On the Complexity of Small Description and Related Topics *

Osamu Watanabe

Department of Computer Science, Tokyo Institute of Technology
Meguro-ku Ookayama, Tokyo 152, Japan
e-mail: watanabe@cs.titech.ac.jp

Abstract. The class P/poly is known to be the class of sets with small descriptions, more specifically, polynomial size circuits. In this paper, we discuss the problem of obtaining the polynomial size circuits for a given set in P/poly by using the set as an oracle. Recent results on upper and lower bounds of the relative complexity of this problem are presented. We also introduce two related research topics — query learning and identity mapping network — and explain how they are related to this problem.

1 Introduction

Sets with a small description have often been the subject of investigation in various contexts of computational complexity theory. In particular, sets in P/poly [KL80] are regarded as standard examples of sets with a small description, for those sets have several sorts of "small descriptions": polynomial size advice function, polynomial size circuits, sparse oracle set, etc. In this paper, we discuss the complexity of computing "small description" for sets in P/poly.

In order to be more specific, we fix our notion of "small description." It is known that P/poly is the class of sets having polynomial size circuits. (Here circuits are acceptors of strings in \{0, 1\}*. ) Intuitively, we consider those polynomial size circuits as "small description." More precisely, we consider some polynomial time computable binary predicate \(I_0\), circuit interpreter, that evaluates a given circuit on a given input. That is, for a given circuit (description) \(w\) and a string \(x \in \{0, 1\}^*\), \(I_0(w, x)\) simulates \(w\) on \(x\) and yields true if and only if \(w\) accepts \(x\). It is easy to show that such a predicate exists; furthermore, we can prove that every set \(L\) in P/poly has some function \(g_L\) with the following property:

\[(a) \exists p: \text{polynomial}, \forall n \geq 0 \ [ |g(0^n)| \leq p(n) ], \text{ and} \]
\[(b) \forall n \geq 0, \forall x \in \{0, 1\}^* \ [ x \in L \leftrightarrow I_0(g(0^n), x) ].\]

That is, \(g_L(0^n)\) is regarded as a description of some circuit recognizing \(L^n\). Thus, a function such as \(g_L\) is called a circuit generator for \(L\). In this paper\(^2\), the complexity

---

* This research was supported in part by Takayanagi Foundation of Electronics and Science Technology (1992).

\(^2\) We may be able to discuss similarly by considering the complexity of a sparse set to which \(L\) is reducible. Note, however, that these two complexity notions are not identical (see [GW91]).
of obtaining "small description" for \( L \) is interpreted as the complexity of computing a circuit generator for \( L \).

In this paper, the "relative" complexity is used for measuring the complexity of circuit generators. That is, for a given \( L \) in \( P/poly \), we discuss the complexity of computing its circuit generator when \( L \) is used as an oracle. For example, we say that \( L \) has a polynomial time computable circuit generator \( g_L \) if \( g_L \) is polynomial time computable relative to \( L \). Note also that a circuit generator for \( L \) is not unique in general. Thus, we cannot conclude that \( L \)'s circuit generator is not polynomial time computable until we prove that no circuit generator for \( L \) is polynomial time computable.

While there has been a considerable amount of research on the complexity of sets with a small description (see, e.g., [Boo92, LOW92]), the complexity of "small description" itself has not been asked nor investigated well. However, researchers have begun to notice the importance of this subject recently, and some interesting results have been obtained in last two years. In this paper, we explain these recent results and related open problems by interpreting them for the complexity of circuit generators.

Another purpose of this paper is to explain the importance of our subject, the complexity of circuit generators, by showing its close relationship to two different topics: polynomial time query learnability, and the complexity of identity mapping networks.

Query learning is one paradigm of computational learning. Suppose someone, who is called a teacher, has some set, which is called a target concept, in his mind. Roughly speaking, query learning is to obtain the target concept by asking queries on it to the teacher.

Angluin [Ang87] showed a learning algorithm for regular sets represented by deterministic finite automata. Since then, several polynomial time query learning algorithms have been invented [Ang88, BR87, Ish89, Sak88]. On the other hand, it has been believed widely that some concept class is not polynomial time query learnable. Indeed, some negative results have been obtained from certain cryptographic assumptions [AK91, KV89, Val84]. However, we have not been able to get similar negative results from a weaker assumption such as \( P \neq NP \). Watanabe and Gavaldà [WG92] proposed one approach for proving negative results and demonstrated its possibility. Here we explain their approach and show that our subject, the complexity of circuit generators, plays a key role in this approach.

Watanabe [Wat91] introduced a new computation model, a three layer network computing identity mapping (in short, identity mapping network) for a model of data compression or pattern learning.

A three layer network is a special type of logical circuit, which consists of three set of nodes — input nodes, internal nodes, and output nodes — and logical circuits — encoder and decoder — between them (Fig. 1.1). A network has the same number of input and output nodes, and usually has smaller number of internal nodes. (One can consider any node in the network as an internal node so long as no output node is connected directly to any input node.)

Let \( N^n \) be a three layer network with \( n \) input/output nodes. We can naturally regard it as a function from \( \{0, 1\}^n \) to \( \{0, 1\}^n \). A string in \( \{0, 1\}^n \) given to a network is called an input pattern. For any input pattern set \( X^n \subseteq \{0, 1\}^n \), we say that \( N^n \)