

A Note on the Relation between Polynomial Time Functionals and Constable's Class \mathcal{K}

P. Clote*

clote@informatik.uni-muenchen.de

Institut für Informatik, Ludwig-Maximilians-Universität München, Oettingenstr. 67,
D-80538 München.

Abstract. A result claimed without proof by R. Constable in a STOC73 paper is here corrected: a strictly increasing function f is presented for which Constable's class $K(f)$ is properly contained in $FP(f)$, the collection of functions polynomial time computable in f .

Introduction

In [10] A. Cobham² was the first to isolate the notion of polynomial time computable function. There he characterized the class FP of polynomial time computable functions as the smallest function algebra \mathcal{L} containing certain initial functions and closed under composition and a certain variant of primitive recursion called *bounded recursion on notation*. Since Cobham's seminal work, a number of other complexity classes have been characterized by function algebras, such as linear space [18]³, logspace [15], polynomial space [20], exponential time [17], certain general complexity classes [22], AC^0 , AC^k , NC [9, 1], $ACC(2)$, $ACC(6)$, TC^0 [7], etc.⁴

One of the first persons to consider type 2 functional computational complexity was R. Constable, who in [11] introduced a type 2 machine model and related programming language, and then studied polynomial time reducibilities between functions. On p. 118 of [11], Constable defined the class $\mathcal{L}(f)$ to be the collection of functions of the form $\lambda \mathbf{x} F(f, \mathbf{x})$, where F is a type 2 polynomial time computable operator; i.e. $\mathcal{L}(f)$ is the collection of all functions polynomial time computable in f . From subsequent work of K. Mehlhorn [16] and especially of B. Kapron and S. Cook [14], it is known that Constable's class $\mathcal{L}(f)$ can alternately be defined in a machine independent manner as the smallest class of functions containing Cobham's initial functions together with f , and closed under composition and bounded recursion on notation.

* Part of this research supported by NSF CCR-9408090 and US-Czech Science and Technology Program 93025.

² J. Edmonds [12] independently isolated the notion of *good* (i.e. polynomial time) algorithm.

³ Ritchie's work was actually prior to that of Cobham.

⁴ For a survey, see chapter 10 of [23] or [4].

Drawing on analogy with the Kalmár elementary functions⁵ Constable defined $K(f)$ to be the smallest class of functions containing the initial functions $+$, $-$, \times , $\lfloor x/y \rfloor$, f and closed under the operations of substitution, explicit transformation, length bounded addition $f(x, \mathbf{y}) = \sum_{i=0}^{\lfloor x \rfloor} h(i, \mathbf{y})$ and length bounded multiplication $f(x, \mathbf{y}) = \prod_{i=0}^{\lfloor x \rfloor} h(i, \mathbf{y})$. The class K is defined as previously, but without the initial function f .⁶ On page 118 of [11], the following claim is stated as a theorem without proof.

Claim I

- (a) For all f , $K(f) \subseteq \mathcal{L}(f)$.
- (b) For all non-decreasing f , $K(f) = \mathcal{L}(f)$.

While part (a) is clearly true, the purpose of this note is to present a counterexample to part (b). On the same page of [11], the following claim is stated as a corollary without proof.

Claim II $K = \mathcal{L}$.

Since Cobham [10] had characterized the polynomial time computable functions FP by the function algebra \mathcal{L} , this corollary would have given a very elegant characterization of FP . This corollary is unlikely to be true, since by consideration of a binary tree whose inner parent nodes perform the sum [resp. product] of the values associated with the children, it is easy to see that $K \subseteq NC \subseteq FP$. Hence if NC is properly contained in the class FP of polynomial time computable functions, then the claimed corollary is false. Similarly, though our proof proceeds slightly differently, it is easy to see that for functions f_L, c_L associated with C. Wilson's oracle separating NC from P , it is the case that

$$K(f_L) = K(c_L) \subseteq \mathcal{F}NC^L \subset \mathcal{F}P^L = \mathcal{L}(f)$$

and hence that $K(f) \subset \mathcal{L}(f)$ for a particular non-decreasing function.⁷

I believe it is worth clarifying the status of these claims (for which no proofs can be found in the literature), since on page 194 of [23], Claim II is stated as Theorem 10.27 again without proof.

R. Constable's work [11] was quite seminal. K. Mehlhorn [16]⁸ subsequently studied polynomial time reducibility between functions, and (essentially) defined the collection BFF of basic feasible type 2 functionals as the straightforward extension of Cobham's machine-independent characterization of the polynomial

⁵ The elementary functions [13] form the smallest class containing certain initial functions including 2^x and closed under composition, bounded addition $g(x, y) = \sum_{i=0}^x h(i, y)$, and bounded multiplication $g(x, y) = \prod_{i=0}^x h(i, y)$. It is known that this class coincides with those functions computable in time (or space) bounded by a finite stack of 2's topped by the length of the input.

⁶ In the terminology of [11], $K(f()) = [+ , - , \times , \lfloor x/y \rfloor , f() ; Os , \sum_{\lfloor x \rfloor} , \prod_{\lfloor x \rfloor}]$.

⁷ All unexplained notation is later introduced.

⁸ [16] contains results from Mehlhorn's Ph.D. dissertation written under the direction of R. Constable.

time computable functions to type 2.⁹ Mehlhorn proved (essentially) that BFF equals the collection of type 2 functionals $F(f, x)$ for which there exists a function oracle Turing machine M such that $M(f, x) = F(f, x)$ for all f, x and the runtime of $M(f, x)$ is bounded by $|G(f, x)|$ where $G \in BFF$. Mehlhorn measured the cost for function oracle calls as 1 (unit cost).

In [14], B. Kapron and S. Cook proved a difficult extension of Mehlhorn's theorem. Their result states that a type 2 functional $F \in BFF$ if and only if there is a Turing machine M computing F , for which the runtime of $M(f, x)$ is bounded by $P(|f|, |x|)$, where P is a *second order polynomial*. B. Kapron, A. Ignjatovic and the author [6] then characterized type 2 AC^0 functionals and gave a characterization of type 2 NC functionals.¹⁰

The plan of this paper is as follows. In section 1 basic definitions and background results are given. In section 2 we prove that $K(f)$ is properly contained in $\mathcal{L}(f)$ for some increasing f . In section 3 we show K contains TC^0 and ACC , and pose the question whether K contains $ALOGTIME$ and $LOGSPACE$.

1 Definitions

1.1 Oracle Turing machine

In [11], R. Constable introduced a natural programming language including *function oracle calls*, and with respect to this model defined the notion of polynomial time type 2 functional. In [14], B. Kapron and S. Cook defined the notion of *norm* (or length) of a function, and studied polynomial time (function) oracle machines, running in second order polynomial time.

Definition 1. The length of x in binary satisfies $|x| = \lceil \log_2(x + 1) \rceil$. The *length* or *norm* $|f|$ of function f is

$$|f|(n) = \max_{|x| \leq n} |f(x)|.$$

Let f_1, \dots, f_m be variables ranging over \mathbf{N}^N and x_1, \dots, x_n be variables ranging over \mathbf{N} . The collection C of second order polynomials $P(f_1, \dots, f_m, x_1, \dots, x_n)$ is defined inductively as follows.

- (i) for any integer c , $c \in C$,
- (ii) for every $1 \leq i \leq n$, $x_i \in C$,
- (iii) if $P, Q \in C$ then $P + Q \in C$ and $P \cdot Q \in C$,
- (iv) if $P \in C$ then $f_i(P) \in C$ for $1 \leq i \leq m$.

⁹ Mehlhorn called such functionals polynomial time computable operators. This class of functionals was then studied by M. Townsend [21], who called them *POLY*, and later by B. Kapron and S. Cook [14], who first denoted this class as *BFF*, the basic feasible functionals of type 2.

¹⁰ The characterization of NC in [6] extended Mehlhorn's approach; the parallel analogue of [14] will appear in the journal version of [6].

In the Kapron-Cook model, a type 2 functional F is *polynomial time computable* if there is a second order polynomial P and an oracle Turing machine M , such that $F(f, x) = M(f, x)$, where the runtime of $M(f, x)$ is at most $P(|f|, |x|)$. Here, the cost for a function oracle call $f(y)$ is $|f(y)|$.

The main result of [14] was a characterization of type 2 polynomial time computable functionals — the function algebra BFF of *basic feasible functionals* equals the type 2 polynomial time computable functionals. The formal definition of BFF , given in Definition 19, is a straightforward type 2 generalization of Cobham’s function algebra \mathcal{L} . It immediately follows from [14] that R. Constable’s class $\mathcal{L}(f)$ is equal to the function algebra $[0, I, s_0, s_1, \#, f; \text{COMP, BRN}]$, which is equal to the collection of functions $\lambda \mathbf{x} F(f, \mathbf{x})$, where $F \in BFF$.

1.2 Circuit families

An *oracle boolean circuit* is a directed acyclic graph. Nodes with fan-in 0 are labeled by $x_1, \dots, x_n, 0, 1$. All other nodes, called *gates* are labeled by one of $\wedge, \vee, \neg, ?$, the latter called an *oracle gate*. Nodes labeled by \neg have fan-in 1, all other gates have arbitrary fan-in. All nodes except for the unique *output* node have arbitrary fan-out; the output node has fan-out 0. The *size* $s(\alpha)$ of a circuit α is the number of gates. The *depth* $d(\alpha)$ of a circuit is defined recursively as follows.

$$d(\alpha) = \begin{cases} 0 & \text{if } \alpha \text{ is labeled by } x_1, \dots, x_n, 0, 1 \\ 1 + d(\beta) & \text{if } \alpha \text{ is } \neg\beta \\ 1 + \max_{1 \leq i \leq m} (d(\beta_i)) & \text{if } \alpha \text{ is } \bigvee_{i=1}^m \beta_i \text{ or } \bigwedge_{i=1}^m \beta_i \\ |m| + \max(d(\beta_1), \dots, d(\beta_m)) & \text{if } \alpha \text{ is } ?(\beta_1, \dots, \beta_m) \end{cases}$$

If $A \subseteq \{0, 1\}^*$, $x \in \{0, 1\}^*$, then an oracle boolean circuit α *computes* on input A, x in the obvious manner. Formally, one defines a function $Eval(\alpha, A, w)$, where

$$Eval(\alpha, A, w) = \begin{cases} \text{label of } \alpha & \text{if } \alpha \text{ is labeled by } x_1, \dots, x_n, 0, 1 \\ \neg Eval(\beta, A, w) & \text{if } \alpha \equiv \neg\beta \\ \bigvee_{1 \leq i \leq m} Eval(\beta_i, A, w) & \text{if } \alpha \text{ is } \bigvee_{i=1}^m \beta_i \\ \bigwedge_{1 \leq i \leq m} Eval(\beta_i, A, w) & \text{if } \alpha \text{ is } \bigwedge_{i=1}^m \beta_i \\ 1 & \text{if } \alpha \equiv ?(\beta_1, \dots, \beta_m) \\ & \text{and } (\beta_1(w), \dots, \beta_m(w)) \in A \\ 0 & \text{if } \alpha \equiv ?(\beta_1, \dots, \beta_m) \\ & \text{and } (\beta_1(w), \dots, \beta_m(w)) \notin A \end{cases}$$

AC_k^A is the collection of all languages computed by a logtime uniform family $\langle \alpha_n : n \in \mathbf{N} \rangle$ of oracle boolean circuits, where $s(\alpha_n) = n^{O(1)}$ and depth $d(\alpha_n) = O(\log^k(n))$ (see [2]). NC_k^A is similarly defined, but where \wedge -gates and \vee gates are restricted to be of fan-in 2. AC_k^\emptyset and NC_k^\emptyset are usually designated by AC^k and NC^k , and $NC = \cup AC^k = \cup NC^k$.¹¹ Most often in the literature, AC^k ,

¹¹ Here we follow Wilson’s convention of defining boolean complexity classes of *languages* rather than functions, as well as his convention of writing AC_k in place of AC^k to allow for an oracle superscript.

NC^k and NC refer to *function classes* rather than classes of relations. Here, we define \mathcal{FNC} to be the class of functions $f(\mathbf{x})$ such that

- (i) f is of polynomial growth rate; i.e. there is a multivariable polynomial p such that

$$|f(x_1, \dots, x_n)| \leq p(|x_1|, \dots, |x_n|)$$

- (ii) the bitgraph $A_f \in NC$, where $A_f(i, \mathbf{x})$ holds iff the i -th bit of $f(\mathbf{x})$ is 1.

Similarly the classes of functions of polynomial growth rate whose bitgraph belongs to AC^k [resp. NC^k] is designated here as \mathcal{FAC}^k [resp. \mathcal{FNC}^k]. Finally, \mathcal{FNC}^L [resp. \mathcal{FAC}_k^L , \mathcal{FNC}_k^L] denotes the class of functions of polynomial growth rate whose bitgraph belongs to the relativized class NC^L [resp. AC_k^L , NC_k^L].

1.3 Function algebras

Definition 2. If \mathcal{X} is a set of functions and OP is a collection of operations, then $[\mathcal{X}; \text{OP}]$ denotes the smallest set of functions containing \mathcal{X} and closed under the operations of OP . The set $[\mathcal{X}; \text{OP}]$ is called a *function algebra*. The *characteristic function* $c_p(\mathbf{x})$ of a predicate P satisfies

$$(1) \quad c_p(\mathbf{x}) = \begin{cases} 1 & \text{if } P(\mathbf{x}) \\ 0 & \text{else,} \end{cases}$$

where P is often written in place of c_P . If \mathcal{F} is a class of functions, then \mathcal{F}_* is the class of predicates whose characteristic function belongs to \mathcal{F} .

Definition 3. The *successor* function $s(x) = x + 1$; the *binary successor* functions s_0, s_1 satisfy $s_0(x) = 2 \cdot x$, $s_1(x) = 2 \cdot x + 1$; the *smash* function $x \# y = 2^{|x| \cdot |y|}$; the n -place projection functions $I_k^n(x_1, \dots, x_n) = x_k$; I denotes the collection of all projection functions.

Definition 4. The function f is defined by *composition* (COMP) from functions h, g_1, \dots, g_m if

$$f(x_1, \dots, x_n) = h(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)).$$

Definition 5 (A. Cobham [10]). The function f is defined by *bounded recursion on notation* (BRN) from g, h_0, h_1, k if

$$\begin{aligned} f(0, \mathbf{y}) &= g(\mathbf{y}) \\ f(s_0(x), \mathbf{y}) &= h_0(x, \mathbf{y}, f(x, \mathbf{y})) \text{ if } x \neq 0 \\ f(s_1(x), \mathbf{y}) &= h_1(x, \mathbf{y}, f(x, \mathbf{y})) \end{aligned}$$

provided that $f(x, \mathbf{y}) \leq k(x, \mathbf{y})$ for all x, \mathbf{y} . Define the algebras

$$\begin{aligned} \mathcal{L} &= [0, I, s_0, s_1, \#; \text{COMP}, \text{BRN}] \\ \mathcal{L}(f) &= [0, I, f, s_0, s_1, \#; \text{COMP}, \text{BRN}]. \end{aligned}$$

The class FP consists of the collection of polynomial time computable functions.

Theorem 6 A. Cobham [10]. $FP = \mathcal{L}$.

See [19] for a detailed proof of Cobham's theorem.

Definition 7 (R. Constable [11]). The function f is defined by *weak summation* (WSUM) [resp. *weak product* (WPROD)] from g if $f(x, \mathbf{y})$ equals

$$\sum_{i=0}^{|x|} g(i, \mathbf{y}) \quad [\text{resp. } \prod_{i=0}^{|x|} g(i, \mathbf{y})].$$

Define the algebras

$$\begin{aligned} K &= [0, I, s_0, s_1, +, \div, \times, \lfloor x/y \rfloor; \text{COMP, WSUM, WPROD}] \\ K(f) &= [0, I, s_0, s_1, +, \div, \times, \lfloor x/y \rfloor, f; \text{COMP, WSUM, WPROD}]. \end{aligned}$$

Definition 8 (P. Clote [9]). Assume that $h_0(x, \mathbf{y}), h_1(x, \mathbf{y}) \leq 1$. The function f is defined by *concatenation recursion on notation* (CRN) from g, h_0, h_1 if

$$\begin{aligned} f(0, \mathbf{y}) &= g(\mathbf{y}) \\ f(s_0(x), \mathbf{y}) &= s_{h_0(x, \mathbf{y})}(f(x, \mathbf{y})), \text{ if } x \neq 0 \\ f(s_1(x), \mathbf{y}) &= s_{h_1(x, \mathbf{y})}(f(x, \mathbf{y})). \end{aligned}$$

This scheme is written in the abbreviated form

$$\begin{aligned} f(0, \mathbf{y}) &= g(\mathbf{y}) \\ f(s_i(x), \mathbf{y}) &= s_{h_i(x, \mathbf{y})}(f(x, \mathbf{y})). \end{aligned}$$

Definition 9. The function $\text{BIT}(i, x) = \text{MOD}2(\lfloor \frac{x}{2^i} \rfloor)$ yields the coefficient of 2^i in the binary representation of x ; $\text{MOD}2(x) = x - 2 \cdot \lfloor \frac{x}{2} \rfloor$. The algebra A_0 is defined to be

$$[0, I, s_0, s_1, \text{BIT}, |x|, \#; \text{COMP, CRN}].$$

The algebra $A_0(f)$ is defined to be

$$[0, I, s_0, s_1, \text{BIT}, |x|, \#, f; \text{COMP, CRN}].$$

Definition 10. The function f is defined by *weak bounded recursion on notation* (WBRN) from g, h, k if

$$\begin{aligned} F(0, \mathbf{x}) &= g(\mathbf{x}) \\ F(s_0(n), \mathbf{x}) &= h_0(n, \mathbf{x}, F(n, \mathbf{x})), \text{ if } n \neq 0 \\ F(s_1(n), \mathbf{x}) &= h_1(n, \mathbf{x}, F(n, \mathbf{x})) \\ f(n, \mathbf{x}) &= F(|n|, \mathbf{x}) \end{aligned}$$

provided that $F(n, \mathbf{x}) \leq k(n, \mathbf{x})$ holds for all n, \mathbf{x} . The algebra A is defined to be

$$[0, I, s_0, s_1, \text{BIT}, |x|, \#; \text{COMP, CRN, WBRN}].$$

The algebra $A(f)$ is defined to be

$$[0, I, s_0, s_1, \text{BIT}, |x|, \#, f; \text{COMP, CRN, WBRN}].$$

Part (a) of the following theorem appears in [8]. Part (b) is straightforward from the techniques there developed.

Theorem 11 (P. Clote [8]). (a) $\mathcal{FNC} = A$.
(b) For all $L \subseteq \mathbf{N}$, $\mathcal{FNC}^L = A(c_L)$.

2 Main results

Lemma 12. *A is closed under wsum.*

Proof Let $f(x, \mathbf{y}) = \sum_{i \leq |x|} g(i, \mathbf{y})$, where $g \in A$. We must show that $f(x, \mathbf{y}) \in A$. The idea is to formalize using CRN and WBRN the “addition tree”, a binary tree where values $g(i, \mathbf{y})$ are placed at the leaves and internal nodes have as values the sum of their two children. To do so, we proceed as follows.

In [9] explicit definitions of a number of functions were given in the algebra A_0 . Such functions included $+$, \div , rev , where

$$x \div y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{else} \end{cases}$$

and $rev(x) = y$ holds if the binary representation of y is the reverse of the binary representation of x . In A_0 , define the *bounded exponential* function

$$(2) \quad Exp(a, b) = 2^{\min(a, |b|)}$$

by

$$Exp(x, y) = \lfloor rev(exp0(x, y, s_1(y))) / 2 \rfloor$$

where the auxiliary function $exp0$ is defined by

$$\begin{aligned} exp0(x, y, 0) &= 1 \\ exp0(x, y, s_i(z)) &= \begin{cases} s_1(exp0(x, y, z)) & \text{if } |z| = x \leq |y| \vee |z| = |y| \leq x \\ s_0(exp0(x, y, z)) & \text{else.} \end{cases} \end{aligned}$$

Define functions $\ell(x)$ [resp. $n(x)$] whose value is the number of *leaves* [resp. *nodes*] of a full binary tree T_x which will be associated with x :

$$(3) \quad \ell(x) = Exp(|x|, 2 \cdot |x|)$$

$$(4) \quad n(x) = 2 \cdot \ell(x) - 1.$$

Thus $1 \leq \ell(x) = 2^{\|x\|} \leq 2 \cdot |x|$ is a power 2 and

$$\begin{aligned} n(x) &= 2 \cdot \ell(x) - 1 < 2 \cdot 2^{\|x\|} \leq 4 \cdot |x| \\ 2^{n(x)} &\leq 2^{4 \cdot |x|} \leq (2^{|x|})^4 \leq (2 \cdot x)^4. \end{aligned}$$

Since $Exp(n(x), (2 \cdot x)^4) = 2^{n(x)}$, we can freely use $2^{n(x)}$ in defining functions in A_0 and A .

Let T_x be the full binary tree having $\ell(x)$ many leaves and altogether $n(x)$ many nodes, whose nodes are labeled as follows: leaves are labeled $0, \dots, \ell(x) - 1$

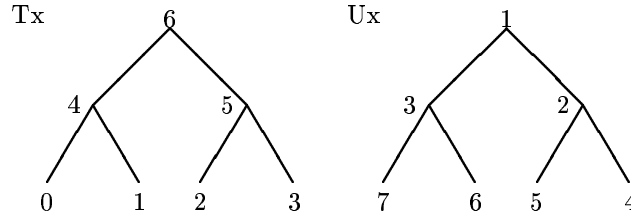
from left to right, then parents of leaves are labeled $\ell(x), \dots, \ell(x) + \frac{\ell(x)}{2} - 1$ from left to right, etc.

More formally, define

$$\begin{aligned} LC(i, x) &= n(x) - [2 \cdot (n(x) - i) - 1] \\ RC(i, x) &= n(x) - [2 \cdot (n(x) - i)]. \end{aligned}$$

Then $L(i, x)$ [resp. $R(i, x)$] is the left [resp. right] child of i in T_x , provided that $\ell(x) \leq i < n(x)$.

Using the scheme WBRN we will formalize a computation on the tree T_x . Though not used in the formal definition, it is helpful to compare the label of a node in T_x with the label of the same node in the tree U_x , defined as follows. Let U_x be the full binary tree having $\ell(x)$ many leaves, and altogether $n(x)$ many nodes, where the root is labeled 1 and for all internal nodes labeled by ν , the right child is labeled by $2 \cdot \nu$ and the left child is labeled by $2 \cdot \nu + 1$. For example, if $x = 7$ (or $4 \leq x \leq 7$) then $\ell(x) = 2^2 = 4$, $n(x) = 7$, and T_x, U_x are given as follows.



Define

$$f(i, x, \mathbf{y}) = \begin{cases} g(i, \mathbf{y}) & \text{if } i \leq |x| \\ 0 & \text{if } |x| < i < \ell(x) \\ f(LC(i, x), x, \mathbf{y}) + f(RC(i, x), x, \mathbf{y}) & \text{if } \ell(x) \leq i < n(x) \\ 0 & \text{if } n(x) \leq i \end{cases}$$

It is easily verified that

$$(5) \quad f(n(x) - 1, x, \mathbf{y}) = \sum_{i=0}^{|x|} g(i, \mathbf{y})$$

since f assigns values $g(0, \mathbf{y}), \dots, g(|x|, \mathbf{y})$ to the leaves of T_x and then computes level-by-level the pairwise sums so that $\sum_{i=0}^{|x|} g(i, \mathbf{y})$ is assigned at the root $n(x) - 1$ of T_x .

Define a pairing function

$$pair(x, y) = 2^{\max(|x|, |y|)+1} \cdot (2^{\max(|x|, |y|)} + y) + (2^{\max(|x|, |y|)} + x).$$

and its projections

$$\begin{aligned} left(pair(x, y)) &= x \\ right(pair(x, y)) &= y \end{aligned}$$

Following [9], to encode the sequence (t_1, \dots, t_n) , let m be the least power of 2 greater than or equal to $|t_i| + 1$, for $i \leq n$. Then define the sequence number

$$\langle t_1, \dots, t_n \rangle$$

encoding the sequence to be $pair(t, 2^m)$, where

$$t = \sum_{i=1}^n (2^{m-1} + t_i) \cdot 2^{m \cdot (i-1)}.$$

If t encodes a sequence $\langle t_1, \dots, t_n \rangle$ of length n , then define the β function by $\beta(t, 0) = n$ and for $1 \leq i \leq n$, $\beta(t, i) = t_i$. On p. 166 of [3], the functions $pair$, $left$, $right$, β were shown to belong to A_0 . (For a systematic presentation of details, see [4].)

It is then straightforward to show that a sequence concatenation function \frown exists in A_0 for which if $s = \langle s_1, \dots, s_n \rangle$ and $t = \langle t_1, \dots, t_m \rangle$ then $s \frown t = \langle s_1, \dots, s_n, t_1, \dots, t_m \rangle$. In [9] it is shown that the function $MSP(x, y) = \lfloor x/2^{|y|} \rfloor$ belongs to A_0 .

Now define $F(z, x, \mathbf{y})$ by

$$\begin{aligned} F(0, x, \mathbf{y}) &= \langle \quad \rangle \\ F(s_i(z), x, \mathbf{y}) &= H(z, x, \mathbf{y}, F(z, x, \mathbf{y})) \end{aligned}$$

where $H(z, x, \mathbf{y}, u) = F(z, x, \mathbf{y}) \frown \langle h(z, x, \mathbf{y}) \rangle$ and

$$h(z, x, \mathbf{y}) = \begin{cases} g(|z|, \mathbf{y}) & \text{if } |z| < |x| \\ 0 & \text{if } |x| \leq |z| < \ell(x) \\ \beta(F(z, x, \mathbf{y}), LC(|z|, x)) + \beta(F(z, x, \mathbf{y}), RC(|z|, x)) & \text{if } \ell(x) \leq |z| < n(x) \\ 0 & \text{else.} \end{cases}$$

Thus $F(z, x, \mathbf{y}) = \langle f(0, x, \mathbf{y}), \dots, f(|z|-1, x, \mathbf{y}) \rangle$. Define $G(z, x, \mathbf{y}) = F(|z|, x, \mathbf{y})$. In [7] it was shown that if $g \in A_0$, then the *maximum* function

$$m_g(x, \mathbf{y}) = \max\{g(i, \mathbf{y}) : i \leq |x|\}$$

belongs to A_0 . It follows that if $g \in A$, then $m_g \in A$. Hence,

$$\begin{aligned} \sum_{i=0}^{|x|} g(i, \mathbf{y}) &\leq m_g(x, \mathbf{y}) \cdot (|x| + 1) \\ &\leq m_g(x, \mathbf{y}) \# (2 \cdot x) \end{aligned}$$

and so $\sum_{i=0}^{|x|} g(i, \mathbf{y})$ is bounded by a function in A . Thus G is defined by WBRN from functions in A and hence belongs to A . Finally, define K by

$$\begin{aligned} K(x, \mathbf{y}) &= G(2^{n(x)} - 1, x, \mathbf{y}) \\ &= F(n(x), x, \mathbf{y}) \\ &= \langle f(0, x, \mathbf{y}), \dots, f(n(x) - 1, x, \mathbf{y}) \rangle. \end{aligned}$$

It follows from (5) that

$$\beta(K(x, \mathbf{y}), n(x)) = \sum_{i=0}^{|x|} g(i, \mathbf{y})$$

and hence A is closed under wsum. \square

Theorem 13. $K \subseteq A$. For any f , $K(f) \subseteq A(f)$.

Proof By replacing sum by product, the proof of the previous lemma can immediately be modified to yield that A is closed under wprod.¹² The initial functions $0, I, s_0, s_1, +, \times$ of K all belong to A — the first three belong to A by definition; in [9], it was shown that $+ \in A_0 \subseteq A$; \times is known to belong to NC (even NC^1), hence by Theorem 11 belongs to A . The algebra A is closed under composition, and by the previous lemma, under wsum and wprod. It follows that $K \subseteq A$. The same proof, in the presence of an additional initial function f , yields the inclusion $K(f) \subseteq A(f)$. \square

In [24], C. Wilson relativized the bounded fan-in boolean circuit model to allow oracle gates (see section 1.2), and for this model constructed an oracle L for which the class NC^L is properly contained in polynomial time P^L . Wilson's circuit families were defined to be logspace uniform (logtime uniformity may be assumed), where the logspace machine outputting the blueprint of the circuit does not have access to the oracle. Oracle calls cost $\lfloor \log_2 k \rfloor$ depth, where k is the size of the query x .

Theorem 14 C. Wilson [24]. *There is an oracle $L \subseteq \mathbf{N}$, for which $NC^L \subset P^L$.*

It easily follows from this theorem that there exists $L \subseteq \mathbf{N}$, such that $FNC^L \subset FP^L$.

Definition 15. For $L \subseteq \mathbf{N}$, let f_L be defined by

$$f_L(x) = \begin{cases} 2x & \text{if } x \notin L \\ 2x + 1 & \text{if } x \in L. \end{cases}$$

$$c_L(x) = \begin{cases} 0 & \text{if } f_L(x) = 2x \\ 1 & \text{if } f_L(x) = 2x + 1 \end{cases} \quad f_L(x) = \begin{cases} s_0(x) & \text{if } c_L(x) = 0 \\ s_1(x) & \text{if } c_L(x) = 1 \end{cases}$$

It easily follows from this observation that for any $L \subseteq \mathbf{N}$, $A(f_L) = A(c_L)$ and $FP^L = \mathcal{L}(f_L) = \mathcal{L}(c_L)$. Let L be as in Theorem 14. Putting everything together, we have the following theorem.

¹² In showing that G is definable by wbrn, note that

$$\prod_{i=0}^{|x|} g(i, \mathbf{y}) \leq m_g(x, \mathbf{y})^{|x|+1} \leq m(x, \mathbf{y}) \#(2 \cdot x).$$

Theorem 16. *There exists a strictly increasing function f such that $K(f)$ is properly contained in $\mathcal{L}(f)$.*

Proof Let $f = f_L$, where L is the oracle from Theorem 14. Then

$$K(f_L) \subseteq A(f_L) = A(c_L) = \mathcal{FNC}^L \subset \mathcal{FPL} = \mathcal{L}(c_L) = \mathcal{L}(f_L).$$

□

3 Complexity classes contained in K

In [2], TC^0 is defined as the class of logtime uniform, constant depth, polynomial size threshold circuits.

Theorem 17. $TC^0 \subseteq K$.

Proof By [7],

$$TC^0 = [0, I, s_0, s_1, |x|, \text{BIT}, \#, \times; \text{COMP}, \text{CRN}].$$

By definition,

$$K = [0, I, s_0, s_1, +, \div, \times, \lfloor x/y \rfloor; \text{COMP}, \text{WSUM}, \text{WPROD}].$$

We begin by showing that the initial functions of TC^0 can be defined in K .

$$\begin{aligned} |x| &= \left(\sum_{i \leq |x|} 1 \right) \div 1 \\ 2^{|x|} &= \left\lfloor \frac{\prod_{i \leq |x|} 2}{2} \right\rfloor \\ \overline{sg}(x) &= 1 \div x = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{else} \end{cases} \\ sg(x) &= 1 \div \overline{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{else} \end{cases} \\ c_{\leq}(x, y) &= sg(x \div y) = \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{else} \end{cases} \\ cond(x, y, z) &= \overline{sg}(x) \cdot y + sg(x) \cdot z = \begin{cases} y & \text{if } x = 0 \\ z & \text{else} \end{cases} \\ \min(x, y) &= cond(x \div y, x, y). \end{aligned}$$

From these functions, definitions by case are admissible in K . Definitions of the form $f(x, \mathbf{y}) = \sum_{i \leq |x|} h(i, x, \mathbf{y})$ and $f(x, \mathbf{y}) = \prod_{i \leq |x|} h(i, x, \mathbf{y})$ are allowed by WSUM and WPROD. Define

$$Exp(a, b) = 2^{\min(a, |b|)} = \prod_{i \leq |b|} g(i, a, b)$$

where

$$g(i, a, b) = \begin{cases} 2 & \text{if } i < \min(a, |b|) \\ 1 & \text{else.} \end{cases}$$

Further define

$$\begin{aligned} MSP(x, i) &= \lfloor \frac{x}{Exp(i, x)} \rfloor \\ \text{BIT}(i, x) &= MSP(x, i) \div 2 \cdot \lfloor \frac{MSP(x, i)}{2} \rfloor \\ x \# y &= \lfloor \frac{\prod_{i \leq |y|} 2^{|x|}}{2^{|x|}} \rfloor = 2^{|x| \cdot |y|} \end{aligned}$$

Thus all the initial functions of TC^0 belong to K .

To show closure of K under CRN, we first show closure under sharply bounded quantifiers and sharply bounded μ -operator. Suppose that $R(x, \mathbf{z})$ is a relation whose characteristic function belongs to K . Then the characteristic function of $(\exists x \leq |y|)R(x, \mathbf{z})$ [resp. $(\forall x \leq |y|)R(x, \mathbf{z})$] is

$$sg(\sum_{x \leq |y|} c_R(x, \mathbf{z})) \quad [\text{resp. } \overline{sg}(\sum_{x \leq |y|} \overline{sg}(c_R(x, \mathbf{z})))].$$

Define $(\mu i < |y|)R(i, \mathbf{z})$ to be the least $i < |y|$ satisfying $R(i, \mathbf{z})$, if such exists, and otherwise 0. Then $(\mu i < |y|)R(i, \mathbf{z})$ can be defined in K by $\sum_{i \leq |y|} g(i, y, \mathbf{z})$, where

$$g(i, y, \mathbf{z}) = \begin{cases} 1 & \text{if } (\forall j \leq |y|)(j \leq i \rightarrow \neg R(j, \mathbf{z})) \\ 0 & \text{else.} \end{cases}$$

Now suppose that the function k is defined from $h_0, h_1 \in K$ by a simple restricted version of CRN as follows.

$$(6) \quad \begin{aligned} k(0, \mathbf{y}) &= 1 \\ k(s_0(x), \mathbf{y}) &= s_{h_0(x, \mathbf{y})}(k(x, \mathbf{y})) \\ k(s_1(x), \mathbf{y}) &= s_{h_1(x, \mathbf{y})}(k(x, \mathbf{y})). \end{aligned}$$

Then k is definable in K by

$$k(x, \mathbf{y}) = \begin{cases} 1 & \text{if } x = 0 \\ (\mu z < |4 \cdot x|)[\phi(x, \mathbf{y}, z)] & \text{else} \end{cases}$$

where $\phi(x, \mathbf{y}, z)$ is

$$[|z| = |x| + 1 \wedge (\forall i < |x|)(\text{BIT}(i, z) = h_{\text{BIT}(i, x)}(MSP(x, i + 1), \mathbf{y}))].$$

Suppose that f is defined from $g, h_0, h_1 \in K$ by CRN:

$$\begin{aligned} f(0, \mathbf{y}) &= g(\mathbf{y}) \\ f(s_0(x), \mathbf{y}) &= s_{h_0(x, \mathbf{y})}(f(x, \mathbf{y})) \\ f(s_1(x), \mathbf{y}) &= s_{h_1(x, \mathbf{y})}(f(x, \mathbf{y})). \end{aligned}$$

Let k be defined from h_0, h_1 by the simpler version (6) of CRN. We have that $k \in K$ and

$$f(x, \mathbf{y}) = \begin{cases} 2^{|x|} \div k(x, \mathbf{y}) & \text{if } g(\mathbf{y}) = 0 \\ (g(\mathbf{y}) \div 1) \cdot 2^{|x|} + k(x, \mathbf{y}) & \text{else.} \end{cases}$$

Thus K contains all the initial functions of TC^0 and is closed under the operations COMP and CRN, hence $TC^0 \subseteq K$. \square

Remark. Since modular counting

$$f(x, \mathbf{y}, m) = \left(\sum_{i \leq |x|} g(i, \mathbf{y}) \right) \bmod m$$

is definable in K , it follows that $ACC \subseteq K$ (see [2] for definition of ACC). Moreover, note that the majority quantifier of [2] is directly definable by $c_{\leq} \left(\sum_{i \leq |x|} g(i), \frac{x}{2} \right)$.

Question 18. (i) $ALOGTIME \subseteq K$?

(ii) $FLOGSPACE \subseteq K$?

(iii) To what complexity class does K correspond?¹³

By [3, 7], $ALOGTIME$ and $FLOGSPACE$ are characterized by function algebras involving respectively k -bounded recursion on notation

$$\begin{aligned} f(0, \mathbf{y}) &= g(\mathbf{y}) \\ f(s_0(x), \mathbf{y}) &= h_0(x, \mathbf{y}, f(x, \mathbf{y})) \\ f(s_1(x), \mathbf{y}) &= h_1(x, \mathbf{y}, f(x, \mathbf{y})) \end{aligned}$$

where $f(x, \mathbf{y})$ is bounded by the constant k , and *length bounded recursion on notation*

$$\begin{aligned} f(0, \mathbf{y}) &= g(\mathbf{y}) \\ f(s_0(x), \mathbf{y}) &= h_0(x, \mathbf{y}, f(x, \mathbf{y})) \\ f(s_1(x), \mathbf{y}) &= h_1(x, \mathbf{y}, f(x, \mathbf{y})) \end{aligned}$$

where $f(x, \mathbf{y}) \leq |k(x, \mathbf{y})|$. It does not seem clear how to go about defining either of these schemes using WSUM and WPROD.

¹³ H.-J. Burtschick (personal correspondence) has suggested a possible relation to uniform, polysize arithmetic circuits.

4 Concluding remarks

In this paper, we have used Wilson's oracle separation of NC from P in order to refute Claim II(b) and cast doubt on Claim I. The area of higher type computational complexity abounds in open problems. Unlike the case where function classes \mathcal{FNC} , \mathcal{FAC}^k , FP , etc. have functions of the same (polynomial) growth rate, it may be that growth rates may distinguish between type 2 complexity classes. In particular, it would be interesting to know the answers of the questions below.

For economy of space, we refer the reader to [6] for definitions of the functional complexity classes \mathcal{A}_0 , \mathcal{A} which are respectively the type 2 analogues of A_0 , A , as BFF below is the type 2 analogue of FP .

Definition 19 Townsend [21]. F is defined from H, G_1, \dots, G_m by functional composition if for all \mathbf{f}, \mathbf{x} ,

$$F(\mathbf{f}, \mathbf{x}) = H(\mathbf{f}, G_1(\mathbf{f}, \mathbf{x}), \dots, G_m(\mathbf{f}, \mathbf{x}), \mathbf{x}).$$

F is defined from G by expansion if for all $\mathbf{f}, \mathbf{g}, \mathbf{x}, \mathbf{y}$,

$$F(\mathbf{f}, \mathbf{g}, \mathbf{x}, \mathbf{y}) = G(\mathbf{f}, \mathbf{x}).$$

F is defined from G, G_1, \dots, G_m by functional substitution if for all \mathbf{f}, \mathbf{x} ,

$$F(\mathbf{f}, \mathbf{x}) = H(\mathbf{f}, \lambda y.G_1(\mathbf{f}, \mathbf{x}, y), \dots, \lambda y.G_m(\mathbf{f}, \mathbf{x}, y), \mathbf{x}).$$

F is defined from G, H, K by limited recursion on notation (LRN) if for all $\mathbf{f}, \mathbf{x}, y$,

$$\begin{aligned} F(\mathbf{f}, \mathbf{x}, 0) &= G(\mathbf{f}, \mathbf{x}) \\ F(\mathbf{f}, \mathbf{x}, y) &= H(\mathbf{f}, \mathbf{x}, y, F(\mathbf{f}, \mathbf{x}, \lfloor \frac{y}{2} \rfloor)), \text{ if } y \neq 0. \end{aligned}$$

provided that $F(\mathbf{f}, \mathbf{x}, y) < K(\mathbf{f}, \mathbf{x}, y)$ holds for all $\mathbf{f}, \mathbf{x}, y$.

The class of basic feasible functionals BFF is the smallest class of type 2 functionals containing $0, s_0, s_1, i_k^n, \#, Ap$ (defined by $Ap(f, x) = f(x)$) and closed under functional composition, expansion, functional substitution, and LRN.

Define the type 2 analogue \mathcal{K} of Constable's K as follows. The functional $F(x, \mathbf{y}, \mathbf{f})$ is defined by $WSUM$ [resp. $WPROD$] from G if $F(x, \mathbf{y}, \mathbf{f})$ equals

$$\sum_{i=0}^{|x|} G(i, \mathbf{y}, \mathbf{f}) \quad [\text{resp. } \prod_{i=0}^{|x|} G(i, \mathbf{y}, \mathbf{f})]$$

Definition 20. The class \mathcal{K} is the smallest class of type 2 functionals containing $0, s_0, s_1, i_k^n, +, \div, \times, \lfloor x/y \rfloor, Ap$ and closed under functional composition, expansion, functional substitution, $WSUM$ and $WPROD$.

Recall the following result from [6].

Theorem 21. $\mathcal{A}_0 \subset \mathcal{A} \subset BFF$.

It follows from this paper that $\mathcal{A}_0 \subset \mathcal{K} \subseteq \mathcal{A} \subset BFF$.

Question 22. Does there exist a function f , such that $K(f) \subset A(f) \subset FP(f)$? If so, can f be chosen to be of polynomial growth rate? Is \mathcal{K} properly contained in \mathcal{A} ?

For classes \mathcal{F} , \mathcal{G} of type 2 functionals, let's say that \mathcal{G} *majorizes* \mathcal{F} if for every functional $F \in \mathcal{F}$ there exists $G \in \mathcal{G}$ such that

$$(\forall \mathbf{f}, \mathbf{x})[F(\mathbf{f}, \mathbf{x}) \leq G(\mathbf{f}, \mathbf{x})].$$

Write $\mathcal{F} \leq \mathcal{G}$ when \mathcal{G} majorizes \mathcal{F} , and $\mathcal{F} < \mathcal{G}$ when $\mathcal{F} \leq \mathcal{G}$ but not $\mathcal{G} \leq \mathcal{F}$.

Question 23. Is it the case that $\mathcal{A}_0 < \mathcal{K} < \mathcal{A} < BFF$?

I would like to thank Chris Wilson for email correspondence and for sending a copy of [24].

References

1. B. Allen. Arithmetizing uniform NC . *Annals of Pure and Applied Logic*, 53(1):1–50, 1991.
2. D. Mix Barrington, N. Immerman, and H. Straubing. On uniformity in NC^1 . *Journal of Computer and System Science*, 41(3):274–306, 1990.
3. P. Clote. Polynomial size frege proofs of certain combinatorial principles. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory and Computational Complexity*, pages 162 – 184. Oxford University Press, 1993.
4. P. Clote. Computation models and function algebras. In E. Griffor, editor, *Handbook of Recursion Theory*. in preparation.
5. P. Clote, B. Kapron, and A. Ignjatovic. Parallel computable higher type functionals. Technical Report BCCS-94-04, Department of Computer Science, Boston College, June 1994.
6. P. Clote, B. Kapron, and A. Ignjatovic. Parallel computable higher type functionals. In *Proceedings of IEEE 34th Annual Symposium on Foundations of Computer Science*, Nov 3–5, 1993. Palo Alto CA. pp. 72–83.
7. P. Clote and G. Takeuti. First order bounded arithmetic and small boolean circuit complexity classes. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 154–218. Birkhäuser Boston Inc., 1995.
8. Peter Clote and Evangelos Kranakis. Boolean functions, invariance groups and parallel complexity. *SIAM J. Comput.* 20:553-590, 1991.
9. P.G. Clote. Sequential, machine-independent characterizations of the parallel complexity classes $ALOGTIME$, AC^k , NC^k and NC . In P.J. Scott S.R. Buss, editor, *Feasible Mathematics*, pages 49–70. Birkhäuser, 1990.
10. A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science II*, pages 24–30. North-Holland, 1965.
11. R. Constable. Type 2 computational complexity. In *5th Annual ACM Symposium on Theory of Computing*, 1973. pp. 108–121.
12. J. Edmonds. Paths, trees, flowers. *Canad. J. Math.*, 17:449–467, 1965.

13. L. Kálmar. Egyszerű példa eldönthetetlen aritmetikai problémára. *Mate és Fizikai Lapok*, 50:1–23, 1943. [In Hungarian with German abstract].
14. B. Kapron and S. Cook. A new characterization of Mehlhorn's poly time functionals. In *Proceedings of IEEE 32th Annual Symposium on Foundations of Computer Science*, pages pp. 342–347, 1991. Journal version in *SIAM J. on Comput.*
15. J.C. Lind. Computing in logarithmic space. Technical Report Project MAC Technical Memorandum 52, Massachusetts Institute of Technology, September 1974.
16. K. Mehlhorn. Polynomial and abstract subrecursive classes. *Journal of Computer and System Science*, 12:147–178, 1976.
17. B. Monien. A recursive and grammatical characterization of exponential time languages. *Theoretical Computer Science*, 3:61–74, 1977.
18. R.W. Ritchie. Classes of predictably computable functions. *Trans. Am. Math. Soc.*, 106:139–173, 1963.
19. H. E. Rose. *Subrecursion: Function and Hierarchies*, volume 9 of *Oxford Logic Guides*. Clarendon Press, Oxford, 1984. 191 pages.
20. D.B. Thompson. Subrecursiveness: machine independent notions of computability in restricted time and storage. *Math. Systems Theory*, 6:3–15, 1972.
21. M. Townsend. Complexity for type-2 relations. *Notre Dame Journal of Formal Logic*, 31:241–262, 1990.
22. K. Wagner. Bounded recursion and complexity classes. In *Lecture Notes in Computer Science*, volume 74, pages 492–498. Springer-Verlag, 1979.
23. K. Wagner and G. Wechsung. *Computational Complexity*. Reidel Publishing Co., 1986.
24. C. Wilson. Relativized nc. *Math. Systems Theory*, 20:13–29, 1987.