Computational Complexity of Boolean Formulas with Query Symbols

ABSTRACT

Toshio SUZUKI

An abstract of
a dissertation submitted to the Doctoral Program
in Mathematics, the University of Tsukuba
in partial fulfillment of the requirements for the
degree of Doctor of Philosophy (Science)

January, 1999

Various researchers have suggested concepts that measures computational complexity of mathematical objects; for example, polynomial time Turing reducibility, Kolmogorov complexity, circuit complexity, and so on. However, we still have a variety of mathematical objects whose nature of computational complexity is unknown.

Arithmetical forcing was introduced by Feferman [5] soon after Cohen's independence proof of the continuum hypothesis. Since Hinman's work [6], arithmetical forcing has been studied in recursion theory. Later, arithmetical forcing and its variations were used as tools to study $P = ?NP$ question by some people. Typical examples are Dowd [4], Ambos-Spies et al. [1], Poizat [8] and Blum and Impagliazzo [3].

In this thesis, by extending the work of Dowd [4] and Poizat [8], we introduce the concept of *forcing complexity*. The forcing complexity of an arithmetical predicate for a given oracle means the minimal size of a finite portion of the oracle that forces the predicate. By using forcing complexity, we study the inner structure of $coNP[A]$ with respect to "almost all" oracles. In particular, we investigate computational complexity of $TAUT[A]$ and $rTAUT[A]$: these are sets of Boolean formulas with query symbols. There are two approaches to formalize the concept of "almost all" oracles. The one is a probabilistic formalization (we consider a subset of the Cantor space contains almost all oracles if it has Lebesgue measure one), and the other is a topological formalization (we consider a subset contains almost all oracles if it is comeager). This thesis is based on [9, 10, 11].

• Basic Concepts •

An oracle Turing machine is an algorithm that can utilize external information. Intuitively speaking, an oracle Turing machine is obtained by allowing a programmer to use special flow control statements of the following form.

```
if $u$ belongs to the oracle /* ⇐ this line is called a "query." */
    then ... ; else ... ;
end-if                       /* $u$ is a bit string. */
```

A set of bit strings, called an oracle, is fixed previously to the computation of a given oracle Turing machine. We denote the set of all bit strings by $\{0,1\}^*$. We identify a given oracle with its characteristic function. Thus, an oracle is a function from $\{0,1\}^*$ to $\{0,1\}$. Each recursive function is computed by a Turing machine; likewise, for a given oracle A, each function recursive in A is computed by an oracle Turing machine with the oracle A. Now, let n be a natural number. According to [4], we introduce an n-ary connective ξ^n. Roughly speaking, $\xi^n(q_1, \dots, q_n)$ asserts that the bit string $q_1 \cdots q_n$ belongs to the oracle that we are considering. More formally, for each oracle A, we introduce an n-ary Boolean function A^n. As an example, we explain the case where $n = 3$. First, we consider $\{0,1\}^3$, i.e. the set of all bit strings of lengths 3, and $\{0,1\}^*$. On these sets, we introduce lexicographic order, where shorter strings have priority and λ denotes the empty bit string.

$$\begin{array}{lllllllll} \{0,1\}^3: & 000, & 001, & 010, & 011, & 100, & 101, & 110, & 111. \\ \{0,1\}^*: & \lambda, & 0, & 1, & 00, & 01, & 10, & 11, & 000, \dots \end{array}$$

Then, let Str(3) be $\{\lambda, \ldots, 000\}$, i.e. the set of the first $2^3 = 8$ elements of $\{0,1\}^*$. We define the function A^3 so that the following diagram commutes, where $\simeq$ denotes the isomorphism with respect to the lexicographic order.

$$\begin{tikzcd}
\{0,1\}^3 \arrow[r, "A^3"] \arrow[d, "\simeq"'] & \{0,1\} \\
\mathrm{Str}(3) \arrow[ur, "A \restriction \mathrm{Str}(3)"'] &
\end{tikzcd}$$

For a given oracle A, we interpret $\xi^3(q_1, q_2, q_3)$ as $A^3(q_1q_2q_3)$. This rather obscure definition of A^n is forced on us because we want that the information contained in A^n be preserved in A^{n+1}, and also because a predicate in a tautology must have a definite number of arguments. The relativized propositional calculus is an extension of the propositional calculus. We get the former by adding a countable set $\{\xi^n : n \geq 1\}$ of connectives to the latter. For each oracle A, $TAUT[A]$ denotes the collection of all (binary representations of) relativized formulas that are tautologies with respect to A. Suppose that r is a positive integer. A relativized formula is called an r-query formula if it has just r-many occurrences of additional connectives. For each positive integer r, $rTAUT[A]$ denotes the collection of all (binary representations of) r-query formulas that are tautologies with respect to A. Each member of $rTAUT[A]$ is called an r-query tautology with respect to A.

• Methodological Features •

(1) Forcing complexity. A finite portion of an oracle is called a *forcing condition.* Let $X(\sim)$ be a unary predicate symbol denoting membership to a given oracle and y be a variable for a bit string. Assume that $\varphi(X)(y)$ is an arithmetical predicate (or, a functional) which is finitely testable (= test fini: see [8]). We say "a forcing condition S forces $\varphi(X)(u)$," where u is a given bit string, if $\varphi(A)(u)$ holds for any oracle A that is an extension of S. Now, assume that A is an oracle. *The forcing complexity of* $\varphi(X)(y)$ *relative to* A is a function $f : \mathbb{N} \to \mathbb{N}$ such that for each natural number n, $f(n)$ is the least number $k \in \mathbb{N}$ of the following property: for any bit string u of length n, if $\varphi(A)(u)$ is true then A has a finite portion S of size at most k such that S forces $\varphi(X)(u)$: in other words, the cardinality of $\mathrm{dom}(S)$ is at most k and for any oracle B extending S, $\varphi(B)(u)$ is true. If f is the forcing complexity of $\varphi(X)(y)$ relative to A and n is a natural number, the value $f(n)$ is called *the forcing complexity of* $\varphi(X)(y)$ *relative to* A *at* n, which is denoted by the following:

$$\mathrm{FC}(\varphi(X)(y), A, n).$$

We present applications of forcing complexity to the study of complexity of Turing machines. The results (a) and (b) in the following are shown by using forcing complexity.

(2) Application of forcing to deterministic algorithms. Dowd [4, Theorem 11] used forcing methods for controlling a nondeterministic Turing machine. We use forcing methods not only for controlling a nondeterministic Turing machine but also for controlling the execution time of a while-loop of a *deterministic* Turing machine, by which we clarify the relationship between the r-query tautologies and fundamental problems in the theory of computational complexity. The result (c) in the following is shown by using this method.

• Main Results •

(a) Clear-cut new proofs of previous results. We present an explicit example of a $coNP[X]$-predicate $\varphi(X)(y)$ with the following property: for each oracle A and for each positive integer n, a lower bound for its forcing complexity is (uniformly) given as follows.

$$\mathrm{FC}(\varphi(X)(y), A, n) \geq \frac{2^{n-1} - n + 1}{n}.$$

By this example, we give a simpler alternative proof of Dowd's result that t-generic oracles do not exist. We also present a clear-cut proof of Dowd's result that the set of all r-generic oracles has Lebesgue measure one, by investigating oracles' hierarchy with respect to forcing complexity.

(b) Results on Cohen-Feferman generic oracles. It is a classical result that the set $\{X : P[X] \neq NP[X]\}$ is comeager in the Cantor space ([7], [8], [3] and [4]). We consider oracles whose forcing complexity is small. By investigating how existence of such oracles affects behavior of a Cohen-Feferman generic oracle, we improve the above classical result. That is, for each positive integer r, we show that the following set is comeager in the Cantor space.

$$\{X : coNP[X] \not\subseteq NP[rTAUT[X]]\}.$$

(c) Control of while-loops by a forcing method. Bennet and Gill [2] showed: "The set $\{X : TAUT[X] \notin P[X]\}$ has Lebesgue measure one in the Cantor space." We consider the problem whether the statement of the above fact remains true when we substitute $rTAUT[A]$ for $TAUT[A]$. For each positive integer r and for each r-generic oracle A (in Dowd's sense), we show the following formula:

$$rTAUT[A] \equiv_T^P TAUT \oplus A.$$

The above formula is shown by constructing a deterministic algorithm whose while-loop's execution time is controlled by a forcing method. Consequently, we have that the following two assertions are equivalent, where R is a well-known complexity class such that $P \subseteq R \subseteq NP$.

(1) The set $\{X : rTAUT[X] \notin P[X]\}$ has Lebesgue measure one in the Cantor space.

(2) The unrelativized classes R and NP are not identical.

References

[1] Ambos-Spies, K., H. Fleischhack, and H. Huwig, "P-generic sets," pp. 58-68 in *ICALP 84*, Lect. Notes Comput. Sci. vol. 172, edited by J. Paredaens, Springer, Berlin, 1984.

[2] Bennett, C. H. and J. Gill, "Relative to a random oracle A, $P^A \neq NP^A \neq co\text{-}NP^A$ with probability 1," *SIAM J. Comput.*, vol. 10 (1981), pp. 96-113.

[3] Blum, M. and R. Impagliazzo, "Generic oracles and oracle classes," pp. 118-126 in *28th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, 1987.

[4] Dowd, M., "Generic oracles, uniform machines, and codes," *Information and Computation*, vol. 96 (1992), pp. 65-76.

[5] Feferman, S., "Some applications of the notions of forcing and generic sets," *Fund. Math.*, vol. 56 (1965), pp. 325-345.

[6] Hinman, P. G., "Some applications of forcing to hierarchy problems in arithmetic," *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 15 (1969), pp. 341-352.

[7] Mehlhorn, K., "On the size of sets of computable functions," pp. 190-196 in *14th Annual Symposium on Switching & Automata Theory*, IEEE Computer Society Press, Los Alamitos, 1973.

[8] Poizat, B., "$Q = NQ$?," *J. Symbolic Logic*, vol. 51 (1986), pp. 22-32.

[9] Suzuki, T., "Complexity of the r-query tautologies: in presence of a generic oracle," *Notre Dame J. Formal Logic, to appear.*

[10] Suzuki, T., "Recognizing tautology by a deterministic algorithm whose while-loop's execution time is bounded by forcing," *Kobe Journal of Mathematics, to appear.*

[11] Suzuki, T., ' Forcing complexity: supplement to "complexity of the r-query tautologies," ' *preprint.*