A NEW RECURSION-THEORETIC CHARACTERIZATION OF THE POLYTIME FUNCTIONS

Stephen Bellantoni and Stephen Cook

Abstract. We give a recursion-theoretic characterization of FP which describes polynomial time computation independently of any externally imposed resource bounds. In particular, this syntactic characterization avoids the explicit size bounds on recursion (and the initial function $2^{|x|\cdot|y|}$) of Cobham.
Key words. Recursion theory, Cobham, Polynomial Time.
Subject classifications. 68Q15, 03D20, 68Q05.

1. Introduction

Cobham [9] characterized the polytime functions as the least class of functions which includes certain initial functions and which is closed under composition and bounded recursion on notation. His characterization has yielded a number of applications; in particular it serves as the basis for several theories of arithmetic ([10], [5], [13]) which formalize aspects of polytime reasoning.

Although it has been fruitful, an unsatisfying aspect of Cobham's characterization arises in the recursion on binary notation. The recursion operator is a powerful one which, however, can only be applied when an explicit size bound is satisfied by the resulting function. Additionally, an initial function $2^{|x|\cdot|y|}$ is needed solely to provide a large enough bound for making recursive definitions.

Leivant's recent elegant characterization of polynomial time [17] suggests that one might be able to dispense with these features controlling the growth rate of functions. Leivant proves that a function is polytime if and only if it can be proved convergent in the logical system $L_2(QF^+)$ using the function's recursion equations and a "surjective" principle. Here $L_2(QF^+)$ is second order logic with comprehension (i.e., definability of sets) for positive quantifier-free fomulas. The system has the string successor functions built in, but nothing like $2^{|x|\cdot|y|}$; and bounded recursion plays no rôle.

Inspired by Leivant's result and its proof, we present here a direct recursion-theoretic characterization of the polytime functions. The result can be read independently of the results in Leivant's paper [17]. Our closure operations are essentially composition and recursion on notation, syntactically restricted in ways which have no direct connection to bounds on growth rates. All the initial functions are small, their outputs being no longer than the outputs of the string successors.

Our results raise the question of how the type of recursion introduced here, "predicative recursion", can be used to define other complexity classes in a "resource free" manner. More generally we may ask whether Leivant's program, using impredicativity to characterize computational complexity, can be carried out directly in a functional setting absent of logic and provability.

1.1. Background. Immerman [16] and others have characterized polytime relations in a way which is also resource free in the sense that there are no explicit bounds in the defining expressions. However, these are frequently characterizations of relations rather than functions; for example the exponential relation $E(y, x) \equiv (y = 2^x)$ is polytime. While there has been some related work using functions, as in [15], the finite model theory setting has imposed output size bounds *a priori*. We approach a different problem, that of controlling the growth rate of functions without introducing explicit bounds.

The work here also differs substantially from the earlier work of Clote and Takeuti [8] which uses logical sorts to distinguish the size of terms; there the explicit construction 2^t creates a term whose logical sort is one greater than the logical sort of t. For example the Key Lemma in [8], like other contemporary work, uses an explicit bound on the recursively defined function.

Our notion of Predicative Recursion on Notation, which was developed independently, is comparable to Leivant's "tiered recurrence" [18]. The functions defined there are the much smaller class of extended polynomials. Following our work ([3]), Leivant and Marion [19], [20] have expanded the results of [18]. Further results are discussed in the conclusion below.

In the subject of program synthesis and automatic theorem proving, Fegaras, Sheard and Stemple [14] have independently formulated a recursion scheme which seems related to the one below. They do not analyze the complexity of the functions computed using their scheme.

1.2. Motivating Example. As a motivating example, consider the definition of the $\odot$ function (similar to the smash function of Buss and others) in [17]. The definition uses recursion on binary strings as follows: $\odot(w, sv) = \oplus(w, \odot(w, v))$ for $s \in \{0, 1\}$, where $\oplus$ (concatenation) has been defined by recursion *on its*

first input, not its second. Leivant shows that convergence of the $\odot$ function is provable in $L_2(QF^+)$. On the other hand, if we take the natural definition of the exponential function, $2^{sx} = \oplus(2^x, 2^x)$, we see that the recursive term 2^x is substituted in the position which was used for recursively defining $\oplus$. The exponential function is not provable in $L_2(QF^+)$.

This suggests the definition below of a class B of functions. In this class, one does not allow recursive terms to be substituted into a position which was used for an earlier definition by recursion.

2. The class B

Each input to a function in B will be either a "normal" input or a "safe" input; we write the normal inputs to the left and separate them from the safe inputs using a semicolon: $f(\overline{x}; \overline{y})$.

We formulate the result using computation on non-negative integers, but the same proof carries over to computation on binary strings as in Leivant [17], replacing 0 with ϵ. We write $|x|$ for the binary length $\lceil \log_2(x+1) \rceil$ of integer x; and the terms "predecessor" and "successor" refer to binary notation. If $\overline{x}$ is a vector of n integers we write $\overline{|x|}$ for the vector $|x_1|, \ldots, |x_n|$, and we write $\overline{f}(\overline{x})$ for $f_1(\overline{x}), \ldots, f_m(\overline{x})$.

B is the smallest class of functions containing the initial functions `i-v` and closed under `vi`, `vii`:

`i.` **(Constant)** 0 (a zero-ary function).

`ii.` **(Projection)** $\pi^{n,m}_j(x_1 \ldots x_n; x_{n+1} \ldots x_{n+m}) = x_j$, for $1 \leq j \leq n+m$.

`iii.` **(Successors)** $s_i(; a) = 2a + i = ai$, for $i \in \{0, 1\}$.

`iv.` **(Predecessor)** $p(; 0) = 0, \quad p(; ai) = a$.

`v.` **(Conditional)**

$$C(; a, b, c) = \begin{cases} b & \text{if } a \bmod 2 = 0, \\ c & \text{otherwise.} \end{cases}$$

`vi.` **(Predicative Recursion on Notation)** Define the new function f by, for $i \in \{0, 1\}$,

$$\begin{aligned} f(0, \overline{x}; \overline{a}) &= g(\overline{x}; \overline{a}), \\ f(yi, \overline{x}; \overline{a}) &= h_i(y, \overline{x}; \overline{a}, f(y, \overline{x}; \overline{a})) \text{ for } yi \neq 0, \end{aligned}$$

where h_i and g are in B.

vii. **(Safe Composition)** Define the new function f by

$$f(\overline{x};\overline{a}) = h(\overline{r}(\overline{x};);\overline{t}(\overline{x};\overline{a}))$$

where h, $\overline{r}$, and $\overline{t}$ are in B.

The polynomial time functions will be exactly those functions in B which have all normal inputs; i.e. no safe inputs.

Functions in B can perform any (polytime) operation on their normal inputs, but can only apply a restricted set of operations to their safe inputs. In particular, the operations allowed on safe inputs do not increase the length by more than an additive constant. Adding a function to B which operates on safe inputs would in general be more powerful than adding the same function on normal inputs, because of the asymmetry of the composition scheme.

To understand Safe Composition, suppose $f(\overline{x};\overline{a})$ is given by a single expression in $\overline{x}$ and $\overline{a}$. Then $f(\overline{x};\overline{a})$ can be defined by Safe Composition (and Projection) from the function symbols occurring in the expression, provided that each sub-expression $g(\overline{s};\overline{t})$ has no a_i appearing in the terms $\overline{s}$.

Observe that in defining a function by recursion, the recursive value $f(y,\overline{x};\overline{a})$ is substituted into a safe position rather than a normal position of h. The predicative composition scheme ensures that this recursive value will stay in a safe position and will not be copied into a normal position. Intuitively, this means that the depth of sub-recursions which h_i performs on y or $\overline{x}$ cannot depend on the value being recursively computed. This mechanism seems to have the effect of preventing the uncontrolled impredicativity which Leivant discussed as the cause of a blowup in complexity.

In concrete terms, we can think of safe positions as input positions where it is safe to substitute a large value without greatly increasing the output size of the function. In contrast, the output size may increase polynomially in the size of the normal inputs. See Lemma 4.1 below. Intuition relating the safe and normal sorts to computation time is made precise in later work [2]. In philosophical terms, we can think of safe positions as input positions where it is safe to substitute an "impredicatively defined" value.

3. B contains PTIME

To prove that every polytime function is in B, we use the Cobham characterization [9] of P as the least class of functions containing the Constant, Projection, Successor functions, and the smash function $2^{|x|\cdot|y|}$; and closed under ordinary composition $h(\overline{g}(\overline{x}))$ and the following rule:

DEFINITION 3.1. **(Bounded Recursion on Notation)** *If h_i, g, and j are in the class then so is f, where*

$$\begin{aligned} f(0,\overline{x}) &= g(\overline{x}), \\ f(yi,\overline{x}) &= h_i(y,\overline{x},f(y,\overline{x})) \quad \text{for } yi \neq 0. \end{aligned}$$

($i \in \{0,1\}$), provided that $f(y,\overline{x}) \leq j(y,\overline{x})$ for all $y,\overline{x}$.

See Rose [23] for a proof that the Cobham functions are all the polynomial time functions, or see [26] for the same result formulated over binary strings. In particular, for each Cobham function f there is a length-bounding monotone polynomial b_f such that $|f(\overline{x})| \leq b_f(\overline{|x|})$.

To prove that B contains every PTIME function f, we first show how to compute the value of $f(\overline{x})$ assuming that we already have a value w which is big enough. Intuitively speaking, w has to have a length at least as great as the maximum depth of recursion used in computing $f(\overline{x})$.

LEMMA 3.2. *Let f be any polytime function. There is a function f' in B and a monotone polynomial p_f such that $f(\overline{a}) = f'(w;\overline{a})$ for all $\overline{a}$ and all w with $|w| \geq p_f(\overline{|a|})$.*

PROOF. The proof is by induction on the length of the derivation of f as a function in the Cobham class. If f is a constant, projection, or successor function then f' is trivially defined using the Constant, Projection or Successor functions of B. In these cases, let $p_f = 0$.

If f is defined by composition, $f(\overline{x}) = h(\overline{g}(\overline{x}))$, then define f' by $f'(w;\overline{x}) = h'(w;\overline{g}'(w;\overline{x}))$. Since the functions $\overline{g}$ are in the Cobham class, they have length-bounding polynomials $\overline{b_g}$. Define p_f such that

$$p_f(\overline{|x|}) = p_h(\overline{b_g}(\overline{|x|})) + \sum_j p_{g_j}(\overline{|x|}).$$

The induction hypothesis can be applied to h and $\overline{g}$ using the facts implied by $|w| \geq p_f(\overline{|x|})$. The desired properties of f' and p_f then follow easily.

The next case is when f is $2^{|x|\cdot|y|}$. This function has a definition using recursion on notation with length-bounding polynomials $b_g(|x|,|y|) = |x| + |y|$ and $b_f(|x|,|y|) = |x| \cdot |y| + 1$. Namely: $g(0,y) = y$; $g(xi,y) = g(x,y)0$; $f(0,y) = 1$; and $f(xi,y) = g(y,f(x,y))$ (where $xi \neq 0$). In this case one can apply the same method as for bounded recursion on notation.

The difficult case is when $f(y,\overline{x})$ is constructed by bounded recursion on notation as in the Definition above. Then g', h'_0 and h'_1 in B are given by the

induction hypothesis. Of course we cannot define $f'(w; y, \overline{x})$ by recursion on y, since y is not in normal position. Instead we introduce a parameter z, and use recursion on z to simulate recursion on y. The recursion parameter z is initialized to w, and the part of the recursion on $|z|$ from $|w|$ down to $|w| - |y|$ corresponds to recursion on y.

To define f' first define, using Predicative Recursion on Notation and Predicative Composition, the functions

$$\begin{array}{ll} & \\ P(0; b) & = b, \\ P(ai; b) & = p(; P(a; b)), \\ P'(a, b;) & = P(a; b). \end{array}$$

$$\begin{array}{ll} \textsc{par}(; a) & = C(; a, 0, 1), \\ Y(z, w; y) & = P(P'(z, w;); y), \\ I(z, w; y) & = \textsc{par}(; Y(z1, w; y)). \end{array}$$

$$\begin{array}{ll} \vee(0; a) & = \textsc{par}(; a), \\ \vee(xi; a) & = C(; \vee(x; a), \textsc{par}(; P(xi; a)), 1). \end{array}$$

These are explained as follows. The function $P(a; b)$ takes $|a|$ predecessors of b. The function $Y(z, w; y)$ produces an initial segment of y, namely y with $|w| - |z|$ rightmost (low-order) bits deleted. As z varies in length from $|w|$ down to $|w| - |y|$, the output of Y varies from y down to the trivial initial segment, 0. Recursion depths with $|z|$ below $|w| - |y|$ will be irrelevant. The function $I(z, w; y)$ satisfies $Y(zj, w; y) = s_{I(z,w;y)}Y(z, w; y)$, for $zj \neq 0$. At each step of the recursion on z we use the function I to look into y and see which step function h_I should be applied. The function $\vee(x; a)$ computes the logical OR of the rightmost $|x|$ bits of a. To continue formally, define

$$\begin{array}{ll} \hat{h}(w; i, a, \bar{b}, c) & = C(;\ i, h'_0(w; a, \bar{b}, c), h'_1(w; a, \bar{b}, c)), \\ \hat{f}(0, w; y, \overline{x}) & = 0, \\ \hat{f}(zj, w; y, \overline{x}) & = C(;\ \vee(w; Y(z1, w; y)), \\ & \qquad g'(w; \overline{x}), \\ & \qquad \hat{h}(w; I(z, w; y), Y(z, w; y), \overline{x}, \hat{f}(z, w; y, \overline{x})),) \\ f'(w; y, \overline{x}) & = \hat{f}(w, w; y, \overline{x}). \end{array}$$

Since f is in the Cobham class, there is a monotone polynomial b_f such that $|f(y, \overline{x})| \leq b_f(|y|, \overline{|x|})$. Letting $p_h = p_{h_0} + p_{h_1}$, define p_f such that

$$p_f(|y|, \overline{|x|}) \equiv p_h(|y|, \overline{|x|}, b_f(|y|, \overline{|x|})) + p_g(\overline{|x|}) + |y| + 1.$$

Fixing y and $\overline{x}$, let w satisfy $|w| \geq p_f(|y|, \overline{|x|})$. We prove below by induction on $|u|$ that: for $|w| - |y| \leq |u| \leq |w|$, $\hat{f}(u, w; y, \overline{x}) = f(Y(u, w; y), \overline{x})$. Since

$Y(w, w; y) = y$, we have $f'(w; y, \overline{x}) = f(y, \overline{x})$ as desired for the main induction of the lemma.

Consider any u with $|w| - |y| \leq |u| \leq |w|$. Since $|w| - |y| \geq 1$ there is z and $j \in \{0, 1\}$ such that $u = zj$. Also note $|zj| = |z1|$ and $Y(z1, w; y) = Y(zj, w; y)$. Since $|w| \geq |Y(z1, w; y)|$, the expression $\vee(w; Y(z1, w; y))$ is 0 if $Y(z1, w; y) = 0$ and is 1 if $Y(z1, w; y) \neq 0$.

First, if $|u| = |zj| = |w| - |y|$ then $Y(z1, w; y) = 0$. Using the definition of $\hat{f}$ and the main induction hypothesis on g, we immediately have $\hat{f}(zj, w; y, \overline{x}) = g'(w; \overline{x}) = f(0, \overline{x}) = f(Y(zj, w; y), \overline{x})$.

Second, if $|zj| > |w| - |y|$ we can assume $\hat{f}(z, w; y, \overline{x}) = f(Y(z, w; y), \overline{x})$. Using monotonicity of p_f and p_{h_i},

$$\begin{aligned} |w| &\geq p_f(|y|, \overline{|x|}) \\ &\geq p_{h_i}(|Y(z, w; y)|, \overline{|x|}, b_f(|Y(z, w; y)|, \overline{|x|})) \\ &\geq p_{h_i}(|Y(z, w; y)|, \overline{|x|}, |f(Y(z, w; y), \overline{x})|). \end{aligned}$$

This allows us to apply the main induction hypothesis for h_i:

$$\begin{aligned} h_i'(w; Y(z, w; y), \overline{x}, \hat{f}(z, w; y, \overline{x})) &= h_i'(w; Y(z, w; y), \overline{x}, f(Y(z, w; y), \overline{x})) \\ &= h_i(Y(z, w; y), \overline{x}, f(Y(z, w; y), \overline{x})). \end{aligned}$$

The condition $|w| - |y| < |zj| \leq |w|$ implies $y \neq 0$ and $Y(z1, w; y) \neq 0$, so by the definition of $\hat{f}$ and the fact $Y(zj, w; y) = s_{I(z,w;y)}Y(z, w; y)$ we have

$$\begin{aligned} \hat{f}(zj, w; y, \overline{x}) &= h'_{I(z,w;y)}(w; Y(z, w; y), \overline{x}, \hat{f}(z, w; y, \overline{x})) \\ &= h_{I(z,w;y)}(Y(z, w; y), \overline{x}, f(Y(z, w; y), \overline{x})) \\ &= f(Y(zj, w; y), \overline{x}), \end{aligned}$$

as desired. □

THEOREM 3.3. *Let $f(\overline{x})$ be a polytime function. Then $f(\overline{x};\,)$ is in B.*

PROOF. Let p_f and f' be obtained using the preceeding Lemma. We will construct a function $b(\overline{x};\,)$ in B such that $|b(\overline{x};\,)| \geq p_f(\overline{|x|})$. Then setting $f(\overline{x};\,) = f'(b(\overline{x};\,); \overline{x})$ finishes the theorem.

First define concatenation of k strings (in reverse order) using one safe position:

$$\begin{aligned} \oplus^2(0; y) &= y, \\ \oplus^2(xi; y) &= s_i(;\oplus^2(x; y)) \text{ for } xi \neq 0, \\ \oplus^k(x_1, \ldots, x_{k-1}; x_k) &= \oplus^2(x_1; \oplus^{k-1}(x_2, \ldots, x_{k-1}; x_k)). \end{aligned}$$

Next define the "declining smash" function using recursion in the safe position of $\oplus^2$ as follows,

$$\begin{array}{ll} \#(0;\,) & = 0, \\ \#(zi;\,) & = \oplus^2(zi;\#(z;\,)) \ \text{ for } zi \neq 0. \end{array}$$

The length of $\#z$ is $|z|(|z|+1)/2 = \Omega(|z|^2)$.

Let a, c be such that $(\sum_j |x_j|)^a + c \geq p_f(\overline{|x|})$ for all $\overline{x}$. Composing $\#$ onto itself a constant number of times and then concatenating a constant to the end gives a function $b^1(z;\,)$ such that $|b^1(z;\,)| \geq (|z|)^a + c$. Using predicative composition again gives $\oplus(\overline{x};\,) = \oplus^k(x_1 \ldots x_{k-1}; x_k)$ and $b(\overline{x};\,) = b^1(\oplus(\overline{x};\,);\,)$ having length at least $p_f(\overline{|x|})$ as desired. □

4. PTIME contains B

To prove that all functions in B are polytime, we first derive a bound on the length of the computed value. Then it is easy to observe that the Predicative Recursion on Notation operator preserves polytime if the output is length-bounded.

LEMMA 4.1. *Let f be a function in B. There is a monotone polynomial q_f such that $|f(\overline{x};\overline{y})| \leq q_f(\overline{|x|}) + \max_i |y_i|$ for all $\overline{x}$, $\overline{y}$.*

PROOF. The proof is by induction on the derivation of f in B. If f is a Constant, Projection, Successor, Predecessor, or Conditional function, then $|f(\overline{x};\overline{y})| \leq 1 + \sum_i |x_i| + \max_i |y_i|$ and therefore we can just take $q_f(\overline{|x|}) = 1 + \sum_i |x_i|$.

If f is defined by Predicative Recursion on Notation then by the induction hypothesis we have q_g, q_{h_0} and q_{h_1} bounding g, h_0 and h_1 respectively. Letting $q_h = q_{h_0} + q_{h_1}$, we have

$$\begin{array}{ll} |f(0,\overline{x};\overline{y})| & \leq q_g(\overline{|x|}) + \max_i |y_i|, \\ |f(zi,\overline{x};\overline{y})| & \leq q_h(|z|,\overline{|x|}) + \max(\max_i |y_i|, |f(z,\overline{x};\overline{y})|). \end{array}$$

Define q_f such that

$$q_f(|z'|,\overline{|x|}) = |z'| \cdot q_h(|z'|,\overline{|x|}) + q_g(\overline{|x|}).$$

We have trivially $|f(0,\overline{x};\overline{y})| \leq q_f(|0|,\overline{|x|}) + \max_i |y_i|$ by the induction hypothesis on g and monotonicity of q_h. Now assuming $|f(z,\overline{x};\overline{y})| \leq q_f(|z|,\overline{|x|}) + \max_i |y_i|$

and $|zi| \neq 0$ (hence $|zi| = |z| + 1$), we have

$$\begin{aligned}
|f(zi, \overline{x}; \overline{y})| &\leq q_h(|z|, \overline{|x|}) + \max(\max_i |y_i|, |f(z, \overline{x}; \overline{y})|) \\
&\leq q_h(|z|, \overline{|x|}) + \max(\max_i |y_i|, q_f(|z|, \overline{|x|}) + \max_i |y_i|) \\
&\leq q_h(|z|, \overline{|x|}) + q_f(|z|, \overline{|x|}) + \max_i |y_i| \\
&\leq q_h(|z|, \overline{|x|}) + \Big(|z| \cdot q_h(|z|, \overline{|x|}) + q_g(\overline{|x|})\Big) + \max_i |y_i| \\
&\leq |zi| \cdot q_h(|z|, \overline{|x|}) + q_g(\overline{|x|}) + \max_i |y_i| \\
&\leq |zi| \cdot q_h(|zi|, \overline{|x|}) + q_g(\overline{|x|}) + \max_i |y_i| \\
&\leq q_f(|zi|, \overline{|x|}) + \max_i |y_i|.
\end{aligned}$$

Therefore, a simple induction on $|z'|$ shows that $|f(z', \overline{x}; \overline{y})| \leq q_f(|z'|, \overline{|x|}) + \max_i |y_i|$ for all z', as desired.

Finally, if f is defined by Safe Composition then using the induction hypothesis on h, $\overline{r}$, and $\overline{t}$,

$$\begin{aligned}
|f(\overline{x}; \overline{y})| &= |h(\overline{r}(\overline{x};); \overline{t}(\overline{x}; \overline{y}))| \\
&\leq q_h(\overline{|r(\overline{x};)|}) + \max_i |t_i(\overline{x}; \overline{y})| \\
&\leq q_h(\overline{q_r}(\overline{|x|})) + \max_i |t_i(\overline{x}; \overline{y})| \\
&\leq q_h(\overline{q_r}(\overline{|x|})) + \max_i \Big(q_{t_i}(\overline{|x|}) + \max_j |y_j|\Big) \\
&\leq q_h(\overline{q_r}(\overline{|x|})) + \textstyle\sum_i q_{t_i}(\overline{|x|}) + \max_i |y_i|.
\end{aligned}$$

Therefore, choosing q_f such that $q_f(\overline{|x|}) \equiv q_h(\overline{q_r}(\overline{|x|})) + \sum_i q_{t_i}(\overline{|x|})$ finishes the result. □

THEOREM 4.2. *Let $f(\overline{x}; \overline{y})$ be a function in B. Then $f(\overline{x}, \overline{y})$ is polytime.*

PROOF. The initial functions are all clearly computable in polynomial time. For Safe Composition observe that the composition of two polynomial time functions is a polynomial time function.

With regard to Predicative Recursion on Notation, it is well known that a recursion on notation can be executed in polynomial time if the result of the recursion is polynomially length-bounded and the step and base functions are polytime. In our case, the length-bound follows from the preceeding lemma.

Alternatively we can observe that the bounding polynomials of the Lemma are computable in Cobham's class and that therefore every instance of Predicative Recursion on Notation is an occurrence of Bounded Recursion on Notation; this gives a reduction of derivations in B to derivations in the Cobham class. □

5. Further comments

The presentation in this paper has used a single sort of variable and has classified input positions as safe or normal. A similar class of functions can be defined using two sorts of variables, ascribing safe or normal sort to both the inputs and outputs. Safe Composition is replaced by ordinary composition respecting the sorts of the inputs and outputs. The initial functions all take on safe type outputs, and Predicative Recursion on Notation results in a safe-valued function. One further adds the following "Raising" rule: if function $f(\overline{x};)$ of all normal inputs is in the class with safe type output, then the function f^{ν} is in the class with normal type output defined by $f^{\nu}(\overline{x};) = f(\overline{x};)$.

In such a two-sorted presentation we could replace the Predecessor and Conditional functions with the single function $N(x;a) = a \bmod 2^{|x|}$. This gives a class BC which (even ignoring output sorts) differs from B; for example $p(;a)$ is not in BC. But (again ignoring output sorts) BC and B are identical on the subset consisting of functions with all normal inputs. Hence BC is as good as B in characterizing the polynomial time functions. The proof for BC is simpler than the proof for B; however, this is somewhat offset by the fact that N is not a constant-time operation.

If we take as a base the class BC and develop a higher-type class along the lines used for the definition of PV^{ω} terms in [13], we obtain a class apparently equal to the Basic Feasible Functionals discussed in [12]. Now if we add the initial function $\lambda a.|a|$ of type (safe $\rightarrow$ normal), the resulting class is still polytime on its type 1 section yet is able to compute the well-quasi-ordering functional which Cook [11] demonstrated is not Basic Feasible. See [25].

6. Directions for research

Philosophically we think of normal values as those which are known in totality, and safe values as those which are "impredicative" in the sense that their definition refers to members of the set N of integers other than those which are immediately known. As Nelson [21] points out and Leivant [17] emphasises, definitions of N are impredicative because they are justified by inductive means, which presuppose an understanding of N. In other words, the validity of defining a function by recursion requires an impredicatively defined concept, namely N. The current work projects the idea of an impredicatively defined set down onto specific members of the set — relative to the fact that we have certain values given to us in their totality, references to other members of N are references to "impredicative values". They are values which you only know exist because you assume the existence of an impredicativly defined

set, N. We control this impredicativity by isolating such values in safe input positions, and performing only operations on them which are constant-time with respect to their size. Thus we have developed a functional analogue of highly predicative reasoning.

From this viewpoint, it is interesting to investigate the complexity classes obtained with predicative recursion by allowing various amounts of information to cross over from the safe inputs to the normal inputs; or, by allowing various amounts of computation to be performed on the safe inputs. This project could be stated technically as follows: explain, for various syntactically defined and highly restricted function classes R, the effect of adding to B the composition scheme $f(\overline{x};\overline{y}) = h(\overline{r}(\overline{x};\overline{y});\overline{t}(\overline{x};\overline{y}))$ where $r \in R$. As mentioned above, the function $r(;y) = |y|$ is permissible without exceeding polytime.

It appears that the methods of this paper can be used to remove the size bounds from Clote's 'sequential' characterizations [7] of AC^k and NC^k, giving resource-independent characterizations of these classes. An alternative approach to characterizing NC^1 is given by Bloch [4] based on the methods of this paper and the work of Allen [1]. Recently, Leivant and Marion [19], [20] have used a method similar to ours and [18] to characterize the Kalmar elementary functions. The class of logspace-decidable problems, the class of linear space functions, and the class of functions computable in polynomial time using Σ_i^p oracles have now been given similar characterizations ([2]).

The results of this paper can be used [2] to give a new proof of Leivant's result [17] that functions provable in $L_2(QF^+)$ include all the polytime functions.

The results also bear on the system PV ([10], [13]), an equational theory with a function symbol for each polytime function together with defining equations for the function based on Cobham's characterization. When f is introduced by bounded recursion on notation, it is necessary to prove in PV that the bounding inequality is satisfied. This requirement contributes significantly to the complication of developing the theory in PV. An alternative development of PV, based on the theorem here rather than Cobham's theorem, should be simpler.

Finally, a category theory translation of the results in this paper has been given by Otto [22].

Acknowledgments

In an earlier draft of this paper we used the functions $D(a;b) = P(b,a)$ and $E(a;b) = P(b,P(b,a))$ as initial functions. We are grateful to Sam Buss for

pointing out to us that these two can be eliminated in favor of the single predecessor function, p. We would like to thank Toniann Pitassi for her helpful comments. We would like to thank Daniel Leivant for his comments on an earlier draft of this paper.

References

[1] B. Allen, "Arithmetizing Uniform NC", *Annals of Pure and Applied Logic*, p. 1-50, v. 53 (1991).

[2] S. Bellantoni, "Predicative Recursion and Computational Complexity", PhD Thesis, University of Toronto, 1992 (to appear).

[3] S. Bellantoni and S. Cook, "A New Recursion-Theoretic Characterization of the Polytime Functions (Extended Abstract)", in *Proc. 24th Symposium on the Theory Of Computing*, ACM, 1992.

[4] S. Bloch, "Functional Characterizations of Uniform Log-depth and Polylog-depth Circuit Families", in *Proc., Structure in Complexity Theory*, to appear, IEEE, 1992.

[5] S. Buss, "Bounded Arithmetic", Ph.D. Dissertation, Princeton University, 1985; reprinted Bibliopolis, Naples, 1986.

[7] P. Clote, "Sequential, machine-independent characterizations of the parallel complexity classes *AlogTIME*, AC^k, NC^k and NC." *MSI Workshop on Feasible Mathematics*, p. 49-70, Birkhauser, 1989.

[8] P. Clote, G. Takeuti, "'Exponential Time and Bounded Arithmetic", in *Annual Conference on Structure in Complexity Theory*, v. 1, p. 125-143, 1986.

[9] A. Cobham, "The intrinsic computational difficulty of functions". In Y. Bar-Hillel ed., *Proc. of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science*, p. 24-30, North Holland, Amsterdam, 1964.

[10] S. Cook, "Feasibly constructive proofs and the propositional calculus", *Proc. 7th Symposium on the Theory Of Computing*, p. 83-97, ACM, 1975.

[11] S. Cook, "Computability and complexity of higher type functions." *Proc. MSRI Workshop on Logic from Computer Science*, p. 51-72, Y. Moschovakis, ed., Springer Verlag, 1992.

[12] S. Cook and B. Kapron, "Characterizations of the Basic Feasible Functionals of Finite Type", *Feasible Mathematics*, p. 71-95, S. R. Buss and P. J. Scott eds., Birkhauser, 1990.

[13] S. Cook and A. Urquhart, "Functional interpretations of feasibly constructive arithmetic", *Annals of Pure and Applied Logic* (to appear). Extended abstract appears in *Proc. 21st Symposium on the Theory of Computing*, p. 107-112, ACM, 1989.

[14] L. Fegaras, T. Sheard, D. Stemple, "Uniform Traversal Combinators: Definition, Use and Properties", in *11th International Conference on Automated Deduction*, June 1992.

[15] Y. Gurevich, "Algebras of Feasible Functions", *Proc. 24th IEEE Conference on Foundations of Computer Science*, p. 210-214, 1983.

[16] N. Immerman, "Languages That Capture Complexity Classes", *SIAM Journal of Computing*, p. 760-778, v. 16 (1987).

[17] D. Leivant, "A foundational delineation of computational feasibility", in *Proc. Sixth Annual IEEE Symposium on Logic in Computer Science*, p. 2-11, IEEE, 1991.

[18] D. Leivant, "Subrecursion and lambda representation over free algebras (Preliminary summary)", *Feasible Mathematics*, p. 281-292, S. Buss and P. Scott, eds., Birkhauser 1990.

[19] D. Leivant, J-Y. Marion, "Purely applicative characterizations of complexity classes (Extended Abstract)", typescript, Department of Computer Science, Indiana University, April 1992.

[20] D. Leivant, J-Y. Marion, "Capturing poly-time in a typed λ-calculus", typescript, Department of Computer Science, Indiana University, April 1992.

[21] E. Nelson, *Predicative Arithmetic*, Princeton University Press, Princeton, 1986.

[22] J. Otto, "Categorical Characterization of Ptime I", manuscript, Dept. of Mathematics, McGill University, 1991.

[23] H. E. Rose, *Subrecursion: functions and hierarchies*, Oxford Logic Guides 9, Clarendon Press, Oxford, 1984.

[24] R. W. Ritchie, "Classes of Predictably Computable Functions", *Transactions of the American Mathematical Society*, p. 139-173, v. 106 (1963).

[25] A. Seth, "There is no Recursive Axiomatization for Feasible Functionals of Type 2" in *Proc. Seventh Annual IEEE Symposium on Logic in Computer Science*, to appear, IEEE, 1992.

[26] G. J. Tourlakis, *Computability*, Reston, 1984.

Manuscript received 15 October 1991

Stephen J. Bellantoni
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4
sjb@theory.utoronto.ca

Stephen A. Cook
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4
sacook@theory.utoronto.ca