

# An applied $\lambda$ -calculus for iteration templates

Harold Simmons

School of Mathematics, The University, Manchester, England

hsimmons@manchester.ac.uk

## Abstract

Let  $\lambda\mathbf{H}$  be the term calculus of Howard's system of constructive ordinals. I use this to name iteration gadgets (generalized ordinal notations). I isolate a family of higher order ordinal functions which give a partial semantics of  $\lambda\mathbf{H}$ . I indicate how  $\lambda\mathbf{H}$  provides an intrinsic measure of each ordinal below the Howard ordinal.<sup>1</sup>

## Contents

1	Introduction . . . . .	1
2	Examples of long iteration . . . . .	3
3	Categorical matters . . . . .	6
4	The mechanics of $\lambda\mathbf{H}$ . . . . .	10
5	Arithmetic in $\lambda\mathbf{H}$ . . . . .	15
6	Canonical notations . . . . .	19
7	Helpful ordinal functions . . . . .	21
8	The Eastwood hierarchy . . . . .	28
	References . . . . .	31

## 1 Introduction

We have been aware of the ordinals for something over 100 years. Each is the order type of a well ordered set and, in most parts of Mathematics, that is exactly how they are used; as measures of length. Occasionally (as in parts of Proof Theory) an ordinal is endowed with more structure. When used in this way we are concerned with its internal behaviour and its complexity. This requires a more general notion of 'ordinal'. Kleene's  $\mathcal{O}$  was perhaps the first attempt to get at this more delicate idea although, with hindsight, we can see that earlier work was concerned with the same problem. In this context phrases such as 'ordinal notation' or 'ordinal presentation' are often used.

There are several applications of ordinals where the results depend crucially on the notation used. The study of hierarchies of functions is a case in point. The root of the problem lies with the limit ordinals. Each (small) limit ordinal  $\lambda$  has a fundamental sequence  $\lambda[\cdot]$  which is used to access  $\lambda$  from below. Often there is an 'obvious' choice for  $\lambda[\cdot]$ , but even for small ordinals there are some alternatives. Thus

$$\omega[r] = r \quad \text{or} \quad \omega[r] = 1 + r$$

(for  $r \in \mathbb{N}$ ) are two fundamental sequence for  $\omega$ . (It turns out that the second one is better than the first.) Some ordinal constructions depends on the use of a particular fundamental sequence to pass across a limit. In these circumstances if we change the

---

<sup>1</sup>This is a slightly revised version of a paper with the same title written about 1998. Since then [18, 19] have been published.

fundamental sequence then we change the result of the construction. In such situations we need a more liberal notion of ‘ordinal’.

For these uses it is not the length, or even the well foundedness of an ordinal that matters. The description of the ordinal is used to track through a nested family of iterations over  $\mathbb{N}$  where every now and then a limit leap is made. In these circumstances we should think of an ordinal as an **iteration template** not as a description of some well ordered set. Even the standard arithmetic can go wrong.

Let  $F$  be a higher order function, say

$$Ffx = f(x + 1)$$

for each  $f : \mathbb{N} \rightarrow \mathbb{N}$  and  $x \in \mathbb{N}$ . The finite iterates  $F^r$  of  $F$  are just composites of  $F$ , and satisfy  $F^s \circ F^r = F^{r+s}$ . The  $\omega^{\text{th}}$  iterate  $F^\omega$  is defined by  $F^\omega fx = F^{\omega[x]}fx$  making use of the selected fundamental sequence. For the particular example we have

$$F^\omega fx = f(x + \omega[x])$$

to illustrate how  $F^\omega$  depends on more than the ordinal  $\omega$ . It is easy to extend these iterations to larger ordinals (with selected fundamental sequences), but then the arithmetic of these exponents needs some care. For instance, if we want

$$F^\beta \circ F^\alpha = F^{\alpha+\beta}$$

for iteration templates  $\alpha, \beta$ , then we have to drop some of the standard ordinal rules. Assuming this rule we have

$$F^{1+\omega}fx = (F^\omega \circ F)fx = F^\omega(Ff)x = (Ff)(x + \omega[x]) = f(1 + x + \omega[x])$$

so that  $1 + \omega$  is required to be a limit template with  $(1 + \omega)[x] = 1 + \omega[x]$  for each  $x \in \mathbb{N}$ . In particular,  $1 + \omega \neq \omega$ .

In this sense each iteration template is a structured family of functions. As such it has an intrinsic complexity which is *not* the same as the ordinal it may represent. Any inflationary and strictly monotone functions  $\omega[\cdot] : \mathbb{N} \rightarrow \mathbb{N}$  can be used as the fundamental sequence for  $\omega$ , and some of these are quite complicated. (In fact, sometimes more liberal versions of a fundamental sequence for  $\omega$  are needed. An example of such a use is given in [20].)

How can we measure the complexity of an iteration template? Before I suggest an answer to that, think of how we measure the complexity of a numeric function. We use Gödel’s T and its associated gadgetry such as the Grzegorzcyk hierarchy. The term calculus of Gödel’s T is an applied  $\lambda$ -calculus  $\lambda\mathbf{G}$  in which many numeric functions of any finite type can be named and evaluated. The syntax used to name a function provides a rather fine (not coarse) indication of its complexity. In general, the higher the types used in the name, the more complicated the function might be. Less fine measure of complexity can be obtained by collapsing this syntax in certain ways.

Each ordinal below  $\epsilon_0$  can be simulated in a part of  $\lambda\mathbf{G}$ , but larger ordinals need bigger parts. These ordinals give an indication of the overall complexity of the associated parts. In general, an exponentiation to base  $\omega$  corresponds to an increase in type level.

In this paper I describe a slightly enriched applied  $\lambda$ -calculus  $\lambda\mathbf{H}$  which does for iteration templates what  $\lambda\mathbf{G}$  does for numeric functions. The calculus  $\lambda\mathbf{H}$  is formed

by adding to  $\lambda\mathbf{G}$  explicit (rather than simulated) gadgets for generating long and nested iterations. It also has a modicum of equational reasoning. In essence  $\lambda\mathbf{H}$  is the term calculus of Howard's system of abstract constructive ordinals [12], although the syntax is not the same. The main difference is that  $\lambda\mathbf{H}$  has no quantificational machinery and no set theoretic power. The only reasoning aspect of  $\lambda\mathbf{H}$  is the  $\omega$ -rule; two notations for functions  $\phi, \psi : \mathbb{N} \rightarrow \mathbb{A}$  are equivalent if all corresponding values  $\phi m$  and  $\psi m$  (for  $m \in \mathbb{N}$ ) are equivalent. The calculus can name many highly complex iteration templates. By applying each such iteration to the ordinals, we name an ordinal. In this respect,  $\lambda\mathbf{H}$  is an all embracing system of ordinal notations (for ordinals below the Howard ordinal).

The paper is written in several phases.

Section 2 and 3 set the scene with several examples of iterations and some background material. The calculus itself is described in Section 4.

The primary purpose of  $\lambda\mathbf{H}$  is to name iteration templates. A secondary purpose is to name for ordinals. To determine the ordinal named by a term we must massage it into a certain form. We need to carry out some arithmetic in  $\lambda\mathbf{H}$ . This is the topic of sections 5 and 6.

The idea behind these naming terms is the use of higher order fixed point extractors. An understanding of  $\lambda\mathbf{H}$  requires an analysis of such gadgets on the actual ordinals. This is carried out in Section 7. Although there is an occasional reference to earlier sections, this section is essentially independent of the rest of the paper.

Several other systems of ordinal notations have been developed. Section 8 gives a pseudo-historical survey of some of these showing how they can be subsumed into  $\lambda\mathbf{H}$ . I indicate how Veblen's original ideas, the use of Schütte brackets, and the Bachmann technique can all be developed in  $\lambda\mathbf{H}$ .

As well as Howard's work [12], this material is related to the work of Aczel [1] and Feferman [9], and the more recent work of Danner [6], [7].

To conclude this introduction, let me say a few words about the notation of the paper.

The dash  $(\cdot)'$  is used for several different constructs, each of which is a kind of successor operation. For  $r \in \mathbb{N}$  (the natural numbers) or  $\alpha \in \text{Ord}$  (the ordinals) we let

$$r' = 1 + r \quad \alpha' = \alpha + 1$$

the obvious successor. For a set  $\mathbb{S}$  we let

$$\mathbb{S}' = \mathbb{S} \rightarrow \mathbb{S}$$

be the space of functions from  $\mathbb{S}$  to  $\mathbb{S}$ . Each operation  $(\cdot)'$  can be iterated. In particular, each set  $\mathbb{S}$  gives a chain  $\mathbb{S}, \mathbb{S}', \mathbb{S}'', \mathbb{S}''', \dots$  of function spaces. There will be other similar uses of  $(\cdot)'$  introduced as the development proceeds.

## 2 Examples of long iteration

How can we organize and analyse long and nested iterations? Before we consider this question, we need a stock of examples to illustrate the aspects we should be aware of.

We are concerned here with the set  $\text{Ord}$  of countable ordinals, and we refer to these as 'ordinals'. This set  $\text{Ord}$  has a distinguished member 0, carries a successor operation  $S$ , and a supremum operation  $\bigvee$  on (countable) subsets. We write  $\alpha, \beta, \gamma$  for arbitrary ordinals, and  $\lambda, \mu, \nu$  for limit ordinals. Any variation from this convention will be indicated.

2.1 EXAMPLES. (a) Working on  $\mathbb{O}rd$  each of the three basic arithmetical operations can be generated by iterating an earlier operation.

Explicit	Base	Step	Leap
$\beta + \alpha = S^\alpha \beta$	$\beta + 0 = \beta$	$\beta + S\alpha = S(\beta + \alpha)$	$\beta + \lambda = \bigvee \{\beta + \alpha \mid \alpha < \lambda\}$
$\beta \times \alpha = (\cdot + \beta)^\alpha 0$	$\beta \times 0 = 0$	$\beta \times S\alpha = (\beta \times \alpha) + \beta$	$\beta \times \lambda = \bigvee \{\beta \times \alpha \mid \alpha < \lambda\}$
$\beta^\alpha = (\cdot \times \beta)^\alpha 1$	$\beta^0 = 1$	$\beta^{S\alpha} = \beta^\alpha \times \beta$	$\beta^\lambda = \bigvee \{\beta^\alpha \mid \alpha < \lambda\}$

Next we may iterate exponentiation to obtain the stacking function

$$\beth(\gamma, \beta, \alpha) = (\beta^\bullet)^\alpha \gamma$$

where

$$\beth(\gamma, \beta, 0) = \gamma \quad \beth(\gamma, \beta, S\alpha) = \beta^{\beth(\gamma, \beta, \alpha)} \quad \beth(\gamma, \beta, \lambda) = \bigvee \{\beth(\gamma, \beta, \alpha) \mid \alpha < \lambda\}$$

for the usual  $\alpha, \beta, \gamma, \lambda$ . We could continue up an analogue of the Grzegorzcyk hierarchy.

(b) The (countable levels of the) Zermelo hierarchy of sets are generated

$$V_\alpha = \mathcal{P}^\alpha \emptyset \quad V_0 = \emptyset \quad V_{\alpha+1} = \mathcal{P}V_\alpha \quad V_\lambda = \bigcup \{V_\alpha \mid \alpha < \lambda\}$$

by iterating the power set operation.

(c) Let  $D$  be the operation which converts a closed subset of a topological space into its subset of limit points. The Cantor-Bendixson rank is obtained by iterating  $D$ .

$$D^\alpha X \quad D^0 = X \quad D^{\alpha+1} = D(D^\alpha X) \quad D^\lambda X = \bigcap \{D^\alpha X \mid \alpha < \lambda\}$$

This, of course, is the example which inspired Cantor to invent the ordinals.

(d) Let  $B$  be the operation which, when supplied with a family  $\mathcal{F}$  of subsets of a topological space will first take  $\mathcal{F}$  and all the complements of members of  $\mathcal{F}$ , and then return all countable unions of this extended family of sets. The borel sets of a (suitable) topological space  $S$  are generated by iterating  $B$  on the topology  $OS$  of  $S$ .

$$\Sigma_1 S = OS \quad \Sigma_{\alpha+1} S = B(\Sigma_\alpha S) \quad \Sigma_\lambda S = B(\bigcup \{\Sigma_\alpha S \mid \alpha < \lambda\})$$

The leap to a limit ordinal is not quite as expected. If we take the ‘obvious’ iteration then  $\Sigma_{1+\alpha} S = B^{\alpha+1}(OS)$  holds for all  $\alpha$ .

(e) Let  $\mathbf{Veb}$  be the operation which converts a normal ordinal function  $f$  into the enumeration of its fixed points. Iterating  $\mathbf{Veb}$  produces Veblen hierarchy  $\phi_f$  on  $f$ .

$$\phi_f 0 = f \quad \phi_f(\alpha + 1) = \mathbf{Veb}(\phi_f \alpha) \quad \phi_f \lambda = \mathbf{Veb}(\bigvee \{\phi_f \alpha \mid \alpha < \lambda\})$$

Again the leap to a limit ordinal is not quite what is expected. We may check that  $\phi_f(1 + \alpha) = \mathbf{Veb}^{\alpha+1} f$  holds, where here we use the ‘obvious’ iteration of  $\mathbf{Veb}$ .

The Veblen hierarchy on  $\omega^\bullet$  provides a notation for each ordinal below the least strongly critical ordinal  $\Gamma_0$ . These days we can notate much larger ordinals. In [16] Schütte generalized Veblen’s method to go quite a bit further. A different method was devised by Bachmann [3] and this is capable of notation considerably larger ordinals. In Section 8 we will see how both these techniques fit into the methods of this paper.<sup>2</sup>

(f) Several ordinal ranking techniques are used in ring theory. For a large class of rings we can define both a Krull dimension and a Gabriel dimension. These are essentially the same thing (when both exist) but the values are out of step by 1. ■

<sup>2</sup>These sections are now out of date.

What is the abstract set up of these examples? We certainly need a set  $\mathbb{A}$ , an element  $a \in \mathbb{A}$ , and an operation  $A : \mathbb{A}'$  which we iterate. We also need some kind of ‘collection’ operation  $\mathcal{A}$  on appropriate subsets of  $\mathbb{A}$ . We then take either  $a(\cdot)$  or  $a[\cdot]$  where

$$\begin{array}{ll} a(0) & = a \\ a(\alpha + 1) & = Aa(\alpha) \\ a(\lambda) & = \mathcal{A}\{a(\alpha) \mid \alpha < \lambda\} \end{array} \quad \begin{array}{ll} a[0] & = a \\ a[\alpha + 1] & = Aa[\alpha] \\ a[\lambda] & = A(\mathcal{A}\{a[\alpha] \mid \alpha < \lambda\}) \end{array}$$

for the usual  $\alpha, \lambda$ . It can be checked that

$$a[1 + \alpha] = a(\alpha + 1)$$

for all  $\alpha$ . There can be different kinds of limit leaps within the same general set up.

These iterations are ‘history insensitive’ since each leap to a limit ordinal  $\lambda$  does not need all ordinals  $\alpha < \lambda$ , any cofinal selection will do. There are some ‘history sensitive’ iterations. Let’s look at various hierarchies of functions on  $\mathbb{N}$  as described in [8].

2.2 EXAMPLES. (a) The Slow, Hardy, and Fast hierarchies are generated by

$$\begin{array}{lll} S_0 & = \mathbf{zero} & H_0 & = id & F_0 & = S \\ S_{\alpha+1} & = S \circ S_\alpha & H_{\alpha+1} & = H_\alpha \circ S & F_{\alpha+1}x & = F_\alpha^{x+1}x \\ S_\lambda x & = S_{\lambda[x]}x & H_\lambda x & = H_{\lambda[x]}x & F_\lambda x & = F_{\lambda[x]}x \end{array}$$

for the usual  $\alpha, \lambda$ . Here  $\mathbf{zero}$  is the constant function with value 0,  $id$  is the identity function, and  $S$  is the successor function (on  $\mathbb{N}$ ). The crucial new component is the use of a selected fundamental sequence  $\lambda[\cdot]$  for each limit ordinal  $\lambda$ . All have the form

$$G_\alpha = J^\alpha g \quad J^0 g = g \quad J^{\alpha+1} g = J(J^\alpha g) \quad J^\lambda g x = J^{\lambda[x]} g x$$

for a certain jump operator  $J : \mathbb{N}''$  and base function  $g : \mathbb{N}'$ .

(b) The descent functionals  $D_\alpha : \mathbb{N}''$  are generated by

$$D_0 \mathbf{Zero} \quad D_{\alpha+1} f = S \circ D_\alpha f \circ f \quad D_\lambda f x = D_{\lambda[x]} f x$$

where  $\mathbf{Zero} : \mathbb{N}''$  with  $\mathbf{Zero} f x = 0$  for each function  $f$  and  $x \in \mathbb{N}$ . Notice that  $D_\alpha = T^\alpha \mathbf{Zero}$  for some appropriate  $T : \mathbb{N}''$ . ■

The new component here is that the iterations do not take place over ordinals but over ordinals with a selected fundamental sequence for each limit ordinal (or a family of selected fundamental sequences). This has an impact on the output of the iteration.

Consider the Slow hierarchy  $S_\bullet$ . Since  $S_r x = x + r$  for each  $r, x \in \mathbb{N}$ , we have

$$S_\omega x = S_{\omega[x]} x = \omega[x] + x$$

to show that this function is sensitive to the choice of  $\omega[\cdot]$ . As we progress up the ordinals these fundamental sequences will become highly nested, and a small change in the choice of  $\omega[\cdot]$  can have a dramatic effect on the iteration. For this reason we need to study something more general than ordinal iteration.

Roughly speaking an **iteration template** is a piece of syntax which, when supplied with the appropriate data, will guide a path through that data to output the result of the notated iteration on the data. We wish to make this intuitive idea can be made precise.

### 3 Categorical matters

The term calculus  $\lambda\mathbf{G}$  of Gödel's T is designed to name various numeric gadgets, most of which are higher order. In particular, it encapsulates the idea of a 'short' iteration, one along the natural numbers. The calculus  $\lambda\mathbf{H}$  is designed to do a similar job for iteration templates of which ordinal iterations form a major part. It encapsulates the idea of a 'long' and nested iteration and, in particular, one along an ordinal. Before we look at the motivating ideas behind  $\lambda\mathbf{H}$  let's take another look at the corresponding motivating ideas for  $\lambda\mathbf{G}$ . These, of course, are well known, but perhaps not in the form we need.

**3.1 DEFINITION.** A **peano structure**  $\mathfrak{A} = (\mathbb{A}, a, A)$  is a set  $\mathbb{A}$  furnished with a distinguished element  $a \in \mathbb{A}$  and a 1-placed operation  $A : \mathbb{A}'$ .

A peano arrow

$$\mathfrak{B} \xrightarrow{\phi} \mathfrak{A}$$

from a peano structure  $\mathfrak{B} = (\mathbb{B}, b, B)$  to a peano structure  $\mathfrak{A}$  is a function  $\phi : \mathbb{B} \rightarrow \mathbb{A}$  where  $\phi b = a$  and  $\phi \circ B = A \circ \phi$ .

These objects and arrows form the category **Peano**. ■

**Peano** is a rather simple category. The forgetful functor  $\mathbf{Peano} \rightarrow \mathbf{Set}$  (the category of sets and functions) has a left adjoint. In particular, **Peano** has an initial object. This is the obvious peano structure  $\mathfrak{N} = (\mathbb{N}, 0, S)$  carried by the natural numbers. (Initiality is just a way of stating the Dedekind characterization of  $\mathbb{N}$ .) Because  $\mathfrak{N}$  is initial there is a unique mediating arrow

$$\begin{array}{ccc} \mathbb{N} & \longrightarrow & \mathbb{A} \\ r & \longmapsto & A^r a \end{array}$$

attached to each peano  $\mathfrak{A}$ . Thus  $r \in \mathbb{N}$  is sent to the  $r$ -fold iterate of  $A$  on  $a$ .

Suppose we fix the set  $\mathbb{A}$ . Unless it is very small  $\mathbb{A}$  can carry many different peano structures. Each such  $\mathfrak{A}$  can be thought of as a supplier of its two attributes. We can gather together into one gadget all the associated mediating arrows.

**3.2 DEFINITION.** For a set  $\mathbb{A}$ , the  $\mathbb{N}$ -iterator on  $\mathbb{A}$  is the function  $I_{\mathbb{A}} : \mathbb{N} \rightarrow \mathbb{A}''$  where

$$I_{\mathbb{A}} r y x = y^r x$$

for all  $r \in \mathbb{N}, y : \mathbb{A}, x \in \mathbb{A}$ . ■

The mediating arrow for  $\mathfrak{A} = (\mathbb{A}, a, A)$  is  $r \mapsto I_{\mathbb{A}} r A a$ . Notice how  $I_{\mathbb{A}}$  consumes the attributes of  $\mathfrak{A}$  in reverse order. There is nothing mysterious about this, it's just to make some of the later descriptions a bit cleaner. There is a recursive description of  $I_{\mathbb{A}}$ . Thus

$$I_{\mathbb{A}} 0 y x = x \quad I_{\mathbb{A}} (S r) y x = y(I_{\mathbb{A}} r y x)$$

for  $r \in \mathbb{N}, y : \mathbb{A}', x \in \mathbb{A}$ .

We want to extend these ideas to longer and nested iterations. We have seen in Examples 2.1 and 2.2 there are two aspects we need to address. Firstly, in many situations the leap to a limit is not always done in the 'obvious' way. We need to treat this aspect as one of the parameters of the situation. Secondly, often a limit ordinal  $\lambda$  is viewed along a selected fundamental sequence  $\lambda[\cdot]$ . A change in  $\lambda[\cdot]$  will change the outcome. We need a set up which can handle a more general notion of an iteration template.

3.3 DEFINITION. A limit creator over a set  $\mathbb{A}$  is a function  $\mathcal{A} : \mathcal{L}(\mathbb{A})$  where

$$\mathcal{L}(\mathbb{A}) = (\mathbb{N} \longrightarrow \mathbb{A}) \longrightarrow \mathbb{A}$$

is a higher order function space. ■

Thus  $\mathcal{A} : \mathcal{L}(\mathbb{A})$  converts each sequence

$$p : \mathbb{N} \longrightarrow \mathbb{A}$$

through  $\mathbb{A}$  into an element  $\mathcal{A}p$  of  $\mathbb{A}$ . Some examples of this notion will help.

3.4 EXAMPLES. (a) Let  $\mathcal{A} : \mathcal{P}\mathbb{A} \longrightarrow \mathbb{A}$  be a ‘collection’ operation on  $\mathbb{A}$ . Then the assignment

$$p \mapsto \mathcal{A}\{pr \mid r \in \mathbb{N}\}$$

(for  $p : \mathbb{N} \longrightarrow \mathbb{A}$ ) is a limit creator on  $\mathbb{A}$ . This is used Examples 2.1(a,b,c). It is ‘history insensitive’. Given an operation  $A : \mathbb{A}'$ , the assignment

$$p \mapsto A(\mathcal{A}\{pr \mid r \in \mathbb{N}\})$$

is a limit creator on  $\mathbb{A}$ . This is used in Examples 2.1(d,e).

(b) Let  $\mathbb{A} = \mathbb{N}'$ . Then

$$Lpx = pxx \quad Lpx = px2 \quad Lpx = px1 \quad Lpx = px0$$

(for  $p : \mathbb{N} \longrightarrow \mathbb{N}'$ ,  $x : \mathbb{N}$ ) defines four different limit creators  $L : \mathcal{L}(\mathbb{N}')$ . All four occur quite naturally in various places. The left hand one is called the **diagonal** limit creator.

(c) For  $\mathbb{A} = \mathbb{N}''$  we have  $\nabla : \mathcal{L}(\mathbb{N}'')$  where

$$\nabla pfx = pxfx$$

for  $p : \mathbb{N} \longrightarrow \mathbb{N}''$  and  $f : \mathbb{N}'', x : \mathbb{N}$ . ■

With this notion we can define the extension of Peano we are interested in.

3.5 DEFINITION. A limit structure  $\mathfrak{A} = (\mathbb{A}, a, A, \mathcal{A})$  is a set  $\mathbb{A}$  furnished with a distinguished element  $a \in \mathbb{A}$ , a 1-placed operation  $A : \mathbb{A}'$ , and a limit creator  $\mathcal{A} : \mathcal{L}(\mathbb{A})$ .

A limit arrow

$$\mathfrak{B} \xrightarrow{\phi} \mathfrak{A}$$

from a limit structure  $\mathfrak{B} = (\mathbb{B}, b, B, \mathcal{B})$  to a limit structure  $\mathfrak{A}$  is a function  $\phi : \mathbb{B} \longrightarrow \mathbb{A}$  where  $\phi b = a$ ,  $\phi \circ B = A \circ \phi$ , and

$$\phi \circ (\mathcal{B}q) = \mathcal{A}(\phi \circ q)$$

for each  $q : \mathbb{N} \longrightarrow \mathbb{B}$ .

These objects and arrows form the category **Limit**. ■

The new clause simply says ‘ $\phi$  commutes with limits’. The composite of two limit arrows is a limit arrow, and so we do have a category.

By Example 3.4(a), a ‘collection’ operation  $\mathcal{A}$  on a peano structure  $(\mathbb{A}, a, A)$ , gives a ‘history insensitive’ limit structure  $(\mathbb{A}, a, A, \mathcal{A})$ . There are ‘history sensitive’ examples.

3.6 EXAMPLE. Let  $F : \mathbb{N}''$ , and let  $\Delta$  be the diagonal limit creator. Then  $(\mathbb{N}', id, F, \Delta)$  is a limit structure. This kind of structure underlies Examples 2.2(a). ■

The forgetful functor  $\mathbf{Limit} \rightarrow \mathbf{Set}$  has a left adjoint (as does the forgetful functor  $\mathbf{Limit} \rightarrow \mathbf{Peano}$ ). In particular,  $\mathbf{Limit}$  has an initial object. This has the form

$$\mathfrak{D} = (\mathbb{O}, 0, S, L)$$

for some set  $\mathbb{O}$  and attributes  $0, S, L$ . The carrier  $\mathbb{O}$  is broader than  $\mathbb{Ord}$ . Its members are sometimes called the (unstructured) tree ordinals. An account of these is given in [8]. One of the aims of this paper is to throw some light on the nature of  $\mathfrak{D}$ .

Since  $\mathfrak{D}$  is initial there is a unique mediating arrow  $\mathfrak{D} \rightarrow \mathfrak{A}$  attached to each limit structure. Let us write  $\alpha, \beta, \gamma, \dots$  for elements of  $\mathbb{O}$  and  $\mathfrak{A}(\cdot)$  for the mediating arrow. Then  $\mathfrak{A}(\cdot) : \mathbb{O} \rightarrow \mathbb{A}$  is generated by

$$\mathfrak{A}(0) = a \quad \mathfrak{A}(S\alpha) = A(\mathfrak{A}(\alpha)) \quad \mathfrak{A}(Lp) = \mathcal{A}(u \mapsto \mathfrak{A}(pu))$$

for each  $\alpha \in \mathbb{O}$  and  $p : \mathbb{N} \rightarrow \mathbb{A}$ . This is the spine of  $\mathfrak{A}$ .

More often than not only a part of the spine of  $\mathfrak{A}$  is considered; that part generated using one specific notation for each ordinal. Let's call this the specific spine.

3.7 EXAMPLES. (a) (0) Since  $\bigvee$  on  $\mathbb{Ord}$  is a 'collection' operation,

$$\mathfrak{Ord} = (\mathbb{Ord}, 0, S, \bigvee)$$

is a limit structure. The specific spine of  $\mathfrak{Ord}$  is just the identity function on  $\mathbb{Ord}$ .

(+) For each  $\beta$  we see that  $\mathfrak{A}_\beta = (\mathbb{Ord}, \beta, S, \bigvee)$  is a limit structure, and the specific spine gives ordinal addition,  $\mathfrak{A}_\beta(\alpha) = \beta + \alpha$ .

( $\times$ ) For each  $\beta \in \mathbb{Ord}$  we see that  $\mathfrak{M}_\beta = (\mathbb{Ord}, 0, \cdot + \beta, \bigvee)$  is a limit structure using  $\cdot + \beta$  as the successor operation. The specific spine gives multiplication,  $\mathfrak{M}_\beta(\alpha) = \beta \times \alpha$ .

( $\cdot$ ) For each  $\beta \in \mathbb{Ord}$  we see that  $\mathfrak{E}_\beta = (\mathbb{Ord}, 1, \cdot \times \beta, \bigvee)$  is a limit structure using  $\cdot \times \beta$  as the successor operation. The specific spine gives exponentiation,  $\mathfrak{E}_\beta(\alpha) = \beta^\alpha$

( $\sqsupset$ ) For each pair of ordinals  $\gamma, \beta$  we see that  $\mathfrak{S}_{\gamma, \beta} = (\mathbb{Ord}, \gamma, \beta^\bullet, \bigvee)$  is a limit structure where the specific spine gives the stacking function.

This trick can be repeated indefinitely.

(b) Let  $\mathbb{V}$  be the Zermelo universe. Then, ignoring the size problem,  $(\mathbb{V}, \emptyset, \mathcal{P}, \bigcup)$  is a limit structure, and the specific spine is the initial stretch of the Zermelo hierarchy.

(c) Suppose  $\mathbb{A} = \mathcal{PS}$  for some set  $\mathbb{S}$ . An inflator or a deflator on  $\mathbb{S}$  is a function  $i : \mathbb{A}'$  or  $d : \mathbb{A}'$  which is monotone and inflationary or deflationary, respectively. Given such functions  $(\mathcal{PS}, \emptyset, i, \bigcup)$  or  $(\mathcal{PS}, \mathbb{S}, d, \bigcap)$  is a limit structure.

(d) Many function hierarchies are obtained as the specific spine of  $\mathfrak{F} = (\mathbb{N}', f, F, L)$  for appropriate  $f : \mathbb{N}', F : \mathbb{N}'', L : \mathcal{L}(\mathbb{N}')$ . For each  $r < \omega$  we have  $\mathfrak{F}(r) = F^r f$  and then  $\mathfrak{F}(\omega) = L(r \mapsto F^{\omega[r]} f)$  where  $\omega[\cdot]$  is the selected fundamental sequence for  $\omega$ .

For instance, consider  $\mathfrak{A} = (\mathbb{N}, id, f \circ -, L)$  where  $L$  is one of the limit creators of Example 3.4(b). The assignments  $\omega[x] = x$  and  $\omega[x] = x + 1$  are two commonly used fundamental sequences. with these  $\mathfrak{A}(\omega)x$  is  $f^z y$  for certain  $y, z$  depending on  $L$  and  $\omega[\cdot]$ .

(e) Using the diagonal limit creator  $\Delta$ , the specific spines of

$$\mathfrak{S} = (\mathbb{N}', \mathbf{zero}, S \circ -, \Delta) \quad \mathfrak{H} = (\mathbb{N}', id, - \circ S, \Delta) \quad \mathfrak{F} = (\mathbb{N}', S, \mathbf{ack}, \Delta)$$

are, respectively, the Slow, the Hardy, and the Fast hierarchies.

(f) Let  $f : \mathbb{N}'$  be any function and define  $F : \mathbb{N}''$  by  $Fg = S \circ g \circ f$  (for  $g : \mathbb{N}$ ). With  $\mathfrak{A} = (\mathbb{N}', \mathbf{zero}, F, \Delta)$  we find that  $\mathfrak{A}(\alpha) = D_\alpha f$  for each  $\alpha \in \text{Ord}$ .

The parameter  $f$  can be uncoupled. Let  $F^* : \mathbb{N}'''$  be given by  $F^*Gg = S \circ Gg \circ g$  for  $G : \mathbb{N}''$ ,  $g : \mathbb{N}'$ . With  $\mathfrak{D} = (\mathbb{N}'', \mathbf{Zero}, F^*, \nabla)$  we have  $\mathfrak{D}(\alpha) = D_\alpha$  for each  $\alpha \in \text{Ord}$ . ■

Each set  $\mathbb{A}$  carries many limit structures. Each such  $\mathfrak{A}$  can be viewed as supplying its three attributes. We can gather into one gadget all the associated mediating arrows.

**3.8 DEFINITION.** For a set  $\mathbb{A}$ , the  $\mathbb{O}$ -iterator on  $\mathbb{A}$  is the function

$$J_{\mathbb{A}} : \mathbb{O} \longrightarrow \mathcal{L}(\mathbb{A}) \longrightarrow \mathbb{A}''$$

generated by

$$J_{\mathbb{A}}0lyx = x \quad J_{\mathbb{A}}(S\alpha)lyx = y(J_{\mathbb{A}}lyx) \quad J_{\mathbb{A}}(Lp)lyx = l(u \mapsto J_{\mathbb{A}}(pu)lyx)$$

for all  $\alpha \in \mathbb{O}$ ,  $p : \mathbb{N} \longrightarrow \mathbb{O}$  and each  $l : \mathcal{L}(\mathbb{A})$ ,  $y : \mathbb{A}'$ ,  $x \in \mathbb{A}$  (and  $u \in \mathbb{N}$ ). ■

To conclude this section we look at a trick we use several times.

**3.9 DEFINITION.** Each limit creator  $\mathcal{A} : \mathcal{L}(\mathbb{A})$  has a lift  $\mathcal{A}' : \mathcal{L}(\mathbb{A}')$  given by  $\mathcal{A}'px = \mathcal{A}q$  where  $qu = pux$  for all  $p : \mathbb{N} \longrightarrow \mathbb{A}'$ ,  $x \in \mathbb{A}$ , and  $u \in \mathbb{N}$ . ■

We can convert each limit structure  $\mathfrak{A} = (\mathbb{A}, a, A, \mathcal{A})$  into a lifted limit structure  $\mathfrak{A}' = (\mathbb{A}', id_{\mathbb{A}}, A \circ -, \mathcal{A}')$  on  $\mathbb{A}'$  where the new successor operation is just composition with  $A$ . We find that  $\mathfrak{A}(\alpha) = \mathfrak{A}'(\alpha)a$  for all  $\alpha \in \mathbb{O}$ . The base value can be uncoupled from the iteration mechanism.

For each set  $\mathbb{A}$  lifting gives us an operation  $\uparrow_{\mathbb{A}} : \mathcal{L}(\mathbb{A}) \longrightarrow \mathcal{L}(\mathbb{A}')$ , so that  $\mathcal{A}'$  is a shorthand for  $\uparrow_{\mathbb{A}} \mathcal{A}$ . Lifting can be iterated. From  $\mathcal{A} : \mathcal{L}(\mathbb{A})$  we generate

$$\mathcal{A}' : \mathcal{L}(\mathbb{A}'), \mathcal{A}'' : \mathcal{L}(\mathbb{A}''), \mathcal{A}''' : \mathcal{L}(\mathbb{A}'''), \dots$$

where

$$\begin{aligned} \mathcal{A}'px &= \mathcal{A}(u \mapsto pux) \\ \mathcal{A}''qyx &= \mathcal{A}'(u \mapsto quyx) = \mathcal{A}(u \mapsto quyx) \\ \mathcal{A}'''rzyx &= \mathcal{A}''(u \mapsto ruz)yx = \mathcal{A}'(u \mapsto ruz)yx = \mathcal{A}(u \mapsto ruzyx) \\ &\vdots \end{aligned}$$

for all  $x : \mathbb{A}$ ,  $y : \mathbb{A}'$ ,  $z : \mathbb{A}$ , ... and sequence

$$p : \mathbb{N} \longrightarrow \mathbb{A}', q : \mathbb{N} \longrightarrow \mathbb{A}'', r : \mathbb{N} \longrightarrow \mathbb{A}''', \dots$$

of functions. Note, however, that this notation omits a lot of information. Thus

$$\mathcal{A}' = \uparrow_{\mathbb{A}} \mathcal{A} \quad \mathcal{A}'' = \uparrow_{\mathbb{A}'}(\uparrow_{\mathbb{A}} \mathcal{A}) \quad \mathcal{A}''' = \uparrow_{\mathbb{A}''}(\uparrow_{\mathbb{A}'}(\uparrow_{\mathbb{A}} \mathcal{A})) \quad \dots$$

and the uniformity of this construction is at the level of *types* not inhabitants.

## 4 The mechanics of $\lambda\mathbf{H}$

The system  $\lambda\mathbf{G}$  is an applied  $\lambda$ -calculus forming the term calculus of Gödel's T. It names many functions in the function space hierarchy on  $\mathbb{N}$ , but has no reasoning facilities. The properties and the complexities of the functions so named, especially the low level ones, have been studied for many years. The long Grzegorzcyk hierarchy (up to  $\epsilon_0$ ) stratifies the complexities that occur, and the ordinals involved are intimately connected with the type levels used in the naming terms. Descriptions of  $\lambda\mathbf{G}$  can be found in [17] and [10], with some useful material in [11]. However, these are not explicitly in terms of a  $\lambda$ -calculus.

The system  $\lambda\mathbf{H}$  is an applied  $\lambda$ -calculus enriched with a modicum of equational reasoning. It is formed by adding to  $\lambda\mathbf{G}$  explicit iteration templates. In a sense  $\lambda\mathbf{H}$  is quite weak, for it has no set theoretical facilities and little reasoning power. However, it has the ability to name some quite large ordinals and some extremely fast functions, way beyond those obtainable in  $\lambda\mathbf{G}$ . In essence  $\lambda\mathbf{H}$  is the term calculus of Howard's system of constructive ordinals, [12]. The eventual aim is to use  $\lambda\mathbf{H}$  to stratify the complexities of iteration templates in the same way that  $\lambda\mathbf{G}$  is used to stratify numeric functions.

The calculus  $\lambda\mathbf{H}$  is related to Feferman's  $OR_1^\omega$  described in section 9.1 of [2]. However,  $OR_1^\omega$  has quite a lot of reasoning facilities. The paper [2] is a useful survey of Gödel's T, Howard's system, and associated material.

The calculus  $\lambda\mathbf{H}$  has syntactic categories Type, Term, Typing-judgement, Typing-derivation, Computation, Equational-judgement, and Equational-derivation.

**4.1 DEFINITION.** The calculus  $\lambda\mathbf{H}$  has two atomic types  $\mathcal{N}$  and  $\mathcal{O}$  which name the set  $\mathbb{N}$  of natural numbers and the set  $\mathbb{O}$  of iteration templates, respectively. All other types are generated from these atoms by arrow formation. Thus if  $\sigma$  and  $\rho$  are types then  $(\sigma \rightarrow \rho)$  is a type. This is the syntactic analogue of the function space constructor.

The types of the calculus  $\lambda\mathbf{G}$  are those generated from  $\mathcal{N}$  alone. ■

(Strictly speaking, we should also add product types with the associated projection gadgets. These are not needed here so we may quietly ignore them.)

We use  $\rho, \sigma, \tau, \dots$  for types. We employ the usual conventions for omitting brackets, namely that which matches well with currying. We let

$$\tau' \text{ abbreviate } \tau \rightarrow \tau$$

so that

$$\tau'' \text{ abbreviates } (\tau \rightarrow \tau) \rightarrow (\tau \rightarrow \tau)$$

and  $\tau''', \tau^{iv}, \tau^v, \dots$  are higher order compounds of  $\tau$ .

Each type  $\tau$  has an orthodox interpretation  $\llbracket \tau \rrbracket$  generated from  $\llbracket \mathcal{N} \rrbracket = \mathbb{N}$  and  $\llbracket \mathcal{O} \rrbracket = \mathbb{O}rd$ . This is *not* the intended interpretation. The intention is that  $\mathcal{O}$  names the set of  $\mathbb{O}$  concrete iteration templates. In other words, the intention is that  $\mathcal{N}$  names the carrier of the initial object of **Peano** and  $\mathcal{O}$  names the carrier of the initial object of **Limit**. Thus  $\lambda\mathbf{H}$  will have two main semantics; the orthodox semantics generated from  $\llbracket \mathcal{O} \rrbracket = \mathbb{O}rd$ , and the intended semantics generated from  $\llbracket \mathcal{O} \rrbracket = \mathbb{O}$ . These are connected via the mediating arrow  $\mathfrak{D} \longrightarrow \mathfrak{D}rd$  in **Limit**. The calculus  $\lambda\mathbf{H}$  is an attempt to gather information about the initial structure  $\mathfrak{D}$ , but will also tell us something about  $\mathfrak{D}rd$ .

For a type  $\sigma$  let

$$\mathcal{L}(\sigma) \text{ abbreviate } (\mathcal{N} \rightarrow \sigma) \rightarrow \sigma$$

so that  $\llbracket \mathcal{L}(\sigma) \rrbracket = \mathcal{L}(\llbracket \sigma \rrbracket)$  where  $\llbracket \sigma \rrbracket = \mathbb{S}$ .

4.2 DEFINITION. For  $\lambda\mathbf{H}$  there are five constants with housing axioms

$$\underline{0} : \mathcal{N} \quad \underline{S} : \mathcal{N}' \quad \bar{0} : \mathcal{O} \quad \bar{S} : \mathcal{O}' \quad \text{Lim} : \mathcal{L}(\mathcal{O})$$

intended to name the attributes of  $\mathfrak{N}$  and  $\mathfrak{D}$  respectively.

For each type  $\sigma$  there are two constants with housing axioms

$$I_\sigma : \mathcal{N} \rightarrow \sigma'' \quad J_\sigma : \mathcal{O} \rightarrow \mathcal{L}(\sigma) \rightarrow \sigma''$$

which name iteration gadgets on  $\sigma$ .

For  $\lambda\mathbf{G}$  the only constants are  $\underline{0}, \underline{S}$ , and the  $I_\sigma$  where  $\sigma$  is a  $\lambda\mathbf{G}$ -type. ■

We distinguish between the natural numbers and the finite iteration templates. Later we will produce a term `fin` which explicitly matches these. The role of `Lim`,  $I_\sigma$ , and  $J_\sigma$  will be explained shortly. Terms are generated using an unlimited stock of identifiers.

4.3 DEFINITION. Each constant and each identifier is a raw term. If  $q$  and  $p$  are raw terms then  $(qp)$  is a raw term. If  $r$  is a raw term,  $\sigma$  a type, and  $y$  an identifier, then  $(\lambda y : \sigma . r)$  is a raw term. We often say ‘term’ when we mean ‘raw term’. ■

Brackets can be omitted from terms in the usual way to match with currying. For each pair of terms  $t, s$  and each  $m \in \mathbb{N}$  we generate a term  $t^m s$  by

$$t^0 s = s \quad t^{m+1} s = t(t^m s)$$

so that  $t^m s$  is the  $m$ -fold iterate of  $t$  on  $s$ . Using this notation we set

$$\underline{m} = \underline{S}^m \underline{0} \quad \bar{m} = \bar{S}^m \bar{0}$$

to produce the numeral  $\underline{m}$  and the finite iteration template  $\bar{m}$ . It is convenient to use

$$u, v, w, \dots \quad \text{for inhabitants of } \mathcal{N} \quad \alpha, \beta, \gamma, \dots \quad \text{for inhabitants of } \mathcal{O}$$

to aid readability.

A statement is a pair  $t : \tau$  where  $t$  is a term and  $\tau$  is a type. We call  $t$  the subject and  $\tau$  the predicate of the statement  $t : \tau$ . Thus, in Definition 4.2, each housing axiom is a statement. In other words, each constant comes with its own housing type. A declaration is a statement  $x : \sigma$  where  $x$  is an identifier. A context is a list

$$\Gamma = x_1 : \sigma_1, \dots, x_l : \sigma_l$$

of declarations. Such a context  $\Gamma$  is legal if the declared identifiers  $x_1, \dots, x_l$  are distinct. A Typing-judgement (or T-judgement for short)

$$\Gamma \vdash t : \tau$$

is a statement  $t : \tau$  in context  $\Gamma$ . We wish to read this as

Within the *legal* context  $\Gamma$  the *well-formed* term  $t$  inhabits the *acceptable* type  $\tau$

but such a reading requires some justification. For us every type is acceptable, and the legality of a context is easy to detect. The well-formedness of a term is more complicated.

Clause	Shape	Proviso
Axiom	$\Gamma \vdash k : \kappa$	$\Gamma$ is legal and $k : \kappa$ is an axiom
Projection	$\Gamma \vdash x : \sigma$	$\Gamma$ is legal and $x : \sigma$ occurs in $\Gamma$
Introduction	$\frac{\Gamma, y : \sigma \vdash r : \rho}{\Gamma \vdash t : \tau}$	$t = \lambda y : \sigma . r \quad \tau = \sigma \rightarrow \rho$
Elimination	$\frac{\Gamma \vdash q : \pi \rightarrow \tau \quad \Gamma \vdash p : \pi}{\Gamma \vdash qp : \tau}$	
Weakening	$\frac{\Gamma \vdash t : \tau}{\Gamma, x : \sigma \vdash t : \tau}$	$x$ is $\Gamma$ -fresh

Table 1: The derivation rules for  $\lambda H$

4.4 DEFINITION. A Typing-derivation (or T-derivation for short)

$$(T) \quad \Gamma \vdash t : \tau$$

is a finite rooted tree of judgements grown according to the rules of Table 1.

A typing-derivation for  $\lambda G$  is built up using only the types and terms of  $\lambda G$ . ■

For the calculus  $\lambda H$  a computation is an organized reduction on terms generated by redex removals and other 1-step reductions. As usual, a redex is a term

$$t^- = (\lambda y : \sigma . r)s$$

an abstraction applied to a term  $s$ . We wish to replace this by the immediate reduct

$$t^+ = r[y := s]$$

obtained by substituting  $s$  for all free occurrences of  $y$  in  $r$ . For such a pair

$$t^- \triangleright t^+$$

is a redex removal as in all  $\lambda$ -calculi. The other 1-step reductions are peculiar to  $\lambda H$ .

4.5 DEFINITION. For each type  $\sigma$  the constants  $I_\sigma$  and  $J_\sigma$  have the 1-step reductions

$$\begin{array}{ll} I_\sigma \underline{0}yx & \triangleright x & J_\sigma \bar{0}lyx & \triangleright x \\ I_\sigma (\underline{S}u)yx & \triangleright y(I_\sigma uyx) & J_\sigma (\bar{S}\alpha)lyx & \triangleright y(J_\sigma \alpha lyx) \\ & & J_\sigma (\underline{L}im p)lyx & \triangleright l(\lambda u : \mathcal{N} . J_\sigma (pu)lyx) \end{array}$$

where  $x, y, l, u, \alpha, p$  are arbitrary terms. These capture the intended meaning of  $I_\sigma$  and  $J_\sigma$  as iteration gadgets over  $\mathcal{N}$  and  $\mathcal{O}$ , respectively. ■

(If we add product types, then all recursors can be obtained from these iterators.)

Let  $\triangleright\triangleright$  be the transitive, structural closure of  $\triangleright$ . Thus each instance  $t^- \triangleright\triangleright t^+$  of  $\triangleright\triangleright$  arises from several redex removals and 1-step reductions submerged within the terms.

Clause	Shape	Remarks
(Axiom reduction)	$\frac{t^- \triangleright t^+}{t^- \triangleright\triangleright t^+}$	A 1-step reduction
(Redex removal)	$\frac{t^- \triangleright t^+}{t^- \triangleright\triangleright t^+}$	$t^- = (\lambda y : \sigma . r) s \quad t^+ = r[y := s]$
(Left application)	$\frac{q^- \triangleright\triangleright q^+}{q^- p \triangleright\triangleright q^+ p}$	
(Right application)	$\frac{p^- \triangleright\triangleright p^+}{qp^- \triangleright\triangleright qp^+}$	
(Abstraction)	$\frac{r^- \triangleright\triangleright r^+}{t^- \triangleright\triangleright t^+}$	$t^- = (\lambda y : \sigma . r^-) \quad t^+ = (\lambda y : \sigma . r^+)$
(Transitive composition)	$\frac{t^- \triangleright\triangleright t^0 \quad t^0 \triangleright\triangleright t^+}{t^- \triangleright\triangleright t^+}$	

Table 2: The computation rules for  $\lambda H$

#### 4.6 DEFINITION. A computation

$$(C) \quad t^- \triangleright\triangleright t^+$$

is a finite rooted tree of reductions grown according to the rules of Table 2. We call  $t^-$  the subject and  $t^+$  the object of the computation. ■

A computation is nothing more than an organized account of the usual informal way of reducing terms. Computations seem to ignore the typing discipline. In fact, all the types in a computation can be erased.

#### 4.7 THEOREM. (Subject Reduction) For a $T$ -derivation and computation

$$(T) \quad \Gamma \vdash t^- : \tau \quad (C) \quad t^- \triangleright\triangleright t^+$$

with a common subject, there is a derivation

$$(T \cdot C) \quad \Gamma \vdash t^+ : \tau$$

with a reduced subject. This can be obtained in a syntactic way from  $T$  and  $C$ .

Suppose we have a pair of terms

$$\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : \tau$$

in a common context with a common type. These two terms may look very different but still have the same interpretation. How can we recognize this similarity of meaning?

Certainly the reduction relation  $\triangleright$  must respect equality. Thus if we can move from  $t_1$  to  $t_2$  via a sequence of computations, then  $t_1$  and  $t_2$  should have the same interpretation.

Let  $\bowtie$  be the reflexive, symmetric, transitive closure of  $\triangleright$ . Each instance  $t_1 \bowtie t_2$  of this relation is witnessed by a collection of computations. By establishing a confluence result for  $\triangleright$ , we see that each instance of  $\bowtie$  needs only a pair of computations.

Each instance  $t_1 \bowtie t_2$  of this relation can be viewed as a proof that  $t_1$  and  $t_2$  have the same interpretation. This is one way of showing equality, but it is not powerful enough to capture all intended equalities. We need some equational reasoning so we extend the syntax of  $\lambda H$ . We attach to each type  $\tau$  a predicate  $\approx_\tau$  together with a formation rule

$$\frac{\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : \tau}{\Gamma \vdash (t_1 \approx_\tau t_2) : \text{Eq}(\tau)}$$

which allows us to produce a new syntactic category of equational statements over  $\tau$ . This construction merely shows that  $(t_1 \approx_\tau t_2)$  is correctly formed in  $\Gamma$ ; it does *not* show that  $t_1$  and  $t_2$  have the same meaning. To verify such an equality we must provide a proof of the fact. Thus we use a new syntactic category of Equational-judgements

$$\Gamma \vdash p : (t_1 \approx_\tau t_2)$$

where  $p$  is a proof term which encapsulates the required justification. These proof terms  $p$  can contain quite a lot of information. As it happens, we don't need this information here, so we will use a simplified version of these judgements.

For us an Equational-judgement (or E-judgement for short) has the shape

$$\Gamma \vdash [t_1 \approx t_2] : \tau$$

and is intended to convey the information that the equality judgement, as to the left

$$\Gamma \vdash (t_1 \approx_\tau t_2) : \text{Eq}(\tau) \quad \Gamma \vdash p : (t_1 \approx_\tau t_2)$$

is correctly formed and there is a proof term  $p$ , as to the right. Of course, we must provide a certificate of correctness for such a judgement.

**4.8 DEFINITION.** An Equational-derivation (or E-derivation for short)

$$(E) \quad \Gamma \vdash [t_1 \approx t_2] : \tau$$

is a finite rooted tree of various kinds of judgements and reductions grown according to the general rules of Table 3 together with the particular  $\omega$ -rule (to be described).  $\blacksquare$

These rules are self explanatory, and encapsulate the congruence and substitution properties of equality. They are use for equational reasoning in all applied  $\lambda$ -calculi. For  $\lambda H$  there is one more rule which encapsulate the notion equality for numeric functions.

**4.9 DEFINITION.** (The  $\omega$ -rule or external induction principle) Suppose both

$$\Gamma \vdash \phi : (\mathcal{N} \rightarrow \tau) \quad \Gamma \vdash \psi : (\mathcal{N} \rightarrow \tau)$$

are derivable. Suppose also that for each  $m \in \mathbb{N}$  the Equality-judgement, as to the left

$$\Gamma \vdash [\phi \underline{m} \approx \psi \underline{m}] : \tau \quad \Gamma \vdash [\phi \approx \psi] : (\mathcal{N} \rightarrow \tau)$$

is derivable. Then the Equality-judgement to the right is allowed.  $\blacksquare$

Clause	Shape
(Reflexive)	$\frac{\Gamma \vdash t : \tau}{\Gamma \vdash [t \approx t] : \tau}$
(Symmetric)	$\frac{\Gamma \vdash [s \approx t] : \tau}{\Gamma \vdash [t \approx s] : \tau}$
(Transitive)	$\frac{\Gamma \vdash [r \approx s] : \tau \quad \Gamma \vdash [s \approx t] : \tau}{\Gamma \vdash [r \approx t] : \tau}$
(Left application)	$\frac{\Gamma \vdash [t_1 \approx t_2] : (\sigma \rightarrow \rho) \quad \Gamma \vdash s : \sigma}{\Gamma \vdash [t_1 s \approx t_2 s] : \rho}$
(Right application)	$\frac{\Gamma \vdash t : (\sigma \rightarrow \rho) \quad \Gamma \vdash [s_1 \approx s_2] : \sigma}{\Gamma \vdash [t s_1 \approx t s_2] : \rho}$
(Abstraction)	$\frac{\Gamma, x : \sigma \vdash [r_1 \approx r_2] : \rho}{\Gamma \vdash [(\lambda x : \sigma. r_1) \approx (\lambda x : \sigma. r_2)] : (\sigma \rightarrow \rho)}$
(Reduction)	$\frac{\Gamma \vdash t : \tau \quad t \triangleright t'}{\Gamma \vdash [t \approx t'] : \tau}$
(Extensional)	$\frac{\Gamma \vdash t : (\sigma \rightarrow \rho)}{\Gamma \vdash [(\lambda x : \sigma. t x) \approx t] : (\sigma \rightarrow \rho)}$
(Weakening)	$\frac{\Gamma \vdash [t_1 \approx t_2] : \tau}{\Gamma, x : \sigma \vdash [t_1 \approx t_2] : \tau}$

Table 3: The general equational reasoning rules

This is an infinitary rule of derivation.

$$\frac{\Gamma \vdash [\phi_0 \approx \psi_0] : \tau \quad \Gamma \vdash [\phi_1 \approx \psi_1] : \tau \quad \dots \quad \Gamma \vdash [\phi_m \approx \psi_m] : \tau \quad \dots \quad (m \in \mathbb{N})}{\Gamma \vdash [\phi \approx \psi] : (\mathcal{N} \rightarrow \tau)}$$

To use it rule we need the existence of infinitely many smaller Equational-derivations. Since we can not possibly write down each one separately, we will be forced to describe some scheme which generates them. This will impose some uniformity on the numerator derivations. In a more powerful reasoning system this rule will be simulated by some finite construction. But again, this will impose some uniformity on its use.

## 5 Arithmetic in $\lambda\mathbf{H}$

The numeric part  $\lambda\mathbf{G}$  of  $\lambda\mathbf{H}$  can be used to name (and compute) a vast collection of numeric functions. Gödel's T provides a mechanism for verifying certain relationships between these functions. The correctness of these results can be seen by observing the

interaction with the standard semantics. In the same way  $\lambda\mathbf{H}$  ought to name a collection of iteration gadgets. The correctness of these will be less clear, partly because we don't have a precise notion of iteration template that is independent of  $\lambda\mathbf{H}$ , and partly because of the underlying semantic category is more complicated. However, we can make some headway in determining the meaning of certain terms of  $\lambda\mathbf{H}$ .

As earlier, for each  $m \in \mathbb{N}$  we set

$$\underline{m} = \underline{S}^m \underline{0} \quad \overline{m} = \overline{S}^m \overline{0}$$

to obtain simple terms where

$$\vdash \underline{m} : \mathcal{N} \quad \vdash \overline{m} : \mathcal{O}$$

hold. The terms  $\underline{m}$  are the numerals which name the natural numbers. The terms  $\overline{m}$  name the finite iteration templates, the finite ordinals. For the time being it is safer to keep  $\underline{m}$  and  $\overline{m}$  separate.

5.1 DEFINITION. Let

$$\begin{aligned} \underline{A} &= \lambda v, u : \mathcal{N} . (\mathbf{I}_{\mathcal{N}u}) \underline{S}v & \overline{A} &= \lambda \beta, \alpha : \mathcal{O} . (\mathbf{J}_{\mathcal{O}\alpha} \mathbf{Lim}) \overline{S}\beta \\ \underline{M} &= \lambda v, u : \mathcal{N} . (\mathbf{I}_{\mathcal{N}u}) (\underline{A}v) \underline{0} & \overline{M} &= \lambda \beta, \alpha : \mathcal{O} . (\mathbf{J}_{\mathcal{O}\alpha} \mathbf{Lim}) (\lambda \gamma : \mathcal{O} \overline{A}\gamma \beta) \overline{0} \\ \underline{E} &= \lambda v, u : \mathcal{N} . (\mathbf{I}_{\mathcal{N}u}) (\underline{M}v) \underline{1} & \overline{E} &= \lambda \beta, \alpha : \mathcal{O} . (\mathbf{J}_{\mathcal{O}\alpha} \mathbf{Lim}) (\overline{M}\beta) \overline{1} \end{aligned}$$

to produce three terms  $\vdash \underline{N} : \mathcal{N} \rightarrow \mathcal{N} \rightarrow \mathcal{N}$  and three terms  $\vdash \overline{O} : \mathcal{O} \rightarrow \mathcal{O} \rightarrow \mathcal{O}$ . ■

It is routine to produce the indicated derivations. Furthermore we find that

$$\underline{A} \underline{n} \underline{m} \triangleright \underline{n} + \underline{m} \quad \underline{M} \underline{n} \underline{m} \triangleright \underline{n} \times \underline{m} \quad \underline{E} \underline{n} \underline{m} \triangleright \underline{n}^{\underline{m}}$$

for  $m, n \in \mathbb{N}$ . The  $\mathcal{N}$ -terms represent the standard arithmetical operations on  $\mathbb{N}$ . Other natural number gadgets can be simulated by tracking their recursive constructions.

The job of  $\overline{A}, \overline{M}, \overline{E}$  is to simulate the analogues of ordinal addition, multiplication, and exponentiation for iteration templates. We will see how to justify these terms later.

The calculus  $\lambda\mathbf{G}$  can not make any explicit reference to an ordinal. However, the local effect of many ordinals can be simulated in  $\lambda\mathbf{G}$ .

5.2 THEOREM. *For each type  $\sigma$  generated from  $\mathcal{N}$  and for each ordinal  $\alpha < \epsilon_0$ , there is a term  $\alpha_\sigma$  of  $\lambda\mathbf{G}$  which simulates the effect of  $\mathbf{J}_\sigma \alpha$ .*

We need not prove this here, but we will give a couple of examples.

5.3 DEFINITION. Let  $\mathbf{fin} = \lambda u : \mathcal{N} . \mathbf{I}_{\mathcal{O}u} \overline{S}\overline{0}$  and  $\omega = \mathbf{Lim} \mathbf{fin}$  to obtain terms  $\vdash \mathbf{fin} : \mathcal{N} \rightarrow \mathcal{O}$  and  $\vdash \omega : \mathcal{O}$ .

For each type  $\sigma$  let

$$\omega_\sigma = \lambda l : \mathcal{L}(\sigma), y; \sigma', x : \sigma . l(\lambda u : \mathcal{N} . \mathbf{I}_{\sigma} u y x)$$

to obtain a term  $\vdash \omega_\sigma : \mathcal{L}(\sigma) \rightarrow \sigma''$ . ■

Trivially  $\text{fin } \underline{m} \triangleright \bar{m}$  for each  $m \in \mathbb{N}$ . Thus  $\text{fin}$  names the insertion of  $\mathbb{N}$  into  $\mathbb{O}$ .  
 If  $\sigma$  is a type generated from  $\mathcal{N}$  then  $\omega_\sigma$  is a term of  $\lambda\mathbf{G}$ , and simulates  $\omega$  over  $\sigma$ .

**5.4 LEMMA.** *For each type  $\sigma$  we have  $\vdash [\text{J}_\sigma \omega \approx \omega_\sigma] : (\mathcal{L}(\sigma) \rightarrow \sigma)$ .*

**Proof.** This is a simple exercise in the use of the  $\omega$ -rule, but let's look at the details.  
 For each  $m \in \mathbb{N}$  we have

$$\text{J}_\sigma(\text{fin } \underline{m})lyx \triangleright y^m x \quad \text{I}_\sigma \underline{m}yx \triangleright y^m x$$

and hence

$$\Gamma \vdash [\text{J}_\sigma(\text{fin } \underline{m})lyx \approx \text{I}_\sigma \underline{m}yx] : \sigma$$

where

$$\Gamma = l : \mathcal{L}(\sigma), y : \sigma'', x : \sigma'$$

is the context. Thus

$$\Gamma \vdash [\lambda u : \mathcal{N}. \text{J}_\sigma(\text{fin } u)lyx \approx \lambda u : \mathcal{N}. \text{I}_\sigma uyx] : (\mathcal{N} \rightarrow \sigma)$$

follows by the  $\omega$ -rule. Next

$$\text{J}_\sigma \omega lyx \triangleright l(\lambda u : \mathcal{N}. \text{J}_\sigma(\text{fin } u)lyx) \quad \omega_\sigma lyx \triangleright l(\lambda u : \mathcal{N}. \text{I}_\sigma uyx)$$

so that

$$\Gamma \vdash [\text{J}_\sigma \omega lyx \approx \omega_\sigma lyx] : \sigma$$

and hence uses of (Abstraction Extensional) of Table 3 gives the required result. ■

The next interesting ordinal can be treated in a similar fashion.

**5.5 DEFINITION.** Let  $\text{eps} = \lambda u : \mathcal{N}. \text{I}_{\mathcal{O}u}(\bar{\mathbf{E}}\omega)\omega$  to obtain a term  $\vdash \text{eps} : \mathcal{N} \rightarrow \mathcal{O}$ .

Using recursion on  $r$  with variation of the parameters  $\sigma$  and  $l$  set

$$\epsilon[0, \sigma, l] = \omega_\sigma l \quad \epsilon[r', \sigma, l] = \epsilon[r, \sigma', l'](\omega_\sigma l)$$

(for each  $r < \omega$  type  $\sigma$ , and term  $l$ ). ■

It is easy to check that if  $\Lambda \vdash l : \mathcal{L}(\sigma)$  for some legal context  $\Lambda$  then  $\Lambda \vdash \epsilon[r, \sigma, l] : \mathcal{O}$ .  
 For a type  $\sigma$  generated from  $\mathcal{N}$ , the terms  $\epsilon[r, \sigma, l]$  belong to  $\lambda\mathbf{G}$ .

**5.6 LEMMA.** *Let  $\Lambda$  be a context with  $\Lambda \vdash l : \mathcal{L}(\sigma)$  for some type  $\sigma$  and term  $l$ . Then*

$$\Lambda \vdash [\text{J}_\sigma(\text{eps } \underline{r})l \approx \epsilon[r, \sigma, l]] : \sigma''$$

*holds for all  $r \in \mathbb{N}$ .*

The proof of this is an easy induction over  $r$  (with allowable variation of  $\sigma$  and  $l$ ). The result can be used to show that ordinals below  $\epsilon_0$  can be simulated locally throughout  $\lambda\mathbf{G}$ . However, The ordinal  $\epsilon_0$  can not, for this measure the limits of  $\lambda\mathbf{G}$ . The term  $\epsilon = \text{Lim eps}$  names the ordinal  $\epsilon_0$ , and this where  $\lambda\mathbf{H}$  starts to outstrip  $\lambda\mathbf{G}$ .

There are terms of  $\lambda\mathbf{H}$  which name much larger ordinals. In this section we exhibit some of these, and later in Section 8 we will calculate the corresponding ordinal values.

5.7 DEFINITION. Let

$$\text{Fix} = \lambda f : \mathcal{O}', \alpha : \mathcal{O} . (\text{J}_{\mathcal{O}}\omega\text{Lim})f(\bar{\mathcal{S}}\alpha) \quad \text{Next} = \text{Fix}(\bar{\mathcal{E}}\omega)$$

to produce terms  $\vdash \text{Fix} : \mathcal{O}''$  and  $\vdash \text{Next} : \mathcal{O}'$ . ■

A simple calculation gives

$$\text{Next } \omega \triangleright (\text{J}_{\mathcal{O}}\omega\text{Lim})(\bar{\mathcal{E}}\omega)(\bar{\mathcal{S}}\omega) \triangleright \text{Lim}(\lambda u : \mathcal{N} . (\text{J}_{\mathcal{O}}(\text{fin } u)\text{Lim})(\bar{\mathcal{E}}\omega)(\bar{\mathcal{S}}\omega))$$

which you should compare with

$$\epsilon = \text{Lim eps} \triangleright \text{Lim}(\lambda u : \mathcal{N} . (\text{I}_{\mathcal{O}}u)(\bar{\mathcal{E}}\omega)\omega)$$

from above. The two terms both name  $\epsilon_0$  but via different fundamental sequences.

The two terms  $(\text{J}_{\mathcal{O}}(\text{fin } u)\text{Lim})$  and  $(\text{I}_{\mathcal{O}}u)$  are pointwise equivalent. More generally, given any judgement  $\Lambda \vdash l : \mathcal{L}(\sigma)$  we may obtain

$$\Lambda \vdash [(\lambda u : \mathcal{N} . \text{J}_{\sigma}(\text{fin } u)l) \approx \text{I}_{\sigma}] : (\mathcal{N} \rightarrow \sigma'')$$

and hence many a use of  $\text{I}_{\sigma}$  can be replaced by a use of  $\text{J}_{\sigma}$ .

Remember that the types  $\mathcal{O}^{(l)}$  are generated by

$$\mathcal{O}^{(0)} = \mathcal{O} \quad \mathcal{O}^{(r+1)} = \mathcal{O}^{(r)'}$$

for each  $r \in \mathbb{N}$ . We write  $\text{J}^{(l)}$  for  $\text{J}_{\sigma}$  where  $\sigma$  is  $\mathcal{O}^{(l)}$ . In particular, we write  $\text{J}$  for  $\text{J}^{(0)} = \text{J}_{\mathcal{O}}$ . This will save a bit of clutter. We also need an associated term naming a limit creator.

Recall that in Section 3 we used the limit lifter  $\uparrow_{\sigma} : \mathcal{L}(\mathbb{S}) \rightarrow \mathcal{L}(\mathbb{S}')$  on a set  $\mathbb{S}$ .

5.8 DEFINITION. For each type  $\sigma$  let

$$\uparrow_{\sigma} = \lambda l : \mathcal{L}(\sigma), q : \mathcal{N} \rightarrow \sigma', x : \sigma . l(\lambda u : \mathcal{N} . qux)$$

to produce a term  $\vdash \uparrow_{\sigma} : \mathcal{L}(\sigma) \rightarrow \mathcal{L}(\sigma')$ . ■

Using these terms  $\uparrow_{\sigma}$  we generate terms  $\text{L}^{(l)}$  by

$$\text{L}^{(0)} = \text{Lim} \quad \text{L}^{(r+1)} = \uparrow_{\sigma} \text{L}^{(r)} \quad \text{where } \sigma = \mathcal{O}^{(r)}$$

(for  $r \in \mathbb{N}$ ), and we let  $\text{L} = \text{L}^{(0)} = \text{Lim}$ . As with concrete limit lifting, this notation omits a lot of information. It can be checked that

$$\vdash \text{J}^{(l)} : \mathcal{O} \rightarrow \mathcal{L}(\mathcal{O}^{(l)}) \rightarrow \mathcal{O}^{(l+2)} \quad \vdash \text{L}^{(l)} : \mathcal{L}(\mathcal{O}^{(l)})$$

so that  $\Lambda \vdash \text{J}^{(l)}\alpha\text{L}^{(l)} : \mathcal{O}^{(l+2)}$  when  $\Lambda \vdash \alpha : \mathcal{O}$ . In particular,  $\Lambda \vdash \text{J}\alpha\text{L} : \mathcal{O}''$  for such  $\alpha$ .

5.9 DEFINITION. For each  $l < \omega$  let

$$[l] = \lambda y : \mathcal{O}^{(l+1)}, y_l : \mathcal{O}^{(l)}, \dots, x : \mathcal{O} . \text{Fix } Zx \quad \text{where } Z = \lambda \alpha : \mathcal{O} . (\text{J}^{(l)}\alpha\text{L}^{(l)})yy_k \cdots y_1\bar{0}$$

using the auxiliary term on the right. ■

We have  $\vdash [l] : \mathcal{O}^{(l+2)}$ , hence  $[l]$  names an operator to be described this in Section 7.

5.10 DEFINITION. Let

$$\Delta[0] = \omega \quad \Delta[1] = \text{Next}\omega \quad \Delta[l+2] = [l] \cdots [0]\text{Next}\omega$$

for each  $l < \omega$ . ■

Almost trivially  $\vdash \Delta[r] : \mathcal{O}$  so that  $\Delta[r]$  names some ordinal  $\Delta[r]$ . In particular,  $\Delta[0] = \omega$  and  $\Delta[1] = \epsilon_0$ . You might calculate  $\Delta[2]$  and  $\Delta[3]$ . We show how to calculate all  $\Delta[l]$  in Section 8. Observe that  $\Delta[r]$  is built using  $\text{J}^{(0)}, \text{J}^{(1)}, \dots, \text{J}^{(r)}$  within a restricted part of the type hierarchy on  $\mathcal{O}$  (and  $\mathcal{N}$ ). Thus it should be possible to put an upper bound on the complexity of the iteration template and the ordinal named by  $\Delta[r]$ .

## 6 Canonical notations

How can we use a term  $\vdash \alpha : \mathcal{O}$  as a notation for an ordinal or an iteration template? For a limit structure  $\mathfrak{A} = (\mathbb{A}, a, A, \mathcal{A})$  such a term should determine an element  $\mathfrak{A}(\alpha) \in \mathbb{A}$ . How is this produced? Applying this process to the limit structure  $\mathfrak{Ord}$  on  $\mathbb{Ord}$  should give the ordinal value  $\llbracket \alpha \rrbracket$  of the term. How can we calculate this? One way is to normalize the term, and then inspect the shape. This is not a practical method, for the normalized term could be very big (and take a long time to produce). We need a different method.

**6.1 DEFINITION.** The algebraic terms are generated recursively as follows.

(Base)  $\bar{0}$  is an algebraic term.

(Step) If  $\alpha$  is an algebraic term then so is  $\bar{S}\alpha$ .

(Leap) If  $\vdash p : (\mathcal{N} \rightarrow \mathcal{O})$  is derivable and for each  $m \in \mathbb{N}$  there is an algebraic term  $\alpha_m$  with  $\vdash [pm \approx \alpha_m] : \mathcal{O}$ , then  $\text{Limp}$  is an algebraic term.

In particular,  $\vdash \alpha : \mathcal{O}$  holds for each algebraic term  $\alpha$ . ■

For an algebraic term  $\alpha$  the element  $\mathfrak{A}(\alpha)$  is easy to produce; we mimic the construction of the spine elements of  $\mathfrak{A}$ . For instance, suppose  $\alpha = \text{Limp}$  where for each  $m \in \mathbb{N}$  we have  $\vdash [pm \approx \alpha_m] : \mathcal{O}$  with  $\alpha_m$  algebraic. By recursion, for each such  $m$  we have an element  $\mathfrak{A}(\alpha_m) \in \mathbb{A}$ , so we set

$$\mathfrak{A}(\alpha) = \mathcal{A}(m \mapsto \mathfrak{A}(\alpha_m))$$

to produce the required element of  $\mathbb{A}$ .

Almost trivially for each  $m \in \mathbb{N}$  the term  $\bar{m}$  is algebraic and  $\llbracket \bar{m} \rrbracket = m$ . After that we see that  $\omega$  is algebraic with  $\llbracket \omega \rrbracket = \omega$ . It can be shown that  $\llbracket \epsilon \rrbracket = \epsilon_0$ . However, most terms we meet are not algebraic.

**6.2 DEFINITION.** A term  $\vdash \alpha : \mathcal{O}$  is *canonical* if there is an algebraic term  $\beta$  such that  $\vdash [\alpha \approx \beta] : \mathcal{O}$  is derivable. ■

For such a term  $\alpha$  we set  $\mathfrak{A}(\alpha) = \mathfrak{A}(\beta)$  and use the structure of  $\beta$  to determine this element of  $\mathfrak{A}$ . (In fact, this trick is embedded in the construction of  $\mathfrak{A}(\text{Limp})$  above.)

We will show that many canonical terms can be generated.

**6.3 DEFINITION.** For each  $k < \omega$  the class  $\mathcal{C}_k$  of terms is generated by recursion on  $k$ .

(0) The class  $\mathcal{C}_0$  is exactly the class of canonical terms.

(k+1) A term  $C$  is in  $\mathcal{C}_{k+1}$  if  $\vdash C : \mathcal{O}^{(k+1)}$  and  $CB \in \mathcal{C}_k$  for each  $B \in \mathcal{C}_k$ .

Thus  $C \in \mathcal{C}_{k+1}$  exactly when  $CA_k \cdots A_0 \in \mathcal{C}_0$  for each  $A_k \in \mathcal{C}_k, \dots, A_0 \in \mathcal{C}_0$ . ■

For example,  $\bar{S} \in \mathcal{C}_1$ . Shortly we will see more interesting examples. An easy exercise shows that each class  $\mathcal{C}_k$  is closed under  $\approx$ , that is  $C \in \mathcal{C}_k$  whenever  $\vdash C : \mathcal{O}^{(k)}$  and  $\vdash [C \approx B] : \mathcal{O}^{(k)}$  for some  $B \in \mathcal{C}_k$ .

**6.4 LEMMA.** For each  $k < \omega$ , if  $\alpha \in \mathcal{C}_0$  then  $J^{(k)}\alpha L^{(k)} \in \mathcal{C}_{k+2}$ .

**Proof.** It suffices to show  $J^{(k)}\alpha L^{(k)} \in \mathcal{C}_{k+2}$  for each algebraic  $\alpha$ . To this end we fix  $C \in \mathcal{C}_{k+1}, B \in \mathcal{C}_k$  and show  $J^{(k)}\alpha L^{(k)}CB \in \mathcal{C}_k$  by induction on the structure of  $\alpha$ .

Since  $J^{(k)}\bar{0}L^{(k)}CB \triangleright B$  the base case,  $\alpha = \bar{0}$ , is immediate.

For the induction step  $\alpha \mapsto \bar{S}\alpha$  we have

$$J^{(k)}(\bar{S}\alpha)L^{(k)}CB \triangleright C(J^{(k)}\alpha L^{(k)}CB)$$

so the required result follows by the induction hypothesis and the given property of  $C$ .

For the induction leap  $\alpha = \text{Limp}$  we show

$$J^{(k)}\alpha L^{(k)}CBA_k \cdots A_1\zeta \in \mathcal{C}_0$$

where  $A_i \in \mathcal{C}_i$  for  $k > i \geq 1$  and  $\zeta \in \mathcal{C}_0$ . Let

$$q = \lambda u : \mathcal{N} . J^{(k)}(pu)L^{(k)}CBA_k \cdots A_1\zeta$$

so that  $\vdash q : \mathcal{N} \rightarrow \mathcal{O}$ . For each  $m \in \mathbb{N}$  the term  $p\bar{m}$  is canonical so that

$$q\bar{m} \triangleright J^{(k)}(p\bar{m})L^{(k)}CBA_k \cdots A_1\zeta \in \mathcal{C}_0$$

by the induction hypothesis. Also

$$J^{(k)}\alpha L^{(k)}CBA_k \cdots A_1\zeta \triangleright L^{(k)}(\lambda u : \mathcal{N} . J^{(k)}(pu)L^{(k)}CB)A_k \cdots A_1\zeta \triangleright \text{Lim}q$$

which leads to the required result. ■

We saw above that  $\bar{S} \in \mathcal{C}_1$ . Using this last result we can generate a sequence of increasingly more interesting examples of members of the classes  $\mathcal{C}_k$ .

**6.5 LEMMA.** (a) For each canonical term  $\beta$  the three terms  $\bar{A}\beta, \bar{M}\beta, \bar{E}\beta$  are in  $\mathcal{C}_1$ .

(b)  $\text{Next} \in \mathcal{C}_1$

(c) For each  $l < \omega$  we have  $[l] \in \mathcal{C}_{l+2}$ .

**Proof.** (a) For each  $\beta, \alpha \in \mathcal{C}_0$  we have

$$\bar{A}\beta\alpha \triangleright J\alpha L\bar{S}\beta \quad \bar{M}\beta\alpha \triangleright J\alpha L(--) \bar{0} \quad \bar{E}\beta\alpha \triangleright J\alpha L(\bar{M}\beta)\bar{1}$$

where

$$(--) \text{ is } (\lambda\gamma : \mathcal{O}\bar{A}\gamma\beta)$$

for the central case.

The left hand case follows by Lemma 6.4.

For the central case we have

$$(--)\gamma \triangleright \bar{A}\gamma\beta \in \mathcal{C}_0$$

by the left hand part, and hence this case follows by Lemma 6.4.

The right hand case follows by another use of Lemma 6.4.

(b) For  $\zeta \in \mathcal{C}_0$  we have  $\text{Next}\zeta \triangleright J\omega L(\bar{E}\omega)(\bar{S}\zeta)$  so Lemma 6.4 and (a) give the result.

(c) Since

$$[l]CBA_{l-1} \cdots A_1A_0 \triangleright (J\omega L)ZA_0 \quad \text{where} \quad ZA_0 \triangleright J^{(l)}\alpha L^{(l)}CBA_{l-1} \cdots A_10$$

the result follows by two uses of Lemma 6.4, to get  $Z \in \mathcal{C}_1$  and  $(J\omega L)ZA_0 \in \mathcal{C}_0$ . ■

We have an immediate consequence of definition 5.10 and Lemma 6.5 .

6.6 THEOREM. For each  $k < \omega$  the term  $\Delta[k]$  is canonical.

Using the terms  $\Delta[\cdot]$  on the limit structure  $\mathfrak{Ord}$  produces a sequence through  $\mathfrak{Ord}$ . To estimate these ordinals we must study certain classes of ordinal functions.

The following two sections were part of the original paper. Since then [18, 19] have been published and these cover quite a bit of the same material. Also, the proof given here of Theorem 7.20 is not complete.

## 7 Helpful ordinal functions

In Section 5 we produced terms **Fix**, **Next**,  $[l]$  of  $\lambda\mathbf{H}$  and used these to obtain terms  $\vdash \Delta[l] : \mathcal{O}$  for each  $l < \omega$ . It is clear that  $\Delta[l]$  names a certain ordinals, but it is not clear what this is (unless  $l < 4$ , say). To determine  $\Delta[l]$  we unravel its construction and figure out what its various components are doing. We have to look at certain higher level ordinal functions, which is not something that is done often.

Most systems of ordinal notations use the idea of a normal function  $f : \mathfrak{Ord}'$ . Here it is more convenient to use a slightly broader class of functions. As usual we search for fixed points of such functions. To do this we represent the value at  $\alpha$  in terms of the  $\alpha$ -fold iterate of a certain other function (perhaps higher order). These other functions have appropriate properties which enable certain calculations to be carried out.

7.1 DEFINITION. The chain  $\mathfrak{Ord}^{(\cdot)}$  of spaces is generated by

$$\mathfrak{Ord}^{(0)} = \mathfrak{Ord} \quad \mathfrak{Ord}^{(r+1)} = \mathfrak{Ord}^{(r)'}$$

for each  $r < \omega$  (where  $\mathfrak{S}'$  is  $\mathfrak{S} \rightarrow \mathfrak{S}$  for each set  $\mathfrak{S}$ ). ■

Thus  $\mathfrak{Ord}^{(0)}$  is just the space  $\mathfrak{Ord}$  of ordinals, and  $\mathfrak{Ord}^{(1)}$  is the space  $\mathfrak{Ord}'$  of ordinal functions. Thereafter, at higher levels  $l$ , we see that  $\mathfrak{Ord}^{(l+2)}$  is a space of higher order ordinal functions. Almost every gadget we look at inhabits one of these spaces  $\mathfrak{Ord}^{(r)}$ . We look at certain special subsets  $\mathfrak{S}$  of  $\mathfrak{Ord}^{(l+1)}$  for arbitrary  $l$ . We want these sets  $\mathfrak{S}$  to be closed under ordinal iteration, and to organize that we use a more general notion.

The space  $\mathfrak{Ord}$  is linearly ordered and carries an actual supremum operation  $\bigvee$  (which, for our purposes, acts only on countable subsets). This can be lifted to higher levels.

7.2 DEFINITION. (base) For each non-empty, countable subset  $\mathcal{G} \subseteq \mathfrak{Ord}'$  the function  $\bigvee \mathcal{G} : \mathfrak{Ord}'$  is given by

$$(\bigvee \mathcal{G})\zeta = \bigvee \{g\zeta \mid g \in \mathcal{G}\}$$

(for  $\zeta \in \mathfrak{Ord}$ ). We call this function  $\bigvee \mathcal{G}$  the **pointwise supremum** of  $\mathcal{G}$ .

(raise) For each  $l < \omega$  and each non-empty, countable subset  $\mathcal{G} \subseteq \mathfrak{Ord}^{(l+2)}$  the function  $\bigvee \mathcal{G} : \mathfrak{Ord}^{(l+2)}$  is given by

$$(\bigvee \mathcal{G})g = \bigvee \{Gg \mid G \in \mathcal{G}\}$$

(for  $g \in \mathfrak{Ord}^{(l+1)}$ ). We call this function  $\bigvee \mathcal{G}$  the **pointwise supremum** of  $\mathcal{G}$ . ■

In (base), on the right ‘ $\bigvee$ ’ is the actual supremum operation known to exist on  $\text{Ord}$  and which is being lifted, whereas on the left, ‘ $\bigvee$ ’ is the pointwise supremum being constructed on  $\text{Ord}'$ . The construction then proceeds by recursion up the levels. Again in (raise), on the right ‘ $\bigvee$ ’ is the operation known to exist on a lower level and which is being lifted, whereas on the left, ‘ $\bigvee$ ’ is the pointwise supremum being constructed.

This definition does *not* require an ordering on  $\text{Ord}^{(l+1)}$ ; the term ‘pointwise supremum’ is merely suggestive. However, we will see that for many uses of this operation there is an underlying ordering, and then the pointwise supremum is an actual supremum. Nevertheless, we need to take some care with this notion, for we might be tempted to assume that it always behaves like an actual supremum.

**7.3 DEFINITION.** For each  $g : \text{Ord}^{(l+1)}$ , the ordinal iterates  $g^\bullet$  of  $g$  are generated by

$$g^0 = id \quad g^{\alpha+1} = g \circ g^\alpha \quad g^\lambda = \bigvee \{g^\alpha \mid \alpha < \lambda\}$$

for the usual  $\alpha$  and  $\lambda$ . (Here  $id$  is the identity function on  $\text{Ord}^{(l)}$ .) ■

When  $l = 0$  this is the usual construction of the ordinal iterates of  $g : \text{Ord}'$ . The same process can be transferred to higher levels using lifted pointwise suprema. Often when such iterates occur there is a tacit requirement that, in the construction of  $g^\lambda$ , the set  $\{g^\alpha \mid \alpha < \lambda\}$  is an ascending chain (in some sense). This will be the case for all uses here.

**7.4 DEFINITION.** For each  $l < \omega$  a class  $\mathbb{S} \subseteq \text{Ord}^{(l+1)}$  is **smooth** if  $f \circ g \in \mathbb{S}$  for each  $f, g \in \mathbb{S}$ , and  $\bigvee \mathcal{G} \in \mathbb{S}$  for each non-empty and countable  $\mathcal{G} \subseteq \mathbb{S}$ . ■

If  $g \in \mathbb{S}$  where  $\mathbb{S}$  is smooth then  $g^\alpha \in \mathbb{S}$  for each non-zero ordinal  $\alpha$ .

## Low level functions

We use two classes of functions, isolated by combinations of simple restrictions.

**7.5 DEFINITION.** A function  $g : \text{Ord}'$  is, respectively

- |      |                       |   |
|------|-----------------------|---|
| (i)  | inflationary          | if $\alpha \leq g\alpha$  |
| (si) | strictly inflationary | if $\alpha < g\alpha$   |
| (m)  | monotone              | if $\beta \leq \alpha \Rightarrow g\beta \leq g\alpha$              |
| (sm) | strictly monotone     | if $\beta < \alpha \Rightarrow g\beta < g\alpha$                    |
| (b)  | big                   | if $\omega^\alpha \leq g\alpha$ (except possibly for $\alpha = 0$ ) |
| (sb) | strictly big          | if $g\alpha$ is critical  |
| (c)  | continuous            | if $g(\bigvee A) = \bigvee g[A]$                                    |

for all ordinals  $\alpha, \beta$ , and all non-empty countable sets  $A$  of ordinals. ■

The two properties (b, sb) are not often named, but often used. The three implications  $si \Rightarrow i$ ,  $sm \Rightarrow i + m$ , and  $i + sb \Rightarrow b$  hold. If  $g$  is continuous then  $g\lambda = \bigvee \{g\alpha \mid \alpha < \lambda\}$  for each limit ordinal  $\lambda$ . In general this is not enough to ensure continuity, but it is for monotone functions. We are concerned almost entirely with the class  $\text{IM}$  of functions that are both inflationary and monotone.

**7.6 DEFINITION.** An ordinal function is, respectively, **fruitful** or **normal** or **helpful** if it is  $i + m + b + c$  or  $sm + b + c$  or  $si + m + sb$ , accordingly. ■

These normal functions are just the usual normal functions which are big. It turns out that *strict* monotonicity is rather too restrictive so we use the larger class of fruitful functions. The helpful functions form a crucial technical device.

There are three main classes of ordinal functions that we use: general functions, fruitful functions (and normal functions), and helpful functions. Usually we write

$$f \text{ for a fruitful function} \quad g \text{ for a general function} \quad h \text{ for a helpful function}$$

to indicate which kind of function is being used. There will be times when these conventions are broken, but then the context will make clear what is meant.

**7.7 LEMMA.** *The class  $\mathbb{IM}$  is smooth.*

The class  $\mathbb{IM}$  can be partially ordered using the pointwise comparison.

$$f \leq g \iff (\forall \alpha : \text{Ord})[f\alpha \leq g\alpha]$$

With this we find that for each non-empty, countable subset  $\mathcal{G} \subseteq \mathbb{IM}$  the pointwise supremum  $\bigvee \mathcal{G}$  is the actual supremum. By intention each smooth class is closed under (non-zero) ordinal iterates. When  $g \in \mathbb{IM}$  the family  $\{g^\alpha \mid \alpha \text{ an ordinal}\}$  of iterates is an ascending chain. This helps with calculations involving iterates.

**7.8 LEMMA.** *If  $g \in \mathbb{IM}$  then  $g^\alpha \circ g^\beta = g^{\beta+\alpha}$  for all  $\alpha, \beta \in \text{Ord}$ .*

**Proof.** We proceed by induction on  $\alpha$ . The base case and the induction step are easy. For the leap to a limit ordinal  $\lambda$  consider any ordinal  $\zeta$  and let  $\eta = g^\beta \zeta$ . Then

$$g^{\beta+\lambda} \zeta = \bigvee \{g^\gamma \zeta \mid \gamma < \beta + \lambda\} = \bigvee \{g^{\beta+\alpha} \zeta \mid \alpha < \lambda\} = \bigvee \{g^\alpha \eta \mid \alpha < \lambda\} = g^\lambda \eta = (g^\lambda \circ g^\beta) \zeta$$

as required. The second equality holds since the functions  $g^\bullet$  form an ascending chain. The third equality uses the induction hypothesis. ■

Limit iterates of  $g \in \mathbb{IM}$  need not be strictly monotone. Suppose  $\lambda$  is additively critical. For each ordinal  $\zeta$  and ordinal  $\alpha < \lambda$  we have  $g^\lambda(g^\alpha \zeta) = (g^\lambda \circ g^\alpha) \zeta = g^{\alpha+\lambda} \zeta = g^\lambda \zeta$  and so  $g^\lambda$  is constant between  $\zeta$  and  $g^\alpha \zeta$ . If  $g$  is continuous then  $g^\lambda = g^\mu$  for some  $\mu \geq \lambda \cdot \omega$ .

The fruitful functions should be familiar. Each is a normal functions modified to be constant for certain stretches of ordinals. The helpful functions are not so familiar.

**7.9 LEMMA.** *The fruitful functions and the helpful functions form smooth classes.*

## Helpful low level functions

Let  $\mathbb{H}^{(1)} \subseteq \text{Ord}^{(1)}$  be the class of helpful functions. We know that  $\mathbb{H}^{(1)}$  is smooth but how do these functions help? We use the interpretation of the term **Fix**.

7.10 DEFINITION. Let  $\mathbf{Fix} : \mathbb{Ord}''$  be the function given by

$$\mathbf{Fix} f \zeta = f^\omega(\zeta + 1)$$

for each (fruitful) function  $f : \mathbb{Ord}'$  and ordinal  $\zeta$ . ■

Although  $\mathbf{Fix}$  is defined to act on any ordinal function, we use it only on fruitful functions. For such a function  $f$  and ordinal  $\zeta$  let  $\zeta[r] = f^r(\zeta + 1)$  for  $r < \omega$ . Then

$$\zeta < \zeta[0] \leq \dots \leq \zeta[r] \leq \dots \quad \text{and} \quad \mathbf{Fix} f \zeta = \bigvee \{\zeta[r] \mid r < \omega\}$$

(by unravelling the definition of  $\mathbf{Fix}$ ). This show that  $\mathbf{Fix}$  is a fixed point extractor.

7.11 LEMMA. For each fruitful  $f : \mathbb{Ord}'$  and  $\zeta \in \mathbb{Ord}$ , the value  $\mathbf{Fix} f \zeta$  is the least ordinal  $\nu$  such that  $\zeta < \nu = f\nu$ .

Fruitful and helpful functions work hand in hand.

7.12 LEMMA. (a) For each fruitful function  $f$  the function  $\mathbf{Fix} f$  is helpful.

(b) For each helpful function  $h$  and ordinal  $\zeta$ , the ordinal function  $\alpha \mapsto h^\alpha \zeta$  is normal.

**Proof.** (a) For fruitful  $f$  let  $h = \mathbf{Fix} f$ . For  $\zeta \in \mathbb{Ord}$  let  $\nu = h\zeta$ . By Lemma 7.11, we have  $\zeta < \nu = f\nu$  with a certain minimality on  $\nu$ . In particular,  $h$  is strictly inflationary.

Consider any  $\zeta \leq \eta$  and let  $\mu = h\eta$ . Then  $\zeta \leq \eta < \mu = f\mu$  and hence  $\nu \leq \mu$  by the minimality of  $\nu$ . This shows that  $h$  is monotone.

Since  $\nu \neq 0$ , we have  $\omega^\nu \leq f\nu = \nu$ , so that  $\nu$  is critical, and hence  $h$  is strictly big.

(b) By construction for each ordinal  $\alpha$  we have  $f(\alpha + 1) = h(f\alpha) > f\alpha$  since  $h$  is strictly inflationary. Also by construction we have  $f\lambda = \bigvee \{f\alpha \mid \alpha < \lambda\}$  for each limit ordinal  $\lambda$ . But if  $\alpha < \lambda$  then  $\alpha < \alpha + 1 < \lambda$  so that  $f\alpha < f(\alpha + 1) \leq f\lambda$  which is enough to show that  $f$  is strictly monotone.

A simple argument now shows that  $f$  is continuous.

For non-zero  $\alpha$  the value  $f\alpha$  is either a value of  $h$  or a supremum of such values. Thus  $f\alpha$  is critical. But  $\alpha \leq f\alpha$  and hence  $\omega^\alpha \leq \omega^{f\alpha} = f\alpha$  to show that  $f$  is big. ■

Exponentiation to base  $\omega$  is fruitful, and hence the term  $\mathbf{Next}$  names a helpful function.

7.13 DEFINITION. Let  $\mathbf{Next} = \mathbf{Fix} \omega^\bullet$ . ■

By Lemma 7.12(a) the function  $\mathbf{Next}$  is helpful. Let  $h$  be any helpful function, and let  $f = \omega^\bullet$ . For each  $\zeta \in \mathbb{Ord}$  we have  $\zeta + 1 \leq h\zeta$  and  $h\zeta$  is critical, so that  $f(\zeta + 1) \leq \omega^{h\zeta} = h\zeta$ . An easy induction gives  $f^r(\zeta + 1) \leq h\zeta$  for all  $r \leq \omega$ , and hence  $\mathbf{Next}\zeta \leq h\zeta$ .

7.14 LEMMA. Suppose  $h : \mathbb{Ord}'$  is helpful.

(a) For all ordinals  $\alpha, \beta, \zeta$  we have  $\zeta + \alpha \leq h^\alpha \zeta$ .

(b) For each (additively) critical ordinal  $\lambda$  we have  $h^\lambda \zeta = h^\lambda 0$  for each ordinal  $\zeta < \lambda$ .

**Proof.** (a) We prove this by induction on  $\alpha$ . The base case,  $\alpha = 0$ , is trivial. For the induction step,  $\alpha \mapsto \alpha + 1$ , since  $h$  is strictly inflationary we have

$$h^{\alpha+1}\zeta = h(h^\alpha\zeta) \geq h^\alpha\zeta + 1 \geq \zeta + \alpha + 1$$

using the induction hypothesis. For the induction leap to a limit ordinal  $\lambda$  we have

$$h^\lambda\zeta = \bigvee \{h^\alpha\zeta \mid \alpha < \lambda\} \geq \bigvee \{\zeta + \alpha \mid \alpha < \lambda\} = \zeta + \bigvee \{\alpha \mid \alpha < \lambda\} = \zeta + \lambda$$

as required.

(b) The iterate  $h^\lambda$  is helpful, and hence monotone, so that  $h^\lambda\zeta \geq h^\lambda 0$ . For the converse we have  $\zeta \leq h^\zeta 0$  (by part (a)) and hence  $h^\lambda\zeta \leq h^\lambda(h^\zeta 0) = h^{\zeta+\lambda} 0$  by Lemma 7.8. But  $\zeta < \lambda$  and  $\lambda$  is (additively) critical so that  $\zeta + \lambda = \lambda$ , to give the required result. ■

## Helpful higher level functions

We introduce the notion of a helpful function  $H : \text{Ord}^{(l+2)}$  on level  $l$ , and show that the interpretation  $[l] : \text{Ord}^{(l+2)}$  of the term  $[l]$  is helpful. These are fixed point extractors, and we use them to generate iteration templates with large ordinal values.

By decomposing the space

$$\text{Ord}^{(l+2)} = \text{Ord}^{(l+1)} \rightarrow \text{Ord}^{(l)} \rightarrow \dots \rightarrow \text{Ord}' \rightarrow \text{Ord} \rightarrow \text{Ord}$$

we see that each function  $H : \text{Ord}^{(l+2)}$  must receive successive arguments

$$h : \text{Ord}^{(l+1)}, h_l : \text{Ord}^{(l)}, \dots, h_1 : \text{Ord}', \zeta : \text{Ord}$$

to return its eventual ordinal value  $Hhh_l \dots h_1\zeta$ . Often these central arguments  $h_l, \dots, h_1$  play a passive role, so we write  $Hh\mathbf{h}\zeta$  for the eventual value. However, we need to take care with this abbreviation, for ‘ $\mathbf{h}$ ’ on its own would mean something different. We do not use this abbreviation in the following definition, but we will in the subsequent analysis.

**7.15 DEFINITION.** For each  $l < \omega$  a function  $H : \text{Ord}^{(l+2)}$  is **helpful** if

- (Help1)  $Hh$  is helpful
- (Help2)  $h^2h_l \dots h_1 \leq Hhh_l \dots h_1$
- (Help3)  $Hhh_l \dots h_2f \leq Hhh_l \dots h_2g$

for all helpful  $h : \text{Ord}^{(l+1)}$ ,  $h_l : \text{Ord}^{(l)}$ ,  $\dots$ ,  $h_1 : \text{Ord}'$ , and  $g : \text{Ord}'$ ,  $f : \text{Ord}'$  with  $f \leq g$ .

Let  $\mathbb{H}^{(l+2)}$  be the set of helpful inhabitants on  $\text{Ord}^{(l+2)}$ . ■

This is a definition by recursion on the level  $l$ . The comparison in (Help2, Help3) takes place in  $\text{Ord}'$ . Since the functions involved are in  $\mathbb{IM}$ , this doesn't lead to difficulties.

For the case  $l = 0$  you should read (Help3) with some care, because the sequence  $h, h_l, \dots, h_2$  is empty. A function  $H : \text{Ord}''$  is helpful precisely when

- (1)  $Hh$  is helpful
- (2)  $h^2 \leq Hh$
- (3)  $Hf \leq Hg$

hold for all helpful  $f, g, h : \text{Ord}'$  with  $f \leq g$ .

The squaring property (H2) is quite powerful, especially when used at higher levels.

7.16 LEMMA. If  $H : \text{Ord}^{(l+2)}$ ,  $h : \text{Ord}^{(l+1)}$ ,  $h_l : \text{Ord}^{(l+1)}$ ,  $\dots$ ,  $h_1 : \text{Ord}'$  are helpful, then  $(hh)^2 \leq Hhh$  (where  $\mathbf{h}$  abbreviates  $h_l \cdots h_1$ ).

**Proof.** We proceed by induction on the level  $l$ . For the base case,  $l = 0$ , the parameter sequence  $\mathbf{h}$  is empty, and the required comparison  $h^2 \leq Hh$  is just (Help2). For the induction step,  $l \mapsto l + 1$ , consider a helpful  $K : \text{Ord}^{(l+3)}$ , as well as the helpful  $H, h, \mathbf{h}$ . By (Help1) we know that  $Hh$  is helpful. Thus, using (Help2) for  $K$  and the induction hypothesis, we have  $KHh\mathbf{h} \geq H^2h\mathbf{h} = H(Hh)\mathbf{h} \geq (Hh\mathbf{h})^2$  as required. ■

In Lemma 7.9 we saw that the class  $\mathbb{H}^{(1)}$  is smooth. We now generalize this.

7.17 LEMMA. For each  $l < \omega$ , the class  $\mathbb{H}^{(l+1)}$  is smooth.

**Proof.** Lemma 7.9 gives the result for  $\mathbb{H}^{(1)}$ . We look at  $\mathbb{H}^{(l+2)}$  for arbitrary  $l < \omega$ .

We show first that  $\mathbb{H}^{(l+2)}$  is closed under composition. To this end consider any  $G, H \in \mathbb{H}^{(l+2)}$ . To show  $G \circ H \in \mathbb{H}^{(l+2)}$  we look at (Help1, Help2, Help3) in turn.

For each  $h \in \mathbb{H}^{(l+1)}$  we have  $Hh \in \mathbb{H}^{(l+1)}$  and hence  $G(Hh) \in \mathbb{H}^{(l+1)}$  to verify (Help1). To verify (Help2) consider any compatible family  $h, \mathbf{h}$  of helpful functions. Then

$$(G \circ H)h\mathbf{h} = G(Hh)\mathbf{h} \geq (Hh\mathbf{h})^2 \geq Hh\mathbf{h} \geq h^2\mathbf{h}$$

as required. Here the first comparison follows by Lemma 7.16, the second follows since  $Hh\mathbf{h}$  is inflationary, and the third uses the (Help2) for  $H$ .

To verify (Help3) observe that

$$(G \circ H)hh_l \cdots h_2g = G(Hh)h_l \cdots h_2g \quad (G \circ H)hh_l \cdots h_2f = G(Hh)h_l \cdots h_2f$$

so the known monotone property of  $G$  gives the required result. (Strictly speaking, this is the argument for  $l \neq 0$ . A slight variant is needed for  $l = 0$ .)

To show that  $\mathbb{H}^{(l+2)}$  is closed under pointwise suprema, consider a non-empty subset  $\mathcal{H}$  of  $\mathbb{H}^{(l+2)}$ . We show that  $\bigvee \mathcal{H}$  is helpful. We look at (Help1, Help2, Help3) in turn.

For each  $h \in \mathbb{H}^{(l+1)}$  we have

$$(\bigvee \mathcal{H})h = \bigvee \{Hh \mid H \in \mathcal{H}\}$$

so the known closure property of  $\mathbb{H}^{(l+1)}$  gives the required property of  $\mathbb{H}^{(l+2)}$ . (Strictly speaking, this is a proof by induction on  $l$ .) This verifies (Help1).

To verify (Help2) consider any compatible  $h, \mathbf{h}$  family of helpful functions. Since  $\mathcal{H}$  is non-empty we have  $(\bigvee \mathcal{H})h\mathbf{h} \geq Hh\mathbf{h} \geq h^2\mathbf{h}$  for any selected any member  $H$  of  $\mathcal{H}$ .

Property (Help3) follows in the same way. ■

Lemmas 7.12 and 7.14 gave us some useful properties of  $h \in \mathbb{H}^{(1)}$ . These lift.

7.18 LEMMA. Let  $H : \text{Ord}^{(l+2)}$ ,  $h : \text{Ord}^{(l+1)}$ ,  $h_l : \text{Ord}^{(l)}$ ,  $\dots$ ,  $h_1 : \text{Ord}'$  be helpful.

(a) For each ordinal  $\zeta \in \text{Ord}$  the function  $f : \text{Ord}'$  given by  $f\alpha = H^\alpha h\mathbf{h}\zeta$  (for  $\alpha \in \text{Ord}$ ) is normal.

(b) The comparison  $H^\alpha h\mathbf{h}\alpha \leq H^{\alpha+1} h\mathbf{h}0$  holds for each ordinal  $\alpha$ .

(c) The equality  $H^\lambda h\mathbf{h}\zeta = H^\lambda h\mathbf{h}0$  holds for each limit ordinal  $\lambda$  and ordinal  $\zeta < \lambda$ .

**Proof.** (a) The function  $h\mathbf{h}$  is helpful, hence strictly inflationary, so that Lemma 7.16 gives  $Hh\mathbf{h}\zeta \geq (h\mathbf{h})^2\zeta = h\mathbf{h}(h\mathbf{h}\zeta) > h\mathbf{h}\zeta$ . In particular

$$f(\alpha + 1) = H(H^\alpha h)\mathbf{h}\zeta > H^\alpha h\mathbf{h}\zeta = f\alpha$$

(using  $H^\alpha h$  in place of  $h$ ).

For each limit ordinal  $\lambda$  and ordinal  $\alpha < \lambda$ , we have  $\alpha + 1 < \lambda$  and so (by the definition of  $f\lambda$ ) we have  $f\alpha < f(\alpha + 1) \leq f\lambda$  using the previous observation.

This shows that  $f$  is strictly monotone.

By construction the function  $f$  is continuous.

Finally, for each  $\alpha$  the function  $H^\alpha h\mathbf{h}$  is helpful, and so takes only critical values. But  $\alpha \leq f\alpha$ , so that  $\omega^\alpha \leq \omega^{f\alpha} = f\alpha$  as required.

(b) Using Lemma 7.16 we have

$$H^{\alpha+1}h\mathbf{h}0 = H(H^\alpha h)\mathbf{h}0 \geq (H^\alpha h\mathbf{h})^2 0 = H^\alpha h\mathbf{h}(H^\alpha h\mathbf{h}0) \geq H^\alpha h\mathbf{h}\alpha$$

where last comparison holds since each helpful function is inflationary.

(c) The comparison  $H^\lambda h\mathbf{h}\zeta \geq H^\lambda h\mathbf{h}0$  is immediate.

For the converse consider any ordinal  $\alpha$  with  $\zeta < \alpha < \lambda$ . Then, using part (b)

$$H^\alpha h\mathbf{h}\zeta \leq H^\alpha h\mathbf{h}\alpha \leq H^{\alpha+1}h\mathbf{h}0 \leq H^\lambda h\mathbf{h}0$$

so that taking the supremum over all  $\alpha < \lambda$  now gives the required result. ■

We can now exhibit the interpretation of the term  $[l]$ .

**7.19 DEFINITION.** For each level  $l$  let  $[l] : \mathbb{Ord}^{(l+2)}$  be the function given by

$$[l]h\mathbf{h} = \mathbf{Fix} f \quad \text{where } f\alpha = h^\alpha \mathbf{h}0 \text{ (for } \alpha \in \mathbb{Ord}\text{)}$$

for each compatible family  $h, \mathbf{h}$  of (helpful) functions. ■

It is important to understand what these functions do, so let's take a look at  $[0]$ .

Consider any helpful function  $h : \mathbb{Ord}'$ . By Lemma 7.12(b) the function  $f : \mathbb{Ord}'$  where  $f\alpha = h^\alpha 0$  (for  $\alpha \in \mathbb{Ord}$ ) is normal. By Lemmas 7.11 and 7.14(b), we have

$$[0]h\zeta = (\text{the least } \nu \text{ with } \zeta < \nu = h^\nu 0) = (\text{the least } \nu \text{ with } 0 < \nu = h^\nu \zeta)$$

for each  $\zeta \in \mathbb{Ord}$ . By definition, this  $\nu$  is non-zero. Hence  $\nu = h^\nu 0$  is either a value of  $h$  (if  $\nu$  is a successor) or is a supremum of such values. Thus  $\nu$  is critical.

By construction,  $[0]h$  is strictly inflationary, and a simple argument shows that it is monotone. We have just seen that  $[0]h$  takes only critical values, and hence  $[0]h$  is helpful. In particular,  $[0]$  satisfies (Help1).

Using the  $\zeta$  insensitivity we have  $[0]h\zeta = \nu = h^\nu 0 = h^\nu \zeta > h^2 \zeta$  and hence  $h^2 \leq [0]h$ . This shows that  $[0]$  satisfies (Help2). A similar argument shows that  $[0]$  also satisfies (Help3). Thus  $[0] : \mathbb{Ord}''$  is helpful.

**7.20 THEOREM.** For each  $l < \omega$  the function  $[l] : \mathbb{Ord}^{(l+2)}$  is helpful.

**Proof.** We have just seen the proof for  $l = 0$ , so let's look at the non-zero case. Thus we show that  $[l+1] : \text{Ord}^{(l+3)}$  is helpful.

Consider a family  $H : \text{Ord}^{(l+2)}$ ,  $h : \text{Ord}(l+1)$ ,  $h_l : \text{Ord}^{(l)}$ ,  $\dots$ ,  $h_1 : \text{Ord}'$  of helpful functions. By Lemma 7.18(a) the function  $f : \text{Ord}'$  given by  $f\alpha = H^\alpha h \mathbf{h} 0$  (for  $\alpha \in \text{Ord}$ ) is normal. By Lemma 7.12(a), the function  $[l+1]H h \mathbf{h} = \mathbf{Fix} f$  is helpful. Since  $h_1, \dots, h_l, h$  are arbitrary, this shows that  $[l+1]H$  is helpful, and hence  $[l+1]$  satisfies (Help1).

The value  $[l+1]H h \mathbf{h} \zeta$  is the least ordinal  $\nu$  such that  $\zeta < \nu = H^\nu h \mathbf{h} 0$  for each  $\zeta \in \text{Ord}$ . By Lemma 7.17 the function  $H^\nu h \mathbf{h}$  is helpful, and so takes only critical values. In particular,  $\nu$  is critical, and hence using Lemma 7.18(c) we have

$$[l+1]H h \mathbf{h} \zeta = H^\nu h \mathbf{h} 0 = H^\nu h \mathbf{h} \zeta > H^2 h \mathbf{h} \zeta$$

to show that  $[l+1]$  satisfies (Help2).

Finally, to verify (Help3) consider helpful  $f, g : \text{Ord}'$  with  $f \leq g$ . (Of course, this ' $f$ ' is not the same as before.) For  $\zeta \in \text{Ord}$  let

$$\mu = [l+1]H h h_l \cdots h_2 f \zeta \quad \nu = [l+1]H h h_l \cdots h_2 g \zeta$$

so we require  $\mu \leq \nu$ . But

$$\mu = H^\mu h h_l \cdots h_2 f \zeta \quad \nu = H^\nu h h_l \cdots h_2 g \zeta$$

with a certain minimality. Also, using Lemma 7.18(a), we have

$$\zeta < \nu \leq H^\nu h h_l \cdots h_2 f 0 \leq H^\mu h h_l \cdots h_2 g 0 = \nu$$

so that  $\mu \leq \nu$ , as required. ■

## 8 The Eastwood hierarchy

Which ordinals does the sequence  $\Delta[\cdot]$  of terms name? We can now answer that. Let

$$\Delta[0] = \omega \quad \Delta[1] = \mathbf{Next}\omega \quad \Delta[l+2] = [l] \cdots [0] \mathbf{Next}\omega$$

for each  $l < \omega$ . We want to put a value on each of these ordinals. We do that by comparing these descriptions with other systems of ordinal notations.

Except for one small interlude it is convenient to take an historical perspective.

God gave us the natural numbers, but forgot to tell us the limit point

$$\Delta[0] = \omega$$

the zeroth ordinal in our sequence. It was left to Cantor to discover  $\omega$  and peer beyond.

It is fair to say that the Cantor normal form gives the first system of ordinal notations. This is based on the exponentiation function  $\omega^\bullet$  to base  $\omega$ , and is good enough to name all the ordinals below  $\epsilon_0$ , the least ordinal  $\epsilon$  with  $\omega^\epsilon = \epsilon$ . Thus this system closes off at

$$\Delta[1] = \mathbf{Next}\omega = \epsilon_0$$

the first ordinal in our sequence.

To go beyond  $\epsilon_0$  we must name larger critical ordinals. We can do this since

$$\epsilon_\alpha = \mathbf{Next}^{1+\alpha}\omega$$

and we can use these in an extended Cantor normal form. This extended system closes off at the least ordinal  $\nu$  with  $\nu = \mathbf{Next}^\nu\omega$ , which is

$$\Delta[2] = [0]\mathbf{Next}\omega = \epsilon_{\epsilon_{\epsilon_{\dots}}}$$

the second ordinal in our sequence. This ordinal rarely gets a mention.

The next breakthrough was made by Veblen in [22]. Let  $\mathbf{Veb} : \text{Ord}''$  be given by

$$\mathbf{Veb}f\zeta = h^{1+\zeta}0 \quad \text{where } h = \mathbf{Fix}f$$

for  $f : \text{Ord}'$  and  $\zeta \in \text{Ord}$ . Thus  $\mathbf{Veb}f$  enumerates the fixed points of a normal (or fruitful) function  $f$ . The Veblen hierarchy  $\phi_f$  on  $f$  is obtained by iterating  $\mathbf{Veb}$ . For historical reasons the limit levels are usually omitted, and we have

$$\phi_f(1 + \alpha, \zeta) = \mathbf{Veb}^{\alpha+1}f\zeta$$

for each  $\alpha, \zeta \in \text{Ord}$ . Using the base function  $f = \omega^\bullet$  this system of notations closes off at  $\Gamma_0$ , the first strongly critical ordinal.

It is not hard to see that

$$(\mathbf{Fix} \circ \mathbf{Veb})f = ([0] \circ \mathbf{Fix})f$$

for each fruitful  $f$  and then

$$(\mathbf{Fix} \circ \mathbf{Veb}^\alpha)f = [0]^\alpha h \quad \text{where } h = \mathbf{Fix}f$$

for each  $\alpha \in \text{Ord}$ . Thus

$$\phi_f(1 + \alpha, \zeta) = \mathbf{Veb}^{\alpha+1}f\zeta = (\mathbf{Veb} \circ \mathbf{Veb}^\alpha)f\zeta = ((\mathbf{Fix} \circ \mathbf{Veb}^\alpha)f)^{1+\zeta}0 = ([0]^\alpha h)^{1+\zeta}0$$

which, for us, is a more convenient description of  $\phi_f$ . (This calculation is a simple example of the shuffle technique describe in [21].)

For  $f = \omega^\bullet$  this Veblen system closes of at the least  $\nu$  with  $\nu = [0]^\nu \mathbf{Next}\omega$ , which is

$$\Delta[3] = [1][0]\mathbf{Next}\omega = \Gamma_0$$

the third ordinal in our sequence.

How can we get beyond  $\Gamma_0$ ? With hindsight we may use the battery of critical ordinals

$$([1]^\alpha [0]^\beta \mathbf{Next})^{1+\gamma}\omega$$

for previously generated ordinals  $\alpha, \beta, \gamma$ . In particular, it can be checked that

$$\Gamma_\alpha = ([1][0]\mathbf{Next})^{1+\alpha}\omega$$

generates the sequence of strongly critical ordinals. This system closes off at the least solution of  $\nu = [1]^\nu [0] \mathbf{Next}\omega$ , which is

$$\Delta[4] = [2][1][0]\mathbf{Next}\omega$$

the fourth ordinal in our sequence. After that it is obvious what to do, we use more and more of the higher order fixed point extractors  $[l]$ .

Historically that is not what happened. To get beyond the closure ordinal of  $\phi_f$  (for  $f = \omega^\bullet$ ) Veblen used the hierarchy to produce a faster normal function  $f^+$ , and then repeated the construction with  $f^+$  as the base. He set

$$f^+\zeta = \phi_f(1 + \zeta, 0) = \mathbf{Veb}^{\zeta+1}f0 = [0]^\zeta h0$$

which matches what we did above. Veblen iterated this process and described an intricate system of indexing the whole family of hierarchies. This is not easy to understand.

In [16] Schütte reorganized and extended Veblen's method to produce some quite large ordinals. A Schütte bracket is an array of ordinals, as on the left, where  $r$  is finite.

$$\begin{pmatrix} \zeta & 1+\alpha(1) & \cdots & 1+\alpha(s) \\ r & 1+i(1) & \cdots & 1+i(s) \end{pmatrix} \quad f \begin{pmatrix} \zeta & 1+\alpha(1) & \cdots & 1+\alpha(s) \\ r & 1+i(1) & \cdots & 1+i(s) \end{pmatrix}$$

Each bracket converts a normal function  $f$  into an ordinal, as on the right, the value of the bracket at  $f$ . This is generated by an intricate recursion over the exponents  $\alpha$  and indexes  $i$  where the size  $s$  of the array is allowed to vary. In fact, there is an explicit definition of these brackets. For ordinals  $\alpha, i$  let  $\mathbf{S} \left[ \begin{smallmatrix} \alpha+1 \\ i+1 \end{smallmatrix} \right] : \text{Ord}''$  be the operator given by

$$\mathbf{S} \left[ \begin{smallmatrix} \alpha+1 \\ i+1 \end{smallmatrix} \right] f\zeta = \left( ([1]^i [0])^{1+\alpha} h \right)^{1+\zeta} 0 \quad \text{where } h = \mathbf{Fix} f$$

for  $f : \text{Ord}'$  and  $\zeta \in \text{Ord}$ . For each such  $f$  let  $f_\bullet$  be the sequence generated by

$$f_0 = f \quad f_{1+r}(1 + \eta) = \mathbf{S} \left[ \begin{smallmatrix} \eta \\ r+1 \end{smallmatrix} \right] f_0$$

for each  $r < \omega$  and  $\eta \in \text{Ord}$ . (We may set  $f_{1+r}0 = 1$ .) It can be shown that

$$f \begin{pmatrix} \zeta & 1+\alpha(1) & \cdots & 1+\alpha(s) \\ r & 1+i(1) & \cdots & 1+i(s) \end{pmatrix} = \left( \left( \mathbf{S} \left[ \begin{smallmatrix} \alpha(1)+1 \\ i(1)+1 \end{smallmatrix} \right] \circ \cdots \circ \mathbf{S} \left[ \begin{smallmatrix} \alpha(s)+1 \\ i(s)+1 \end{smallmatrix} \right] \right) f \right)_r \zeta$$

for each normal function  $f$  and array of ordinals. The details are set out in [21].

In other words, the Schütte brackets provide a selection of the ordinals that can be named using the two helpful function  $[0]$  and  $[1]$ . In particular

$$\Delta[4] = [2][1][0] \mathbf{Next}\omega$$

is the limit of this technique. This is sometime called the Ackermann ordinal.

Slightly earlier and almost unknown at the time, Bachmann in [3] described a method capable of naming very much larger ordinals. He used uncountable ordinals to name countable ordinals. This is now the standard method of generating ordinal notations. A description of this method can be found in [4], and [14] and [15] contain useful material.

Let  $\Omega = \omega_1$  be the first uncountable ordinal, and let  $\Omega^+ = \epsilon_{\Omega+1}$  be the next critical ordinal beyond  $\Omega$ . We use a certain function

$$\psi : [0, \Omega) \longrightarrow [0, \Omega)$$

which enumerates the critical ordinals. Of course, such a function must be constant for long stretches. The precise details of  $\psi$  are not needed here. Using an iteration of the exponentiation function to base  $\Omega$ , for each  $l < \omega$  let

$$\nabla[l] = \psi((\Omega^\bullet)^l 0)$$

to obtain the standard fundamental sequence  $\nabla[\cdot]$  for the Howard ordinal. For instance

$$\nabla[0] = \psi 0 \quad \nabla[1] = \psi 1 \quad \nabla[2] = \psi \Omega \quad \nabla[3] = \psi(\Omega^\Omega)$$

and so on. With a bit of effort it can be shown that  $\nabla[\cdot]$  and  $\Delta[\cdot]$  are the same sequence.

The Bachmann method can be immersed into  $\lambda\mathbf{H}$ . We attach to  $\xi < \Omega^+$  a battery

$$\{\xi\} : \text{Ord}' \quad \{\xi\}_l : \text{Ord}^{(l+1)}$$

of associated functions (or a battery of terms of  $\lambda\mathbf{H}$ ). We set

$$\{0\} = id \quad \{0\}_0 = \mathbf{Next} \quad \{0\}_{l+1} = [l]$$

and for each non-zero  $\xi = \Sigma + \Omega^\Delta \cdot \alpha$  (where this sum meshes correctly) we set

$$\{\xi\} = \{\Delta\}_0^\alpha \circ \{\Sigma\} \quad \{\xi\}_l = \{\Delta\}_{l+1}^\alpha \{\Sigma\}_l$$

using gadgets assigned earlier. Each non-zero ordinal  $\xi < \Omega^+$  has a decomposition

$$\xi = \Omega^{\xi(s)} \cdot \alpha(s) + \dots + \Omega^{\xi(0)} \cdot \alpha(0)$$

where  $\xi > \xi(s) > \dots > \xi(0)$  and each  $\alpha(i) < \Omega$ . It can be checked that

$$\{\xi\} = \{\xi(0)\}_0^{\alpha(0)} \circ \dots \circ \{\xi(s)\}_0^{\alpha(s)} \quad \{\xi\}_l = \left( \{\xi(0)\}_{l+1}^{\alpha(0)} \circ \dots \circ \{\xi(s)\}_{l+1}^{\alpha(s)} \right) \{0\}_l$$

hold. In particular, there is a close matching between the full decomposition of  $\xi$  and the construction of the associated gadgets.

The enumerating function has many long constant stretches. These are the wild stretches. A ordinal  $\xi < \Omega^+$  is tame if  $\psi(\xi + 1) = \mathbf{Next}(\psi\xi)$ . It can be shown that if  $\xi$  is tame then  $\psi\xi = \{\xi\}_{\epsilon_0}$ . The details of this are set out in [18].

## References

- [1] P. Aczel: Describing ordinals using functionals of transfinite type, *J. Symbolic Logic*, 37 (19972) 35 – 47.
- [2] J. Avigad and S. Feferman: Gödel’s Functional (“Dialectica”) Interpretation, pp 337 – 405 of [5].
- [3] H. Bachmann: Die Normalfunktionen und das Problem der ausgezeichneten Folgen von Ordnungszahlen, *Vierteljahresschr. Naturforsch. Ges. Zürich* 95 (1950) 115 – 147.
- [4] W. Buchholtz and K. Schütte: *Proof Theory of Impredicative Subsystems of Analysis*, Bibliopolis, Naples, 1998.
- [5] S. R. Buss: *Handbook of Proof Theory* North Holland, 1998.
- [6] N. Danner: Ordinal notations in typed  $\lambda$ -calculi, Ph. D. thesis, Indiana, 1999.
- [7] N. Danner: Ordinals and ordinals functions representable in the simply typed  $\lambda$ -calculus, *Annals of P. and A. Logic* 97 (1999) 179 – 201.

- [8] M. Fairtlough and S. S. Wainer: Hierarchies of provably recursive functions, pp. 149–207 of [5].
- [9] S. Feferman: Hereditarily replete functionals over the ordinals, pp 289 – 301 of [13].
- [10] J-Y. Girard: *Proof Theory and Logical Complexity*, vol. 1, Bibliopolis, 1987.
- [11] J-Y. Girard, Y. Lafont and P. Taylor: *Proofs and Types*, Cambridge University Press, 1996.
- [12] W. A. Howard: A system of abstract constructive ordinals *J. Symbolic Logic* 37 (1972) 355 – 374.
- [13] A. Kino, J. Myhill, and R. E. Vesley: *Intuitionism and Proof Theory (Buffalo. New York, 1968)*, North-Holland, Amsterdam, 1968.
- [14] W. Pohlers: *Proof Theory*, Springer L.N.M 1407, Springer-Verlag, Berlin, 1989.
- [15] W. Pohlers: Proof theory and ordinal analysis, *Arch. Math. Logic* 30 (1991) 311 – 376.
- [16] K. Schütte: Kennzeichnung von Ordnungszahlen durch rekursive erklärte Functionen, *Meth. Ann.* 127 (1954) 15 – 32.
- [17] J. R. Shoenfield: *Mathematical Logic*, Addison-Wesley, 1967.
- [18] H. Simmons: A comparison of two systems of ordinal notations, *Archive for Math. Logic* 34 (2004) 65-83.
- [19] H. Simmons: Fruitful and helpful ordinal functions *Archive for Math. Logic* 47 (2008) 677-709.
- [20] H. Simmons: The Ackermann functions are not optimal, but by how much? *J. Symbolic Logic to appear*
- [21] H. Simmons: Generating ordinal notations from below with a non-recursive construction of the Schütte brackets.  
*Unpublished but available in this part of my web-page*
- [22] O. Veblen: Continuous increasing functions of finite and transfinite ordinals, *Trans. Amer. Math. Soc.* 9 (1908) 280 – 292.