Synopsis

The thesis is divided into two main parts. The first part deals with proof systems computable by Boolean circuit families that characterize the complexity class \( \text{NC}^0 \) (bounded fanin, constant depth), which is one of the weakest complexity classes. A proof system for a language \( L \) is a function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) such that \( \text{Range}(f) = L \). We initiate a study of \( \text{NC}^0 \) computable proof systems with an overarching goal of showing that such proof systems cannot capture the language \( \text{Taut} \). We begin by studying \( \text{NC}^0 \) proof systems in the context of regular languages. We give sufficient conditions for a regular language to have a proof system computable in \( \text{NC}^0 \). On the other hand, we show that an explicit regular language does not have a proof system computable in \( \text{NC}^0 \). By generalizing techniques used in constructing proof systems for regular languages, we construct \( \text{NC}^0 \) proof systems for languages complete for various complexity classes ranging from \( \text{NC}^1 \) to \( \text{P} \). It remains open to characterize the regular languages that indeed have proof systems that are computable in \( \text{NC}^0 \). In the context of \( \text{Taut} \), we study \( 2\text{TAUT} \) and show a reduction from \( 2\text{TAUT} \) to the language associated with directed connectivity in terms of proof systems. We show that the set of all undirected graphs that have a path between two fixed vertices \( s \) and \( t \) has an \( \text{NC}^0 \) proof system. Our study shows that the question of whether a language can be generated using these restricted proof systems is unrelated to the computational complexity of their associated membership problem.

In the second part of the thesis, we study the problem of testing if a given arithmetic circuit computes the identically zero polynomial (\( \text{PIT} \)) and give efficient algorithms for certain special cases. We also determine the complexity of other natural problems that arise in the context of arithmetic circuits. We give a multilinearity and identity test for read-thrice formulas. We then give efficient algorithms for \( \text{PIT} \) on polynomials of the form \( f_1 f_2 f_3 \cdots f_m + g_1 g_2 \cdots g_s \) where \( f_i \)s and \( g_i \)s are presented as read-once formulas. We show a hardness of representation for the elementary symmetric polynomial against read-once formulas with the added restriction that every leaf is labeled \( ax \) where \( a \) is a non-zero field element and \( x \) is a variable. Finally, we study some natural problems in the context of arithmetic circuits. These include counting the number of monomials, and checking
if a given monomial has non-zero coefficient in the polynomial computed by a given arithmetic circuit. We observe that even for monotone (no negative constants) read-twice formulas, counting the number of monomials is \#P-hard. We also show that checking if the coefficient of a monomial is zero in a polynomial computed by a read-once formula is in logspace.