Functions over free algebras definable in the simply typed lambda calculus

Daniel Leivant
Computer Science Department, Indiana University, Bloomington, IN 47405, USA

Dedicated to Corrado Boehm

Abstract

Leivant, D., Functions over free algebras definable in the simply typed lambda calculus, Theoretical Computer Science 121 (1993) 309–321.

We show that a function over a free algebra is definable in the simply typed λ-calculus (modulo the Böhm–Berarducci embedding) iff it is generated by predicative monotonic recurrence. Monotonic recurrence here is the generalization of iteration-with-parameters from $\mathbb{N}$ to arbitrary free algebras, and our predicativity condition uses the notion of tiers introduced by Leivant (1990). In fact, we show that the same functions are generated by tiered monotonic recurrence whether 2 tiers or all finite tiers are used.

1. Introduction

Church showed that a function over $\mathbb{N}$ is definable in the untyped λ-calculus iff it is computable by a Turing machine. The functions over $\mathbb{N}$ definable in the *simply typed* λ-calculus $\mathbf{1}\lambda$ form a dramatically more restricted class: Schwichtenberg [15] and Statman [17] showed that these are exactly the functions definable by composition from addition, multiplication, and the case function **if** $x=0$ **then** y **else** z. Thus, even the predecessor function is not definable. The question of providing a subrecursive characterization of the functions definable in $\mathbf{1}\lambda$ is of interest because it clarifies the computational nature of $\mathbf{1}\lambda$, a calculus which is at the core of typed applicative programs and of denotational semantics. More generally, we wish to provide, given an arbitrary free algebra $\mathbb{A}$, a subrecursive characterization of the functions over $\mathbb{A}$ that are definable in $\mathbf{1}\lambda$. This generalization is especially relevant to computer science,

Correspondence to: D. Leivant, Computer Science Department, Indiana University, Bloomington, IN 47405, USA. Email: leivant@iuvax.cs.indiana.edu.

where the canonical medium of computation is not $\mathbb{N}$ but $\{0, 1\}^*$, which is isomorphic to the algebra of binary words, and where other free algebras, such as the algebra of binary trees, also play an important role.

A subrecursive characterization of the kind above was stated in [9]: a function over a free algebra is definable in $\mathbf{1}\lambda$ (modulo the Böhm–Berarducci embedding) iff it is generated by predicative monotonic recurrence. Monotonic recurrence denies access by the recurrence functions to the components of the recurrence argument, thus forcing the computation to follow the structure of the recurrence argument "without backtracking" (see below). This computation mechanism is related to nonerasing Turing machines [18] and similar monotonic computing devices, and has been shown in finite model theory to underly time-bounds vs. space-bound computational complexity [8, 1, 2]. Monotonic use of information also plays a central role in linear logic, and has been shown to underly the paradoxes of thermodynamics [4]. Our predicativity condition uses a notion of tiers, akin to the ranks used to sort out uses of set-variables in predicative analysis (see e.g. [7]), an idea that goes back to Russell [14]. We present here a complete proof of that characterization. Moreover, we prove that exactly the $\mathbf{1}\lambda$-definable functions are obtained whether predicative monotonic recurrences uses two tiers or all finite tiers.

Our characterization in terms of two tiers is related to a subrecursive characterization discovered independently by Zaionc [19]. However, Zaionc's characterization uses a certain closure-under-substitution condition, which is not a priori effectively verifiable, seems technically less natural than tiering, and lacks the foundational link with predicativity. Moreover, the concept of tiering appears to be a powerful generic tool in relating computational complexity to functional programs and to proof theoretic strength. It was discovered independently (in a slightly different guise) by Bellantoni and Cook [3], who used it to give a subrecursive characterization of the poly-time functions. This work has been further developed in [5, 10]. A result particularly relevant to the present paper is the characterization of poly-time by tiered λ-definability over the algebra of words [13].[1] Put together with the main result of the present paper, the latter result puts into focus two distinct limitations of the computational mechanism of the simply typed λ-calculus: The presence of types confines computation to predicative recurrence, i.e. to polynomial time, whereas the absence of a backtracking mechanism excludes "re-use of space", thereby further restricting computation to *monotonic* predicative recurrence.

2. Monotonic recurrence over free algebras

2.1. *Recurrence over free algebras*

The computation spaces we consider are free algebras, where a free algebra $\mathbb{A}$ is the set of closed terms generated from constructors $c_1 \ldots c_k$ ($k > 0$), with $arity(c_i) = r_i \geqslant 0$.

[1] A related proof theoretic characterization of poly-time based on predicative tiering was obtained in [11].

That is, $\mathbb{A}$ is generated inductively by: if $\xi_1 \ldots \xi_{r_i} \in \mathbb{A}$, then $\boldsymbol{c}_i(\xi_1 \ldots \xi_{r_i}) \in \mathbb{A}$ $(i = 1 \ldots k)$. When $r_i = 0$ we identify $\boldsymbol{c}_i()$ with $\boldsymbol{c}_i$. For example: (1) the algebra $\mathbb{N}$ of natural numbers (i.e. unary numerals) has constructors $\mathbf{0}$ of arity 0 and $\boldsymbol{s}$ of arity 1. (2) The algebra $\mathbb{W}$ of finite words over the alphabet $\{0, 1\}$ has constructors $\boldsymbol{\varepsilon}$ of arity 0, and $\mathbf{0}$ and $\mathbf{1}$ of arity 1. Each term can be identified with a word over $\{0, 1\}$, for example, 011 is identified with $\mathbf{011\varepsilon} = \mathbf{0}(\mathbf{1}(\mathbf{1}(\boldsymbol{\varepsilon})))$. (3) The algebra of unlabeled binary trees has constructors $\boldsymbol{\varepsilon}$ of arity 0 and $\boldsymbol{p}$ of arity 2. (4) The algebra of binary trees with leaves labeled by $\{0, 1\}$ has constructors $\mathbf{0}$ and $\mathbf{1}$ of arity 0 and $\boldsymbol{p}$ of arity 2.

The schema of *recurrence*[2] over the natural numbers is one of the oldest and better known computational schemas. A function f is *defined by recurrence* from functions g_1 and g_2 if

$$f(\mathbf{0}, \vec{x}) = g_1(\vec{x}),$$

$$f(\boldsymbol{s}(n), \vec{x}) = g_2(f(n, \vec{x}), \vec{x}, n)$$

(where $arity(f) = arity(g_1) + 1 = arity(\vec{x}) + 1 = arity(g_2) - 1$). More generally, recurrence over an algebra $\mathbb{A}$ as above has k clauses, one for each constructor:

$$f(\boldsymbol{c}_i(z_1 \ldots z_{r_i}), \vec{x}) = g_i(f_1 \ldots f_{r_i}, \vec{x}, \vec{z}), \quad \text{where } f_j = f(z_j, \vec{x}), \ i = 1 \ldots k.$$

This reflects the inductive definition of $\mathbb{A}$: the values $f(\zeta, \vec{\xi})$ $(\zeta, \vec{\xi} \in \mathbb{A})$ are computed successively in tandem with the stepwise generation of $\zeta \in \mathbb{A}$.

We call the functions g_i above the *recurrence functions*, the first r_i arguments of g_i (as in the template above) the *critical arguments*, and the first argument of f (in the template) the *recurrence argument*. The *PR functions over* $\mathbb{A}$ are the functions defined from the constructors by recurrence and explicit definition (i.e. projections and composition).

We shall be particularly interested in a restricted form of recurrence, *monotonic recurrence*,[3] where the recurrence functions have no direct access to the components of the recurrence argument

$$f(\boldsymbol{c}_i(z_1 \ldots z_{r_i}), \vec{x}) = g_i(f_1 \ldots f_{r_i}, \vec{x}), \quad \text{where } f_j = f(z_j, \vec{x}).$$

For example, the functions *case*, *add*, *mlt*, and *exp* over $\mathbb{N}$ are all defined by monotonic recurrence

$$case(\mathbf{0}, x_0, x_s) = x_0,$$

$$case(sn, x_0, x_s) = x_s;$$

[2] Recurrence over $\mathbb{N}$ was initially known as "recursion", but redubbed in many recent mathematical logic texts as "primitive recursion" so as to avoid confusion with recursion (i.e. fixpoint) constructs in programming languages.

[3] Monotonic recurrence over $\mathbb{N}$ is often dubbed "iteration with parameters", but the term "iteration" would be a misnomer when referring to other algebras.

$$add(sn, x) = s(add(n, x)),$$
$$add(\mathbf{0}, x) = x;$$
$$mlt(\mathbf{0}, x) = \mathbf{0},$$
$$mlt(sn, x) = add(mlt(n, x), x);$$
$$exp(\mathbf{0}) = s\mathbf{0},$$
$$exp(sn) = add(exp(n), exp(n)).$$

But the definition by recurrence of the predecessor function *prd* is *not* monotonic:

$$prd(\mathbf{0}) = \mathbf{0},$$
$$prd(sn) = n.$$

2.2. Tiered recurrence

A spectrum of tiered (or "predicative") recurrences was introduced in [9]; the use of the first two tiers was discovered independently also by Simmons [16] and by Bellantoni and Cook [3]. The conceptual motivation can be summarized, for $\mathbb{A} = \mathbb{W}$, as follows [10–12].[4] The schema of recurrence over $\mathbb{W}$ can be construed as an implicit definition of $\mathbb{W}$ as the domain of the recurrence argument. On the one hand, that domain contains ε and is closed under **0** and **1** and on the other, since f is *determined* by the recurrence, the domain includes nothing else. However, this definition of $\mathbb{W}$ makes sense modulo the assumption that the recurrence functions are already well defined over $\mathbb{W}$, i.e. that $\mathbb{W}$ is already delineated. Thus, this implicit definition of $\mathbb{W}$ is circular. To break the circularity we consider recurrence as a generic template that defines a sequence $\mathbb{W}_0, \mathbb{W}_1 \ldots$ of sets, where $\mathbb{W}_i$ is the set delineated by the use of recurrence over $\mathbb{W}_j$ for $j < i$. Although in retrospect all $\mathbb{W}_i$'s are copies of $\mathbb{W}$, their operational meanings differ. The elements of $\mathbb{W}_0$ are used as discrete, "local", objects, e.g. a word $w \in \mathbb{W}_0$ can be accessed only bitwise, as a "store" of binary values. However, a word $w \in \mathbb{W}_{i+1}$ can be used as "template" for recurrence over $\mathbb{W}_j$ for $j \leqslant i$. Further discussion of predicative recurrence and related issues can be found in [12].

The concept of distinct computational roles for elements of a free algebra $\mathbb{A}$ is captured algebraically as follows. We posit a sequence $\mathbb{A}_0, \mathbb{A}_1 \ldots$ of copies of $\mathbb{A}$, called *tiers*.[5] Each tier $\mathbb{A}_j$ has its own copy $c_i^j : \mathbb{A}_j^{r_i} \to \mathbb{A}_j$ of each constructor c_i of $\mathbb{A}$ (when in no danger of confusion, we drop the tier superscript). We let $\mathbb{A}_*$ be the many-sorted structure with $\mathbb{A}_0, \mathbb{A}_1 \ldots$ as universes, and with the corresponding copies of the constructors. Functions over $\mathbb{A}_*$ are sorted; in particular, function composition must respect sorts. We write $tier(\boldsymbol{t})$ for the tier of (the value of) the term $\boldsymbol{t}$. If f is a function

[4] We take as an example $\mathbb{W}$ rather than $\mathbb{N}$ because the operational meaning of the base tier below is trivial for $\mathbb{N}_0$ but not for $\mathbb{W}_0$.

[5] We avoid phrases such as "level", "order", "rank", or "height", because they are already laden with connotations, some of which might be used in the same context with tiering.

over $\mathbb{A}_*$, i.e. $f: \mathbb{A}_{i_1} \times \cdots \times \mathbb{A}_{i_k} \to \mathbb{A}_j$ for some $i_1, \ldots, i_k, j \geqslant 0$, then we write $tier(f)$ for the tier j of the range of f.

The schema of *tiered monotonic recurrence* (*at tier* j) is then

$$f(c_i^j(z_1 \ldots z_{r_i}), \vec{x}) = g_i(f_1 \ldots f_{r_i}, \vec{x}) \quad \text{with } f_j =_{\text{df}} f(z_j, \vec{x}), \quad i = 1 \ldots k,$$

where $j > tier(f)$.[6] That is, if for $i = 1 \ldots k$ we have $g_i: \mathbb{A}_l^{r_i} \times \mathscr{A} \to \mathbb{A}_l$, where $\mathscr{A} = \mathbb{A}_{t_1} \times \cdots \times \mathbb{A}_{t_n}$ ($n = arity(\vec{x})$), and $l < j$, then $f: \mathbb{A}_j \times \mathscr{A} \to \mathbb{A}_l$ can be defined by the k equations above. The set of functions over $\mathbb{A}_*$ defined by monotonic recurrence of tiers $< t \leqslant \omega$, $MR_t(\mathbb{A}_*)$, is generated from the constructors by explicit (sorted) definitions and j-tiered recurrence with $j < t$. We denote by $MR_t(\mathbb{A})$ the set of functions over $\mathbb{A}$ that are obtained from functions in $MR_t(\mathbb{A}_*)$ by disregarding the distinction between tiers.

Lemma 2.1. *Every* $f \in MR_\omega(\mathbb{A}_*)$ *is constant with respect to arguments of tier* $< tier(f)$.

Proof. By induction on the derivation of $f \in MR_\omega(\mathbb{A}_*)$. The lemma is trivial for constructors and projection functions.

Suppose that f is defined by composition, say $f(\vec{x}) = h(\vec{x}, g(\vec{x}))$. If $tier(g) < tier(h)$ then h is constant with respect to its last argument, and the lemma's statement follows trivially from the induction assumption applied to h. If $tier(g) \geqslant tier(h)$, then any argument x of tier $< tier(f) = tier(h)$ is of tier $< tier(g)$, and so both g and h are constant with respect to x, by induction assumption. Thus, f is constant with respect to x.

Finally, suppose that f is defined by tiered monotonic recurrence,

$$f(c_i^j(z_1 \ldots z_{r_i}), \vec{x}) = g_i(f_1 \ldots f_{r_i}, \vec{x}), \quad i = 1 \ldots k.$$

The recurrence argument does not fall under the lemma's provisions, because $j > tier(f)$ by the tiering condition. By induction assumption, g_i is constant with respect to each argument x of tier $< tier(f) = tier(g_i)$. Thus, by secondary induction on z, it follows that $f(z, \vec{x})$ is constant with respect to each such argument x, for every z. □

2.3. Examples of tiered recurrence

- Let $\mathbb{A}$ be a free algebra as above. For every $j > m \geqslant 0$ we have a tier-reduction function $\rho_{jm}: \mathbb{A}_j \to \mathbb{A}_m$, defined by

$$\rho_{jm}(c_i^j(x_1 \ldots x_{r_i})) = c_i^m(\rho_{jm} x_1 \ldots \rho_{jm} x_{r_i}), \quad i = 1 \ldots k.$$

Thus, objects of upper tiers can be used in lower tiers, in contrast to Lemma 2.1.

[6] We have $tier(f) = tier(g_1) = \cdots = tier(g_k)$ by the definition itself. This condition is slightly different from the condition in [10], where only the critical arguments of the functions g_i are referred to, and the tiering condition is therefore satisfied vacuously if these arguments are absent. This is because [10] refers to models of computation in which, intuitively, bitwise access to data is basic, so that the recurrence argument of functions like *case*, where critical arguments are absent, is functionally innocuous.

- For every $j > i \geqslant 0$ we have an addition function $add_{ji}: \mathbb{N}_j \times \mathbb{N}_i \to \mathbb{N}_i$, defined by

$$add_{ji}(\mathbf{0}^j, x) = x,$$
$$add_{ji}(s^j(n), x) = s^i(add_{ji}(n, x)).$$

- For every i,j,m with $j,m > i \geqslant 0$ we have a multiplication function $mlt_{jmi}: \mathbb{N}_j \times \mathbb{N}_m \to \mathbb{N}_i$, defined by

$$mlt_{jmi}(\mathbf{0}^j, x) = \rho_{mi} x,$$
$$mlt_{jmi}(s^j(n), x) = add_{mi}(x, mlt_{jmi}(n, x)).$$

- For every $j > i \geqslant 0$ we have a discriminator function $case_{ji}: \mathbb{N}_j \times \mathbb{N}_i^2 \to \mathbb{N}_i$, defined by

$$case_{ji}(\mathbf{0}^j, x_0, x_1) = x_0,$$
$$case_{ji}(s^j(n), x_0, x_1) = x_s,$$

- Similar functions can be defined for any free algebra. For example, the additive function on $\mathbb{W}$ is simply concatenation, i.e. $add_{ji}: \mathbb{W}_j \times \mathbb{W}_i \to \mathbb{W}_i$, is defined by

$$add_{ji}(\varepsilon^j, x) = x,$$
$$add_{ji}(\mathbf{0}^j(w), x) = \mathbf{0}^i(add_{ji}(w, x)),$$
$$add_{ji}(\mathbf{1}^j(w), x) = \mathbf{1}^i(add_{ji}(w, x)).$$

A multiplicative function $mlt_{jmi}: \mathbb{W}_j \times \mathbb{W}_m^3 \to \mathbb{W}_i$, for $j, m > i$, is defined by

$$mlt_{jmi}(\varepsilon^j, x_\varepsilon, x_0, z) = \rho_{mi} x_\varepsilon,$$
$$mlt_{jmi}(\mathbf{0}^j(w), x_\varepsilon, x_0, x_1) = add_{mi}(x_0, mlt_{jmi}(w, x_\varepsilon, x_0, x_1)),$$
$$mlt_{jmi}(\mathbf{1}^j(w), x_\varepsilon, x_0, x_1) = add_{mi}(x_1, mlt_{jmi}(w, x_\varepsilon, x_{0,1})).$$

For example, $mlt_{jmi}(\mathbf{011}\varepsilon, x_\varepsilon, x_0, x_1) = x_0 x_1 x_1 x_\varepsilon$ (concatenation).

- Consider the recurrence equations for *exp*: $exp(sn) = g(exp(n))$, where $g(z) = add(z, z)$. The definitions of all tiered addition functions require that the first argument be of higher tier than the second; thus, g cannot be obtained from these functions. Indeed, there is no definition of the exponential function using tiered recurrence and explicit definitions [10].[7]
- Consider the cubic function, $n^3 = mlt(n, mlt(n, n))$. We can stratify this definition by taking the inner multiplication to be mlt_{221} and the outer one to be mlt_{210}, thus defining the cubic function as a function $\mathbb{N}_2 \to \mathbb{N}_0$. However, the cubic function can be defined as a function $\mathbb{N}_1 \to \mathbb{N}_0$, though at the cost of iterating recurrence; in fact, two tiers suffice to generate all tier-recursive functions (over any free algebra) [10].

[7] For *monotonic* recurrence this follows from Theorem 3.6 and the characterization of [15, 17].

3. Lambda definability

3.1. The simply typed λ-calculus

The simply typed λ-calculus, **1λ**, is widely recognized as the core of applicative programming and of programming language semantics. The *types* of **1λ** are generated inductively by: o is a type; if σ and τ are types, then so is $\sigma \to \tau$. The type o is intended to denote a set of basic data objects, and $\sigma \to \tau$ the set of functions from objects of type σ to objects of type τ. The type expressions $o^k \to o$ are abbreviations, defined recursively by $(o^0 \to o) =_{\mathrm{df}} o$, $(o^{k+1} \to o) =_{\mathrm{df}} o \to (o^k \to o)$. More generally, the abbreviation $(\sigma_k \ldots \sigma_1) \to \tau$ is defined by recurrence on k: $() \to \tau =_{\mathrm{df}} \tau$, and $(\sigma_{k+1} \ldots \sigma_1) \to \tau =_{\mathrm{df}} (\sigma_{k+1} \to ((\sigma_k \ldots \sigma_1) \to \tau))$.

For each type τ we posit a denumerable supply of variables of type τ (which we superscript by τ when convenient). The *λ-terms* of **1λ** are defined inductively: Each variable of type τ is also a λ-term of type τ; if E is a λ-term of type σ and x is a variable of type τ, then $\lambda x.E$ is a λ-term of type $\tau \to \sigma$; if E is a λ-term of type $\tau \to \sigma$ and F is a λ-term of type τ then EF is a λ-term of type σ. We write $E_1 E_2 \ldots E_n F$ for $(\ldots((E_1 E_2) E_3) \ldots E_n) F$.[8] We write $[F/x]E$ for the result of substituting F for all free occurrences of the variable x in E; more generally, $[F_1 \ldots F_n / x_1 \ldots x_n]$ is the result of simultaneously substituting $F_1 \ldots F_n$ for all free occurrences of $x_1 \ldots x_n$, respectively. We write $E[\vec{x}]$ for E if all free variables in E are among $\vec{x} = x_1, \ldots, x_n$; assuming a canonical ordering of the variables, we then write $E[F_1 \ldots F_n]$ for $[F_1 \ldots F_n / x_1 \ldots x_n]E$.

The computational meaning of λ-abstraction is conveyed by the *β-reduction rule*, $(\lambda x.E)F \to_\beta [F/x]E$. We write $=_\beta$ for β-equality, i.e. the least equivalence relation containing $\to_\beta$, and we say that a λ-term E is *normal* if $E \to_\beta F$ for no F. It is well known that every sequence of β-reductions in **1λ** terminates (with a normal λ-term); in particular, every λ-term is β-equal to some normal λ-term.

3.2. Lambda representation of data and functions

Böhm and Berarducci [6] insightfully generalized Church's λ-representation of the natural numbers to arbitrary free algebras. Given a free algebra $\mathbb{A}$ as above, let $c_1 \ldots c_k$ be variables, where c_i is of type $\tau_i =_{\mathrm{df}} o^{r_i} \to o$, and write $\vec{c}$ for the sequence $c_1 \ldots c_k$. For each $\xi \in \mathbb{A}$ let $\xi\{\vec{c}\}$ be the same as ξ, but with $\mathbf{c}_i$ replaced by c_i, i.e. we define by recurrence on ξ: $(\mathbf{c}_i(\xi_1 \ldots \xi_{r_i}))\{\vec{c}\} = c_i(\xi_1\{\vec{c}\}) \cdots (\xi_{r_i}\{\vec{c}\})$ $(i = 1 \ldots k)$. More generally, if $\vec{H} = (H_1 \ldots H_k)$, with $type(H_i) = \tau_i$, then we write $\xi\{\vec{H}\}$ for the λ-term defined as above with H_i in place of c_i. Now let $\bar{\xi} =_{\mathrm{df}} \lambda \vec{c}.\xi\{\vec{c}\}$. Note that for every $\xi \in \mathbb{A}$, $\bar{\xi}$ is of type $\alpha =_{\mathrm{df}} ((\tau_1 \ldots \tau_k) \to o)$, and that for $\vec{H}$ as above, $\bar{\xi}\vec{H} =_\beta \xi\{\vec{H}\}$. For example, each $\xi \in \mathbb{N}$ is represented by $\bar{\xi} =_{\mathrm{df}} \lambda s^{o \to o}, z^o.s^{[\xi]}(z)$ (where $f^{[\xi]}$ denotes the ξ'th iterate of the

[8] It is natural to associate multiplication to the right for algebra terms, where higher-order functions are not used, but to the left for λ-terms.

function f), i.e. by Church's ξ'th numeral. Taking $\mathbb{A}=\mathbb{W}$ as another example, the algebra element $\xi=\mathbf{011}\varepsilon$ is represented by $\bar{\xi}=\lambda z^{o\to o}, u^{o\to o}, e^{o}.zuue$.

A function $f\colon \mathbb{A}^n \to \mathbb{A}$ is *defined* in $\mathbf{1}\lambda$ by a λ-term E of type $\alpha^n\to\alpha$, if $E\bar{\xi}_1\dots\bar{\xi}_n =_\beta \overline{f(\xi_1\dots\xi_n)}$ for every $\xi_1\dots\xi_n \in \mathbb{A}$.

3.3. Tiered monotonic recursive functions are definable

Lemma 3.1. *Let $\mathbb{A}$ be a free algebra as above. Suppose*

$$f\colon \mathbb{A}_{j_1}\times\cdots\mathbb{A}_{j_p}\times\mathbb{A}_{l_1}\times\cdots\times\mathbb{A}_{l_q}\to\mathbb{A}_j$$

is in $MR_\omega(\mathbb{A}_*)$, where $j_1\dots j_p\leqslant j$ and $l_1\dots l_q>j$. Then there is a λ-term $F[x_1^o\dots x_p^o, y_1^\alpha\dots y_q^\alpha]$ of type o such that, for every $\xi_1\dots\xi_p$, $\eta_1\dots\eta_q\in\mathbb{A}$,

$$F[\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\bar{\eta}_1\dots\bar{\eta}_q] =_\beta f(\vec{\xi},\vec{\eta})\{\vec{c}\}.$$

Corollary 3.2. *Every function over $\mathbb{A}$ generated by tiered monotonic recurrence is definable in $\mathbf{1}\lambda$.*

Proof. Let f, F be as in the Lemma 3.1. Then f is defined by $\lambda w_1^\alpha\dots w_p^\alpha, y_1^\alpha\dots y_q^\alpha.\lambda\vec{c}.F[w_1\vec{c}\dots w_p\vec{c}, y_1\dots y_q]$. $\square$

Proof of Lemma 3.1. By induction on the derivation of $f\in MR_\omega(\mathbb{A}_*)$. If f is a constructor c_i^j over $\mathbb{A}_j$, then let $F[x_1\dots x_{r_i}] =_{\mathrm{df}} c_i x_1\dots x_{r_i}$.

If f is a projection, then the statement of the lemma is trivial.

Suppose that f is defined by composition. By Lemma 2.1 we may assume that all arguments of f are of tier $\geqslant j$, since other arguments can be instantiated to a constant algebra-term. Thus, without loss of generality, $f\colon \mathbb{A}_j\times\mathbb{A}_{i_1}\times\mathbb{A}_{i_2}\to\mathbb{A}_j$, where $i_2>i_1>j$, and

$$f(x,y_1,y_2)=h(g_1(x,\vec{y}),\dots,g_m(x,\vec{y})),$$

where $tier(g_1)\leqslant\cdots\leqslant tier(g_m)$. Write t_i for $tier(g_i)$. Referring again to Lemma 2.1, we may assume that the arguments of h are all of tier $\geqslant tier(h)=j$. Thus, we may assume, without loss of generality, that there is one g_i for each one of the following conditions: $t_i=j$, $j<t_i<i_1$, $t_i=i_1$, $i_1<t_i<i_2$, $t_i=i_2$, and $i_2<t_i$. By induction assumption there are λ-terms $H[v_1,\dots,v_6]$ and $G_i[x,y_1,y_2]$ $(i=1\dots6)$, such that, for all $\xi,\eta_1\dots\eta_5\in\mathbb{A}$,

$$H[\xi\{\vec{c}\},\bar{\eta}_1,\dots,\bar{\eta}_5] =_\beta h(\xi,\eta_1,\dots,\eta_5)\{\vec{c}\}$$

and, for all $\xi,\eta_1,\eta_2\in\mathbb{A}$,

$$G_i[\xi\{\vec{c}\},\bar{\eta}_1,\bar{\eta}_2] =_\beta g_i(\xi,\eta_1,\eta_2)\{\vec{c}\},\quad i=1,2,$$

$$G_i[\xi\{\vec{c}\},\eta_1\{\vec{c}\},\bar{\eta}_2] =_\beta g_i(\xi,\eta_1,\eta_2)\{\vec{c}\},\quad i=3,4,$$

$$G_i[\xi\{\vec{c}\},\eta_1\{\vec{c}\},\eta_2\{\vec{c}\}] =_\beta g_7(\xi,\eta_1,\eta_2)\{\vec{c}\},\quad i=5,6.$$

Let

$$F[x^o, y_1^\alpha, y_2^\alpha] =_{\mathrm{df}} H[G_1[x, y_1, y_2], \lambda\vec{c}.G_2[x, y_1, y_2], \lambda\vec{c}.G_3[x, y_1\vec{c}, y_2]$$

$$\lambda\vec{c}.G_4[x, y_1\vec{c}, y_2], \lambda\vec{c}.G_5[x, y_1\vec{c}, y_2\vec{c}], \lambda\vec{c}.G_6[x, y_1\vec{c}, y_2\vec{c}]].$$

Then, for $\xi, \eta_1, \eta_2 \in \mathbb{A}$,

$$F[\xi\{\vec{c}\}, \bar{\eta}_1, \bar{\eta}_2] =_\beta H[G_1[\xi\{\vec{c}\}, \bar{\eta}_1, \bar{\eta}_2], \lambda\vec{c}.G_2[\xi\{\vec{c}\}, \bar{\eta}_1, \bar{\eta}_2],$$

$$\lambda\vec{c}.G_3[\xi\{\vec{c}\}, \eta_1\{\vec{c}\}, \bar{\eta}_2], \lambda\vec{c}.G_4[\xi\{\vec{c}\}, \eta_1\{\vec{c}\}, \bar{\eta}_2],$$

$$\lambda\vec{c}.G_5[\xi\{\vec{c}\}, \eta_1\{\vec{c}\}, \eta_2\{\vec{c}\}], \lambda\vec{c}.G_6[\xi\{\vec{c}\}, \eta_1\{\vec{c}\}, \eta_2\{\vec{c}\}]]$$

(by definition of F and β-reductions)

$$=_\beta H[g_1(\xi, \eta_1, \eta_2)\{\vec{c}\}, \overline{g_2(\xi, \eta_1, \eta_2)}, \ldots, \overline{g_6(\xi, \eta_1, \eta_2)}]$$

(by induction assumption for the g_i's)

$$=_\beta h(g_1(\xi, \eta_1, \eta_2), \ldots, g_6(\xi, \eta_1, \eta_2))\{\vec{c}\}$$

(by induction assumption for h)

$$= f(\xi, \eta_1, \eta_2)\{\vec{c}\} \quad \text{(by definition of } f\text{)}.$$

Finally, suppose that f is defined by tiered monotonic recurrence. Without loss of generality,

$$f(c_i^l(u_1 \ldots u_{r_i}), x, y) = g_i(f(u_1, x, y) \ldots f(u_{r_i}, x, y), x, y), \quad i = 1 \ldots k,$$

where $tier(x), tier(f) < l \leqslant tier(y)$. By induction assumption, for each $i = 1 \ldots k$ there is a λ-term $G_i = G_i[z_1 \ldots z_{r_i}, x, y]$ such that, for all $\zeta_1 \ldots \zeta_{r_i}, \xi, \eta \in \mathbb{A}$,

$$G_i[\zeta_1\{\vec{c}\}, \ldots, \zeta_{r_i}\{\vec{c}\}, \xi\{\vec{c}\}, \bar{\eta}] =_\beta g_i(\zeta_1 \ldots \zeta_{r_i}, \xi, \eta)\{\vec{c}\}.$$

For $i = 1 \ldots k$ let $\vec{d}_i$ be a list $d_{i0} \ldots d_{ir_i}$ of fresh variables of type o. Let $G_i^*[x, y]$ abbreviate $\lambda\vec{d}_i.G_i[\vec{d}_i, x^o, y^\alpha]$, and let $\vec{G}^*[x, y]$ stand for the sequence $G_1^*[x, y] \ldots G_k^*[x, y]$. Finally, define

$$F[w^\alpha, x^o, y^\alpha] =_{\mathrm{df}} w\vec{G}^*[x, y].$$

Fix $\xi, \eta \in \mathbb{A}$. We prove, by induction on ζ, that

$$F[\bar{\zeta}, \xi\{\vec{c}\}, \bar{\eta}] = \zeta\{\vec{J}\}$$

$$=_\beta f(\zeta, \xi, \eta)\{\vec{c}\},$$

where $\vec{J}=\vec{G}^*[\xi\{\vec{c}\},\vec{\eta}]$, i.e. $\vec{J}$ is the list of λ-terms $J_1 \ldots J_k$, where $J_i =_{\mathrm{df}} G_i^*[\xi\{\vec{c}\},\vec{\eta}]$. Indeed, if $\zeta = c_i(\zeta_1 \ldots \zeta_{r_i})$ then

$$\begin{aligned}
F[\bar{\zeta},\xi\{\vec{c}\},\vec{\eta}] &\equiv \bar{\zeta}\vec{J}\\
&=_\beta \zeta\{\vec{J}\}\\
&\equiv J_i(\zeta_1\{\vec{J}\})\cdots(\zeta_{r_i}\{\vec{J}\})\\
&=_\beta J_i(f(\zeta_1,\xi,\eta)\{\vec{c}\})\cdots(f(\zeta_{r_i},\xi,\eta)\{\vec{c}\})\\
&\qquad \text{(by induction assumption for the } \zeta_j\text{'s)}\\
&=_\beta G_i[f(\zeta_1,\xi,\eta)\{\vec{c}\},\ldots,f(\zeta_{r_i},\xi,\eta)\{\vec{c}\},\xi\{\vec{c}\},\vec{\eta}]\\
&\qquad \text{(by definition of } J_i \text{ and } \beta\text{-reductions)}\\
&=_\beta g_i(f(\zeta_1,\xi,\eta),\ldots,f(\zeta_{r_i},\xi,\eta),\xi,\eta)\{\vec{c}\}\\
&\qquad \text{(by the main-induction assumption, for } g_i)\\
&= f(\zeta,\xi,\eta)\{\vec{c}\} \quad \text{(by the } i\text{'th recurrence equation for } f). \qquad \square
\end{aligned}$$

3.4. Lambda definable functions are tiered monotonic recursive

Lemma 3.3. *Let $\mathbb{A}$ and $\vec{c}$ be as above. If $F=F[\vec{c}]$ is a normal λ-term of type o, then $F=\xi\{\vec{c}\}$ for some $\xi\in\mathbb{A}$.*

Proof. By induction on F. Since F is normal, it must be of the form $vE_1\ldots E_n$, where v is a variable. Since all variables are among $c_1\ldots c_k$, F is of the form $c_iE_1\ldots E_{r_i}$ for some i $(1\leqslant i\leqslant k)$. By induction assumption each E_j is $\xi_j\{\vec{c}\}$ for some $\xi_j\in\mathbb{A}$ $(j=1\ldots r_i)$. So $F=(c_i(\xi_1\ldots\xi_{r_i}))\{\vec{c}\}$. $\square$

Lemma 3.4. *Suppose that $F[x_1^o\ldots x_p^o, y_1^\alpha\ldots y_q^\alpha,\vec{c}]$ is a normal λ-term of type o, where each y_j occurs once. By Lemma 3.3 there is a function $f:\mathbb{A}^p\times\mathbb{A}^q\to\mathbb{A}$ such that*

$$f(\vec{\xi},\vec{\eta})\{\vec{c}\}=_\beta F[\xi_1\{\vec{c}\},\ldots,\xi_p\{\vec{c}\},\bar{\eta}_1\ldots\bar{\eta}_q,\vec{c}].$$

We have $f\in MR_2(\mathbb{A}_*)$, with $f:\mathbb{A}_0^p\times\mathbb{A}_1^q\to\mathbb{A}_0$.

Corollary 3.5. *Let $\mathbb{A}$ be a free algebra as above. If f is a function over $\mathbb{A}$ definable in* **1**λ, *then $f\in MR_2(\mathbb{A})$.*

Proof. Suppose that f is definable by a (closed) normal λ-term F of type $\alpha^q\to\alpha$, i.e. $(\alpha^q,\tau_1\ldots\tau_k)\to o$. It is easy to see that F must be of the form $\lambda y_1^\alpha\ldots y_q^\alpha.\lambda\vec{c}.F_0[\vec{y},\vec{c}]$. Let $F_0'[z_1\ldots z_n,\vec{c}]$ be like F_0, except that repeated free occurrences of variables y_j are replaced by fresh variables. Define f' by $f'(z_1\ldots z_n)\{\vec{c}\}=_\beta F_0'[\bar{z}_1\ldots\bar{z}_n,\vec{c}]$. By Lemma 3.4 we have $f'\in MR_2(\mathbb{A}_*)$. Since f is defined explicitly from f', we have $f\in MR_2(\mathbb{A})$. $\square$

Proof of Lemma 3.4. By (course-of-value) induction on q, and secondary induction on F. If F is x_i^o, then f is the identity function on $\mathbb{A}_0$. If F is some c_i where $r_i=0$, then f is a constant function. In either case, $f \in MR_2(\mathbb{A}_*)$ trivially. (Note that F cannot be a y_i, since F is of type o.)

If F is an application, then it must be of the form $vH_1 \dots H_m$, where v is a variable, since F is normal. v cannot be one of the x_j's, since these are of type o. Suppose v is c_i. Then $m=r_i$ and $H_1 \dots H_{r_i}$ are all of type 0. By induction assumption there is, for every $j=1 \dots r_i$, a function h_j that corresponds to H_i as in the lemma. Define

$$f(\vec{x},\vec{y})=c_i^0(h_1(\vec{x},\vec{y}),\dots,h_{r_i}(\vec{x},\vec{y})).$$

Then we have

$$F[\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\vec{\eta}_1\dots\vec{\eta}_q] =_\beta c_i H_1[\xi_1\{\vec{c}\}\dots\vec{\eta}_q]\dots H_{r_i}[\xi_1\{\vec{c}\}\dots\vec{\eta}_q]$$

(by definition of F)

$$=_\beta c_i(h_1(\vec{\xi},\vec{\eta})\{\vec{c}\})\cdots(h_{r_i}(\vec{\xi},\vec{\eta})\{\vec{c}\})$$

(by induction assumption for the H_j's)

$$=_\beta f(\vec{\xi},\vec{\eta})\{\vec{c}\} \quad \text{(by definition of } f).$$

If v is a variable y_i, say v is y_1, then $m=k$ (the number of constructors of $\mathbb{A}$); also, H_i is of type τ_i and is without free occurrences of y_1, by the lemma's condition ($i=1\dots k$). Let $G_i[u_1^o \dots u_{r_i}^o, \vec{x}, y_2 \dots y_q]$ be the normal form of $H_i[\vec{x},y_2\dots y_q]\vec{u}$. Then each G_i is of type o, and so, by the main-induction's assumption, we have functions $g_1 \dots g_k \in MR_2(\mathbb{A}_*)$ such that, for all $\zeta_1\dots\zeta_{r_i},\xi_1\dots\xi_p,\eta_2\dots\eta_q \in \mathbb{A}$,

$$g_i(\zeta_1\dots\zeta_{r_i},\xi_1\dots\xi_p,\eta_2\dots\eta_q)\{\vec{c}\}$$
$$=_\beta G_i[\zeta_1\{\vec{c}\},\dots,\zeta_{r_i}\{\vec{c}\},\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\vec{\eta}_2\cdots\vec{\eta}_q]$$
$$=_\beta H_i[\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\vec{\eta}_2\cdots\vec{\eta}_q]\zeta_1\{\vec{c}\}\cdots\zeta_{r_i}\{\vec{c}\}.$$

Now let $f \in MR_2(\mathbb{A}_*)$ be defined by tiered monotonic recurrence:

$$f(c_i^1(z_1\dots z_{r_i}),\vec{x},y_1\dots y_q)=g_i(f_1\dots f_{r_i},\vec{x},\vec{y}), \quad \text{where } f_j =_{\text{df}} f(z_j,\vec{x},\vec{y}),\ i=1\dots k.$$

To conclude the proof we show, by induction on $\zeta \in \mathbb{A}$, that for all $\xi_1\dots\xi_p,\eta_2\dots\eta_q \in \mathbb{A}$, we have

$$F[\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\bar{\zeta},\vec{\eta}_2\cdots\vec{\eta}_q] \equiv \bar{\zeta}\vec{H}^*$$
$$=_\beta \zeta\{\vec{H}^*\}$$
$$=_\beta f(\zeta,\vec{\xi},\vec{\eta})\{\vec{c}\},$$

where H_i^* abbreviates $H_i[\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\vec{\eta}_2\dots\vec{\eta}_q]$ and where we write $\vec{H}^*$ for the sequence $H_1^*\dots H_k^*$. If $\zeta = c_i(\zeta_1\dots\zeta_{r_i})$ then

$$
\begin{aligned}
&F[\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\zeta,\vec{\eta}_2\dots\vec{\eta}_q]\\
&\quad\equiv \zeta H_1^*\cdots H_k^*\\
&\quad\equiv (\lambda\vec{c}.c_i(\zeta_1\{\vec{c}\})\cdots(\zeta_{r_i}\{\vec{c}\}))H_1^*\cdots H_k^*\\
&\quad=_\beta H_i^*(\zeta_1\{\vec{H}^*\})\cdots(\zeta_{r_i}\{\vec{H}^*\})\\
&\qquad\qquad\qquad\qquad\text{(by β-reductions)}\\
&\quad=_\beta H_i^*(f(\zeta_1,\vec{\xi},\vec{\eta})\{\vec{c}\})\cdots(f(\zeta_{r_i},\vec{\xi},\vec{\eta})\{\vec{c}\})\\
&\qquad\qquad\qquad\qquad\text{(by induction assumption for the ζ_j's)}\\
&\quad=_\beta G_i[f(\zeta_1,\vec{\xi},\vec{\eta})\{\vec{c}\},\dots,f(\zeta_{r_i},\vec{\xi},\vec{\eta})\{\vec{c}\},\xi_1\{\vec{c}\},\dots,\xi_p\{\vec{c}\},\vec{\eta}_2\cdots\vec{\eta}_q]\\
&\qquad\qquad\qquad\qquad\text{(by definition of G_i)}\\
&\quad=_\beta g_i(f(\zeta_1,\vec{\xi},\vec{\eta}),\dots,f(\zeta_{r_i},\vec{\xi},\vec{\eta}))\{\vec{c}\},\vec{\xi},\vec{\eta})\{\vec{c}\}\\
&\qquad\qquad\qquad\qquad\text{(by the main-induction's assumption)}\\
&\quad=_\beta f(\zeta,\vec{\xi},\vec{\eta})\{\vec{c}\}\quad\text{(by definition of f).}\quad\square
\end{aligned}
$$

From Corollaries 3.2 and 3.5 we obtain our main result, namely a subrecursive characterization of the functions definable in **1**λ:

Theorem 3.6. *For every free algebra* $\mathbb{A}$,

$$
\begin{aligned}
MR_2(\mathbb{A}) &= MR_\omega(\mathbb{A})\\
&= \textit{the class of functions over } \mathbb{A} \textit{ definable in } \mathbf{1}\lambda.
\end{aligned}
$$

References

[1] S. Abiteboul and V. Vianu, Fixpoint extensions of first-order logic and datalog-like languages, in: *Proc. 4th IEEE Symp. on Logic in Computer Science* (1989) 71–79.

[2] S. Abiteboul and V. Vianu, Generic computation and its complexity, in: *Proc. 23rd ACM Symp. on Theory of Computing* (1991) 209–219.

[3] S. Bellantoni and S. Cook, A new recursion-theoretic characterization of the poly-time functions, *Computational Complexity* **2** (1992) 97–110.

[4] C.H. Bennett, Demons, engines and the Second Law, *Sci. Am.* (November 1986) 108–116.

[5] S. Bloch, Functional characterizations of uniform log-depth and polylog-depth circuit families, in: *Proc. 7th IEEE Conf. on Structure in Complexity* (1992) 193–206.

[6] C. Böhm and A. Berarducci, Automatic synthesis of typed λ-programs on term algebras, *Theoret. Comput. Sci.* **39** (1985) 135–154.

[7] G. Kreisel, La Prédicativité, *Bull. Soc. Math. France* **88** (1960) 371–391.

[8] D. Leivant, Monotonic use of space and computational complexity over abstract structures, Technical Report CMU-CS-89-212, School of Computer Science, Carnegie Mellon Univ., 1989.

[9] D. Leivant, Subrecursion and lambda representation over free algebras, in: S. Buss and P. Scott, eds., *Feasible Mathematics*, Perspective in Computer Science (Birkhäuser-Boston, New York, 1990) 281–292.

[10] D. Leivant, Stratified functional programs and computational complexity, in: *Proc. Conf. Record of the 20th Ann. Symp. on Principles of Programming Languages* (ACM, New York, 1993) 325–333.

[11] D. Leivant, Predicative Peano theories and the axiomatization of feasible mathematics, to appear.

[12] D. Leivant, Abstraction and computational complexity, to appear.

[13] D. Leivant and J.-Y. Marion, Lambda characterizations of poly-time, in: J. Tiuryn, ed., *Fundamenta Informaticae*, Special issue on Typed Lambda Calculi, 1993.

[14] Bertrand Russell, Mathematical logic as based on the theory of types, *Am. J. Math.* **30** (1908) 222–262; reprinted in H. van Heijenoort, ed., *From Frege to Gödel* (Harvard Univ. Press, Cambridge, MA, 1967) 150–182.

[15] H. Schwichtenberg, Definierbare Funktionen im Lambda-Kalkul mit Typen, *Archiv Logik Grundlagenforsch.* **17** (1976) 113–114.

[16] H. Simmons, The realm of primitive recursion, *Arch. Math. Logic* **27** (1988) 177–188.

[17] R. Statman, The typed λ-calculus is not elementary recursive, *Theoret. Comput. Sci.* **9** (1979) 73–81.

[18] Hao Wang, A variant to Turing's theory of computing machines, *J. ACM* **4** (1957) 63–92.

[19] M. Zaionc, λ-definability on free algebras, *Ann. Pure Appl. Logic* **51** (1991) 279–300.