FUNCTION-ALGEBRAIC CHARACTERIZATIONS OF LOG AND POLYLOG PARALLEL TIME

STEPHEN BLOCH

Abstract. The main results of this paper are recursion-theoretic characterizations of two parallel complexity classes: the functions computable by uniform bounded fan-in circuit families of log and polylog depth (or equivalently, the functions bitwise computable by alternating Turing machines in log and polylog time). The present characterizations avoid the complex base functions, function constructors, and a priori size or depth bounds typical of previous work on these classes. This simplicity is achieved by extending the “tiered recursion” techniques of Leivant and Bellantoni & Cook.

Key words. Circuit complexity; subrecursion.
Subject classifications. 68Q15, 03D20, 94C99.

1. Introduction

Researchers in computational complexity have tried for decades to describe computational classes without reference to Turing machines or other specific models of computation. One early success, Cobham (1965), characterized the functions computable in polynomial time as the algebra generated from a few base functions by closure under composition and a form of recursion. Cobham’s characterization was substantially simplified with new techniques in Bellantoni & Cook (1992). By extending these techniques, I characterize two parallel function classes: the functions computable in parallel time proportional to the log, or to a polynomial in the log, of the input length. Here “parallel time” refers to either the alternating Turing machine or the uniform, bounded fan-in circuit family model of computation.

Cobham’s characterization inspired many other recursion-theoretic and logical characterizations of complexity classes. The Cobham characterization provided early support for \( \mathcal{FP} \) as a natural class, but this support was weakened by the explicit bounds and unfamiliar base functions in the characteriza-
tion. Such problems—artificial size or depth bounds, base functions introduced only for their growth rates, computationally complex base functions, and untidy collections of composition and recursion schemes—continued to plague later researchers. Lind (1974) gave a recursion-theoretic characterization of logspace, and Allen (1991) gave logical and recursion-theoretic characterizations of $NC$, both explicitly bounding function growth rates by polynomials. Compton & LaFlamme (1990) presented recursion-theoretic and logical characterizations of uniform $NC^1$, relying on a polynomial size bound implicit in their finite-models approach. Clote (1989) characterized uniform $NC^1$, $AC^0$, and a variety of other parallel classes with a polynomial-bounded recursion scheme, using a complete problem from Buss (1987). A more elegant characterization, in Clote (1993), used the completeness result of Barrington (1989) to simplify the base functions and impose a constant bound on recursively-defined functions. Buss (1986), using a variant of Cobham’s characterization, showed that the functions at each level of the polynomial-time hierarchy are precisely those provably total in a corresponding weak fragment of Peano arithmetic. Buss’s and Clote’s characterizations, however, required a language expressly designed to produce polynomial growth rates. Arai (1992) characterized uniform $NC^1$ with systems of bounded arithmetic based on the “upward tree recursion” of Compton & LaFlamme (1990), while Clote & Takeuti (1992) did the same with a log-depth weak induction scheme; both of these used only the usual language of number theory but explicitly specified $O(\log(n))$ recursion depth (and hence $2^{O(\log(n))} = n^{O(1)}$ size).

By contrast, the results of this paper involve no explicit size or depth bounds, no functions introduced solely to produce desired growth rates, only computationally easy base functions—so easy as to be computable by constant-depth, bounded fan-in circuits—and only one composition and one recursion scheme. Essential to these simpler characterizations is the notion (introduced in Leivant 1990 and Bellantoni & Cook 1992) of classifying the parameters of a function into “tiers” by how they are used in the computation. The classification depends on whether a parameter is used only for local, bitwise operations or is used in its entirety to bound a loop or a recursion. The former sort of parameter has much less impact on the resources necessary to compute the function, and is called “safe,” or “tier 0;” the latter is called “normal,” or “tier 1.” (The “safe/normal” terminology is due to Bellantoni & Cook 1992. The “tier” terminology, due to Leivant 1990, generalizes to higher-numbered tiers. In this paper we shall only be concerned with tiers 0 and 1, so we use the two terminologies interchangeably.) By distinguishing tiers of parameters syntactically, one can bound computational resource costs without unduly limiting