RESTRICTED BRANCHING PROGRAMS AND THEIR COMPUTATIONAL POWER

Christoph Meinel
Sektion Informatik
Humboldt-Universitat zu Berlin
DDR-1086 Berlin, PF 1297

ABSTRACT

In order to acquire insight about arbitrary branching programs, a number of restricted branching program models have been considered. Among these are decision trees, read-once-only branching programs, length-restricted oblivious branching programs, and width-restricted branching programs. In the following we survey some results which characterize the computational power of such restricted models. Interestingly, we are able to establish strong differences in the computational power of deterministic, nondeterministic, parity, or alternating restricted branching programs for most of the mentioned types. For details we refer to [Me89].

INTRODUCTION

One of the fundamental issues of complexity theory is to estimate the relative efficiency of different models of computation. A general program in doing this has been to take abstract models of computation, such as Turing machines, Random Access Machines, Boolean circuits or branching programs, and examine their behavior under certain resource constraints. This leads to the definition of complexity classes which formalize certain computational powers. By examining the meaning of and the relationships between such classes, one seeks to understand the relative strengths of their underlying computational paradigms.

In recent years the concepts of Boolean circuit complexity, branching program complexity and other nonuniform complexities have been (re)discovered. They are based on computational models which are purely combinatorial objects. Apart from the strong practical interest in investigations of such circuit based models, nonuniform complexity classes appear to be more amenable to combinatorial analysis. Results concerning restricted circuit models obtained e.g. in [PSS81, Ra85,
Ra86, An85, Ya85, Ha86] have advanced our knowledge of nonuniform complexity classes as well as of complexity classes in general.

In order to acquire insight about arbitrary branching programs, a number of restricted branching program models have been considered. Among these are decision trees (e.g. [We84, DM89]), read-once-only branching programs (e.g. [Ma76, We88, Ya84, A&86, KW87, KMW88]), length-restricted oblivious branching programs (e.g. [KW89, KMW89]), and width-restricted branching programs (e.g. [Pu84, Ba86, A&86, Me87]). In the following we survey some results which characterize the computational power of such restricted models. Interestingly, we are able to establish strong differences in the computational power of deterministic, nondeterministic, parity, or alternating restricted branching programs for most of the mentioned types. For details see e.g. [Me89].

We assume that the reader is familiar with the two basic computation models of Boolean circuits and Turing machines. In particular, logarithmic space-bounded and polynomial time-bounded Turing machine complexity classes are of special interest in the following.

\[ L = \text{DSPACE}(\log n), \quad NL = \text{NSPACE}(\log n), \quad \text{co-NL} = \text{co-NSPACE}(\log n), \]
\[ AL = \text{ASPACE}(\log(n)) = \text{DTIME}(n^{O(1)}) = P. \]

In order to relate complexity classes defined by Turing machines with circuit-based complexity classes one considers nonuniform Turing machine complexity classes [KL80]. If \( f \) is a function on the natural numbers and if \( K \) denotes any Turing machine complexity class then we define

\[ K/f(n) = \{ A \subseteq \{0,1\}^* \mid \text{there is an advice } \alpha : \mathbb{N} \rightarrow \{0,1\}^* \text{ with } |\alpha(n)| \leq O(f(n)) \text{ and a Turing machine operating within the resource constraints of } K \text{ which accepts } w\#\alpha(|w|) \text{ iff } w \in A \}. \]

For the family \( n^{O(1)} \) of polynomial bounded functions we define

\[ L = L/n^{O(1)}, \quad NL = NL/n^{O(1)}, \quad \text{and } AL = AL/n^{O(1)} = P/n^{O(1)}. \]

1. BRANCHING PROGRAMS AND THEIR COMPUTATIONAL POWER

One of the most important models for investigating the computational complexity of Boolean functions is that of branching programs which generalize the concept of decision trees to decision graphs. The settings of \( n \) input variables determine a flow of control through a