXP = CS and P $\subsetneq$ CS;

(vii) $L \in$ T if and only if DEXP $\subseteq$ T so that DEXP $\subsetneq$ T;

(viii) $L \in$ NP if and only if DEXP $\subseteq$ NP so that DEXP $\subsetneq$ NP $\subsetneq$ NEXP, NP = T, and NP is equal to the class of languages accepted by deterministic Turing machines which operate in time bound $2^{n^k}$ (for all $k \geq 1$).

C. There exists a language $L \in CS$ such that:

(ix) $L \in NP$ if and only if $CS \subseteq NP$ so that $NP = T$ and $CS \subsetneq NP$;

(x) $L \in DLBA$ if and only if $CS \subseteq DLBA$ so that $CS = DLBA$.

D. There exists a language $L \in DLBA$ such that:

(xi) $L \in NP$ if and only if $DLBA \subseteq NP$ so that $NP = T$ and $CS \subsetneq NP$.

There is a simple idea behind the proof of the theorem. Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two families of languages. Let $\mathscr{C}$ be some collection of operations on languages. Suppose $\mathscr{L}_1$ and $\mathscr{L}_2$ are both closed under the operations in $\mathscr{C}$. Further, suppose that $L \in \mathscr{L}_1$ is a language such that $\mathscr{L}_1$ is the smallest family of languages which contains $L$ and is closed under the operations in $\mathscr{C}$. Then $L \in \mathscr{L}_2$ if and only if $\mathscr{L}_1 \subseteq \mathscr{L}_2$.

In this case all of the classes NP, DEXP, NEXP, DLBA, CS, and T are abstract families of languages (AFLs), so we choose $\mathscr{C}$ to be the collection of defining operations for AFLs: union, concatenation, Kleene +, intersection with regular sets, inverse homomorphism, and nonerasing homomorphism. An AFL $\mathscr{L}$ is principal if there is a language $L \in \mathscr{L}$ such that $\mathscr{L}$ is the smallest AFL containing $L$ [6]; in this case $L$ is a principal generator of $\mathscr{L}$. In [1] it is shown that DEXP and NEXP are principal AFLs; in [10] it is shown that DLBA and CS are principal AFLs; in [3] it is shown that NP is an AFL which is not principal, and the same type of argument shows that T is an AFL which is not principal.

To prove part A., let L be a principal generator for NEXP. By definition, DLBA $\subseteq$ CS, DEXP $\subseteq$ NEXP, and NP $\subseteq$ NEXP. In [4] it is shown that CS $\subseteq$ DEXP. In [3] it is shown that NP $\neq$ DLBA, NP $\neq$ CS, NP $\neq$ NEXP, and NP $\neq$ DEXP. Since NEXP is a principal AFL and T is not, NEXP $\neq$ T. Thus (i)-(iv) hold.

To prove part B., let L be a principal generator of DEXP. As noted above, DLBA $\subseteq$ CS $\subseteq$ DEXP. By definition, P $\subseteq$ DEXP. In [3] it is shown that P $\neq$ DLBA and P $\neq$ CS. Since DEXP is a principal AFL and T is not, DEXP $\neq$ T. Thus (v)-(vii) hold. Since DEXP and NEXP are both principal AFLs and NP is not, DEXP $\neq$ NP and NEXP $\neq$ NP. By definition, NP $\subseteq$ NEXP. In [3] it is shown that if DEXP $\subseteq$ NP, then NP = T and NP is equal to the class of languages accepted by deterministic Turing machines which operate in time bound $2^{n^k}$ (for $k \geq 1$). Thus (viii) holds.

To prove part C., let L be a principal generator for CS. In [3] it is shown that CS $\subseteq$ NP implies NP = T, and that CS $\neq$ NP. By definition, DLBA $\subseteq$ CS. Thus (ix) and (x) hold.

To prove part D., let L be a principal generator for DLBA. In [3] it is shown that DLBA $\subseteq$ NP implies that NP = T and CS $\subsetneq$ NP. Thus (xi) holds.

Thus the theorem is established.

The general principle used to prove the theorem has been frequently used in the study of algebraic systems as well as the study of subrecursive hierarchies. It is simply a matter of finding the appropriate notion of "universal" with respect to the class being investigated. However this principle has not been extensively used in the study of complexity classes of functions or languages. To investigate classes of languages, the concept of a principal AFL (or other related concept) appears to be useful, at least for subrecursive classes.

The classes studied in the theorem were chosen because of their possible relationships to problems concerning P vs. NP or DLBA vs. CS. Other classes could be investigated in this way, e.g., classes defined by stack automata or by tape-bounded Turing machines [1,2]--indeed, extensions of the theorem to such classes can be obtained using the principle behind the proof of the theorem and results in [1-3]. Clearly such extensions will be made, and it is assumed that there will be an investigation of pairs $(\mathscr{L}_1, \mathscr{L}_2)$ of classes such that there exists $L \varepsilon \mathscr{L}_1$ with the property that $L \varepsilon \mathscr{L}_2$ if and only if $\mathscr{L}_1 \subseteq \mathscr{L}_2$.

Finally, let us note that Albert Meyer has found a language $L \varepsilon$ CS which does not appear to be a principal generator for CS but which has the property that $L \varepsilon$ DLBA if and only if CS = DLBA.

## References

1. R. Book, S. Greibach, and B. Wegbreit, Time- and tape-bounded Turing acceptors and AFLs, J. Computer System Sci. 4 (1970), 606-621.

2. R. Book, S. Greibach, O. Ibarra, and B. Wegbreit, Tape-bounded Turing acceptors and principal AFLs, J. Computer System Sci. 4 (1970), 622-625.

3. R. Book, On languages accepted in polynomial time, SIAM J. Computing 1 (1972), to appear.

4. S. Cook, Characterizations of pushdown machines in terms of time-bounded computers, JACM 18 (1971), 4-18.

5. S. Cook, The complexity of theorem-proving procedures, Proceedings Third ACM Symposium on Theory of Computing (1971), 151-158.

6. S. Ginsburg and S. Greibach, Principal AFL, J. Computer System Sci. 4 (1970), 308-338.

7. N. Jones and A. Selman, Turing machines and the spectra of first-order formulas with equality, Proceedings Fourth ACM Symposium on Theory of Computing (1972), 157-167.

8. R. Karp, Reducibility among combinatorial problems, Proceedings IBM Symposium on Computational Complexity (to appear).

9. W. Savitch, Relationships between nondeterministic and deterministic tape complexities, J. Comput. System Sci. 4 (1970), 177-192.

10. B. Wegbreit, A generator of context-sensitive languages, J. Comput. System Sci. 3 (1969), 328-348.