Open Least Element Principle and Bounded Query Computation

L.D. Beklemishev *

Steklov Mathematical Institute
Gubkina 8, 117966 Moscow, Russia
e-mail: bekl@mi.ras.ru, beklemi@math.uni-muenster.de

Abstract. We show that elementary arithmetic formulated in the language with a free function symbol $f$ and the least element principle for open formulas (where we assume that the symbols for all elementary functions are included in the language) does not prove the least element principle for bounded formulas in the same language. A related result is that composition and any number of unnested applications of bounded minimum operator are, in general, insufficient to generate the elementary closure of a function, even if all elementary functions are available. Thus, unnested bounded minimum operator is weaker than unnested bounded recursion.

1 Introduction and Motivation

This paper arose out of the problem of separating the schemes of $\Delta_1$-induction and $\Sigma_1$-collection in arithmetic [4,6]. It turns out that this question is closely related to the comparison of different operators generating the elementary closure of a class of functions and to the problems of separating the corresponding systems of subrecursive arithmetic. These questions are also natural from a purely computational point of view.

In this paper we compare the relative strength of bounded $\mu$-operator and bounded recursion. We show that unnested bounded $\mu$-operator is, in general, weaker than unnested bounded recursion. In contrast, it is well known that, when nestings are allowed, each of the two operators together with composition is sufficient to generate the elementary closure of a class of functions.

We compare the strength of the two operators against the problem of computing the maximum of a function on a finite interval. In order to compute $\max_{i \leq x} f(i)$ on a Turing machine with a function oracle for $f$ one needs of order $x$ different oracle queries (see [2]). In particular, any Turing machine that may only ask a bounded number of queries cannot, in general, compute this function. This implies that the class of functions generated from all elementary functions and $f$ by composition does not, in general, coincide with the elementary closure of a function $f$. This idea was used in [2] to show the independence of $\Sigma_1$-collection rule in arithmetic, which improved the result of Parsons on the independence of

* Supported by Alexander von Humboldt Foundation and RFBR grant 98-01-00249.
$\Sigma_1$-collection schema from the set of all true arithmetical $\Pi_2$-sentences (see also $\S$ for a related work on subrecursive degrees).

If unnested applications of bounded $\mu$-operator are allowed on a par with composition, then the available power of computation increases compared to the bounded query oracle Turing machine. It is worth explaining here informally, why such a computation mechanism is still too weak to compute the maximum of $f$.

A $\mu$-operator of the form $\mu i \leq x. R(i)$, where $R(i)$ is a bounded query predicate, can be interpreted in terms of a parallel bounded query machine. To evaluate this operator the machine generates $x + 1$ independent subprocesses, $i$-th process $P_i$ evaluates the predicate $R(i)$ and returns true or false. Then the machine runs through their outputs to find the least $i$ such that $R(i)$ evaluates to true.

Since $R(x)$ is a bounded query predicate, all the subprocesses have a uniform bound on the number of queries each of them may ask. More important still, each subprocess $P_i$ only returns true or false, that is, exactly one bit. This means that the processes cannot exchange too much information. This is crucial for the fact that $\max_{i \leq x} f(i)$ cannot be computed by such a machine: each subprocess $P_i$ can only learn the values of $f$ on a boundedly small subset of the large interval $[0, x]$, but it lacks the ability to communicate, say, the maximum of these values to other processes. (Compare with the usual algorithm of computing the maximum of $f$ by querying successively $f(0), f(1), \ldots, f(x)$. Here, one has to always store the intermediate maximum value of $f$, which may potentially exceed any bounded number of bits.)

Of course, this rough idea will be made more precise in the proof of our main result. This proof is based on a recursion-theoretic diagonal construction that involves a combinatorial argument using infinite Ramsey theorem.

2 Statement of the Results

As usual, for a given predicate $R(x, v)$ the expression $\mu x \leq a. R(x, v)$ denotes the function

$$m(a, v) = \begin{cases} \text{the minimal } x \leq a \text{ such that } R(x, v) \text{ holds,} & \exists x \leq a R(x, v) \\ a + 1, & \text{otherwise.} \end{cases}$$

For a set of functions $K$, let $C(K)$ denote the closure of $K$ and the class of elementary functions $E$ under composition. Further, let $[K, M]$ denote the closure of the class $K \cup E$ under composition and unnested applications of bounded minimum operator, that is, the closure under composition of $K \cup E$ and all functions of the form $\mu x \leq a. R(x, v)$, where $R(x, v) \in C(K)$. Similarly, $[K, BR]$ denotes the closure of $K \cup E$ under composition and unnested applications of bounded recursion schema, that is, primitive recursion bounded by some function from $C(K)$. The closure of $K \cup E$ under composition and (nested) bounded recursion is called the elementary closure of $K$ and is denoted $E(K)$. By a result of Parsons $[9]$, $E(f) = C(f)$, where $f$ denotes the function $f(x) = \langle f(0), \ldots, f(x) \rangle$. 