

The Ackermann functions are not optimal, but by how much?

H. Simmons

School of Mathematics, The University, Manchester, England

hsimmons@manchester.ac.uk

Abstract

By taking a closer look at the construction of an Ackermann function we see that between any primitive recursive degree and its Ackermann modification there is a dense chain of primitive recursive degrees.

Key words: Primitive recursive degree, Ackermann jump

200 AMS Classification: To be done

Preamble

Along with a bit of history, in these introductory remarks I will try to give an informal idea of the aims of this paper. I may use some terminology which you, the reader, may not be familiar with. Don't worry. In the later sections, when we get down to the technical details, I will be more precise.

In 1928 Ackermann [1] produced a function (on the natural numbers) that is not primitive recursive but is clearly recursive in some sense. We may view the function as 2-placed together with a parameter a . The lower levels are

$$\mathfrak{A}_a(0, x) = a + x$$

$$\mathfrak{A}_a(1, x) = a \times x$$

$$\mathfrak{A}_a(2, x) = a^x$$

and these are continued by

$$\mathfrak{A}(i', 0) = \text{some suitable value} \quad \mathfrak{A}_a(i', x') = \mathfrak{A}_a(i, z) \quad \text{where } z = \mathfrak{A}_a(i', x)$$

for $i, x \in \mathbb{N}$. Here $(\cdot)'$ is the successor function. Provided $a \geq 2$ this function is not primitive recursive because of its rate of growth. For every primitive recursive function there is some level i such that $\mathfrak{A}_a(i, \cdot)$ eventually dominates that function.

R. Péter [4] cleaned up the recursive specification. This was then simplified by R.M. Robinson [6]. They also noticed that the parameter a isn't doing much, and removed it. In fact, we can replace the parameter by any 1-placed function f , as we do here.

(Base)	$A(0, 0, x) = fx$	$R(0, x) = fx$	$G(0, x) = fx$
(Outer)	$A(i', 0, x) = A(i, x, x)$	$R(i', 0) = R(i, 1)$	$G(i', 0) = 2$
(Inner)	$A(i, r', x) = A(i, 0, z)$ where $z = A(i, r, x)$	$R(i', x') = R(i, z)$ where $z = R(i', x)$	$G(i', x') = G(i, z)$ where $z = G(i', x)$

The function R is due to Péter and Robinson (for the case where f is the successor function). The function G is essentially due to Grzegorzcyk. I am not sure where the function A comes from, although it is related to the fast growing hierarchy.

Provided f is strictly inflationary and monotone, each of these functions is not primitive recursive in f , and all have the same primitive recursive degree of complexity (in fact, Kalmár elementary complexity).

Are these examples optimal in any sense? They are not. I will show that between the primitive recursive degree of f and that of the functions there is a dense set of primitive recursive degrees.

Most of the published proof of the Ackermann result provide very little insight. They seem to be nothing more than a bunch of tricks. We read the proof, see that it does work, but we don't see why. Part of my aim here is to try to provide some structure in which to place this and similar proofs. Of course, with this kind of topic there will always be tricks used, but there is no harm in looking for underlying themes.

Let's rework the description of the three functions A, R, G .

A jump function is a higher order function which converts a 1-place function

$$h : \mathbb{N} \longrightarrow \mathbb{N}$$

into another 1-placed function. Here we are interested in three such jump functions

$$\mathbf{Ack} \qquad \mathbf{Rob} \qquad \mathbf{Grz}$$

given by

$$\mathbf{Ack} \ h x = h^{x+1} x \qquad \mathbf{Rob} \ h x = h^{x+1} 1 \qquad \mathbf{Grz} \ h x = h^x 2$$

for each 1-placed function h and each $x \in \mathbb{N}$. We then find that

$$A(i, r, x) = (\mathbf{Ack}^i f)^{1+r} x \qquad R(i, x) = \mathbf{Rob}^i f x \qquad G(i, x) = \mathbf{Grz}^i f x$$

for each $i, r, x \in \mathbb{N}$, and we immediately catch a glimmer of some structure. The Ackermann result is telling us something about the power of these three jump functions. This is partly brought out by relativizing the result to a suitable base function f .

The primitive recursive degrees measure the complexity of functions up to primitive recursive equivalence. Two functions have the same primitive recursive degree if each is primitive recursive in the other. A use of one of the three jump functions lifts each primitive recursive degree \mathbf{d}_0 to a strictly higher one \mathbf{d}_1 (and this is independent of which jump we use). This suggests an obvious question. Are there any primitive recursive degrees strictly between \mathbf{d}_0 and \mathbf{d}_1 ?

Let \mathbb{I} be the set of reals $0 < p < 1$ and let \mathbb{J} be the set of rationals in this interval. I will show there is a \mathbb{J} -indexed family

$$(\mathbf{d}_p \mid p \in \mathbb{J})$$

of primitive recursive degrees with

$$\mathbf{d}_0 < \mathbf{d}_p < \mathbf{d}_1 \qquad p < q \implies \mathbf{d}_p < \mathbf{d}_q$$

for all rationals $p, q \in \mathbb{J}$.

The complexity of a function can be measured using two components; its rate of growth and the complexity of its graph. For most of the functions we deal with here the graph has very low complexity.

The analysis splits into two parts.

In Section 1 and 2 we look at rates of growth. For each suitable function f (strictly inflationary and monotone) there is an easy way of increasing its rate of growth. We hit it with one of the jump functions. We may iterate that process through \mathbb{N} and eventually produce a function f_ω with a very much greater rate of growth. After some preparation we exhibit in Theorem 2.5 a family of functions F_p , indexed by the reals, which sit between f and f_ω , and as p increases the leap in rate of growth is considerable.

In Section 3 and 4 we turn our attention to the complexity of relations, in particular the complexity of graphs of functions. We need to show that the complexity of the graph of each of the functions A, R, G is no greater than that of the graph of the parent function f . Section 4 describes a way of doing this. The result is far from new, but the method used may be of independent interest.

Finally, in Section 5, we look at the complexity of functions, and exhibit a dense set of primitive recursive degrees between the base function f and its modification f_ω .

In this paper I deal only with the jump function **Ack** and its associated function A . Similar methods can be used to deal with **Rob** and **Grz**, but there is little point in producing the details of minor variations.

Much of this material was first worked out in [8]. As there I am grateful to Adam Cichon, Matt Fairtlough, and Stan Wainer who first got me interested in this topic. Alex Wilkie showed me the trick, Lemma 1.8, on which the whole proof depends.

There doesn't seem to be much literature on this topic (as opposed to hierarchies beyond the level we look at). There is a wealth of information in [5], but reading that book can be hard going. The three books [7, 9, 10] give more modern accounts, each with its own slant. I must also mention the two volume work [3]. A very useful source of all kinds of information on this and related topics. Finally, you may find it worthwhile to locate a copy of [2].

Contents

1	Snakes and ladders	3
2	The interpolation technique	8
3	Complexity of relations	12
4	The derivation technique	15
5	Dense chains of p.r. degrees	21
	References	24

1 Snakes and ladders

In this and the next section we consider the rate of growth of a function and how that notion can be stratified.

1.1 DEFINITION. A snake is a 1-placed function

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$

which is monotone, that is

$$x \leq y \implies fx \leq fy$$

for $x, y \in \mathbb{N}$.

A snake is, respectively,

slow fast

if

$$fx \leq x \quad x < fx$$

for all $x \in \mathbb{N}$. ■

We mostly deal with fast snakes, but the crucial Lemma 2.3 makes use of a slow snake. The identity function **id** on \mathbb{N} is a slow snake, and the successor function is a fast snake.

Each n -placed function ϕ on \mathbb{N} can be converted into its bounding snake $\bar{\phi}$. We set

$$\bar{\phi}x = \max\{\phi(x_1, \dots, x_n) \mid x_1, \dots, x_n \leq x\}$$

for each $x \in \mathbb{N}$. This snake $\bar{\phi}$ can be used to determine the rate of growth of ϕ in the sense described below. This snake $\bar{\phi}$ need not be fast. If we want a fast snake $\bar{\bar{\phi}}$ then the following will do.

$$\bar{\bar{\phi}}x = \max\{\bar{\phi}x, x + 1\}$$

Since snakes are monotone the obvious way to compare them is pointwise, that is

$$f \leq g \iff (\forall x)[fx \leq gx]$$

for snakes f and g . For our purposes this comparison isn't good enough. We don't care what a snake does for the first few inputs; it's what it does eventually that matters.

1.2 DEFINITION. For snakes f and g we let

$$f \sqsubseteq_a g \quad \text{mean} \quad (\forall x)[a \leq x \implies fx \leq gx]$$

for each $a \in \mathbb{N}$. We let

$$f \sqsubseteq g \quad \text{mean} \quad (\exists a)[f \sqsubseteq_a g]$$

to obtain the comparison of **eventual domination**. ■

Note that \sqsubseteq_0 is just the pointwise comparison.

The composite of two snakes is a snake. In particular, the finite iterates

$$f^0 = \mathbf{id}, f^1 = f, f^2, f^3, \dots, f^{r+1} = f \circ f^r, \dots$$

of a snake are all snakes.

1.3 LEMMA. For fast snakes f, g, h we have

$$\left. \begin{array}{l} r \leq s \\ x \leq y \end{array} \right\} \implies f^r x \leq f^s y \quad f \sqsubseteq_a g \implies f^{r+1} \sqsubseteq_a g^{r+1}$$

and

$$f \sqsubseteq_a g \implies h \circ f \sqsubseteq_a h \circ g \quad \text{and} \quad f \circ h \sqsubseteq_a g \circ h$$

for all $a, r, s, x, y \in \mathbb{N}$.

For a fast snake f we certainly have

$$f^0 \sqsubseteq_0 f^1 \sqsubseteq_0 f^2 \sqsubseteq_0 \cdots \sqsubseteq_0 f^r \sqsubseteq_0 \cdots$$

but we want to get beyond that. This is where the jump $\mathbf{Ack}f$ is useful.

1.4 LEMMA. *For each fast snake f the jump $\mathbf{Ack}f$ is also a snake with $f \sqsubseteq_0 \mathbf{Ack}f$. Furthermore, we have*

$$f \sqsubseteq_a g \implies \mathbf{Ack}f \sqsubseteq_a \mathbf{Ack}g$$

for fast snakes f, g , and $a \in \mathbb{N}$.

Proof. We have

$$\mathbf{Ack}f0 = f0$$

and the strictly inflationary property of f gives

$$fx < f^2x < f^3x < \cdots < f^{x+1}x = \mathbf{Ack}fx$$

to deal with $x > 0$.

We are given

$$fx \leq gx$$

for all $x \geq a$. The snake properties of f and g give

$$f^r x \leq g^r x$$

for all r , in particular we have

$$\mathbf{Ack}fx = f^{x+1}x \leq g^{x+1}x = \mathbf{Ack}g$$

as required. ■

Observe that for each fast snake f we have

$$f^r \sqsubseteq_r \mathbf{Ack}f$$

for each $r \in \mathbb{N}$. Thus, in terms of rate of growth, $\mathbf{Ack}f$ jumps above each of the iterates of f (hence the terminology). In particular, we can not have

$$\mathbf{Ack}f \sqsubseteq f$$

for any fast snake f . To see that suppose

$$\mathbf{Ack}f \sqsubseteq_a f$$

for some a . Then, with $x = a + 1$ we have

$$fx < f^2x \leq f^{x+1}x \leq fx$$

which is impossible.

The obvious thing to do now is iterate this jump.

1.5 DEFINITION. For each fast snake f the Ackermann ladder

$$(f_i \mid i < \omega)$$

is generated by

$$f_0 = f \quad f_{i+1} = \mathbf{Ack} f_i$$

that is

$$f_i = \mathbf{Ack}^i f$$

for each $i < \omega$. We also set

$$f_\omega x = f_x x$$

(for $x \in \mathbb{N}$) to obtain the diagonal f_ω of the ladder. ■

For a fast snake f we have

$$f = f_0 \sqsubseteq_0 f_1 \sqsubseteq_0 f_2 \sqsubseteq_0 \cdots \sqsubseteq_0 f_i \sqsubseteq_0 \cdots \quad (i < \omega)$$

and a few simple calculations shows that f_ω is a fast snake with

$$f_i \sqsubseteq_i f_\omega$$

for each $i < \omega$.

At this point it is worth relating this ladder to the 3-placed Ackermann function A based on f . The following is known, but I want to make a comment about the proof.

1.6 THEOREM. For each fast snake f we have

$$A(i, r, x) = (\mathbf{Ack}^i f)^{r+1} x = f_i^{r+1} x$$

for all $i, r, x \in \mathbb{N}$.

Proof. In fact, this result holds for every 1-placed function f (but then we don't have the ladder comparisons).

For $i, r \in \mathbb{N}$ let

$$[i, r] \text{ abbreviate } (\forall x)[A(i, r, x) = (\mathbf{Ack}^i f)^{r+1} x]$$

$$[i] \text{ abbreviate } (\forall r)[i, r]$$

so that

$$(\forall i, r)[i, r]$$

is our objective. We first show that

$$(B) \quad [0, 0]$$

$$(O) \quad [i] \implies [i', 0]$$

$$(I) \quad [i, 0] \wedge [i, r] \implies [i, r']$$

for each $i, r \in \mathbb{N}$. With these a routine double induction gives the required result. ■

Later we will use this format for other constructions and proofs.

For each fast snake f we have its chain of iterates followed by its ladder of jumps culminating in the snake f_ω . All the steps on the way to f_ω are primitive recursive in f , but f_ω is not. It grows too fast. We return to this in Section 5.

We conclude this section with two technical results which we use in the next section.

1.7 LEMMA. *Suppose, for fast snakes g and h we have*

$$(\forall x)[x < \mathbf{Ack}ha \implies gx \leq hx]$$

for some $a \in \mathbb{N}$. Then

$$(\forall x)[x \leq a \implies \mathbf{Ack}gx \leq \mathbf{Ack}hx]$$

also holds.

Proof. Let

$$[r] \text{ abbreviate } (\forall x)[x \leq a \implies g^{r+1}(x) \leq h^{r+1}(x)]$$

for $r \in \mathbb{N}$. We verify $[r]$ for $r \leq a$. We proceed by induction on r .

The base case $[0]$ holds since $a < \mathbf{Ack}ha$ and hence (using the given hypothesis)

$$x \leq a \implies x < \mathbf{Ack}ha \implies gx \leq hx$$

for the required result.

For the induction step $r \mapsto r+1$, suppose that $r+1 \leq a$ and $x \leq a$, and set $y = h^{r+1}x$. Then

$$y < h^{r+2}x \leq h^{a+1}x \leq h^{a+1}a = \mathbf{Ack}ha$$

so that

$$gy \leq hy = h^{r+2}x$$

(by the given hypothesis). Also the Induction Hypothesis $[r]$ gives

$$g^{r+1}x \leq h^{r+1}x = y$$

so that $g^{r+2}x \leq gy$, from which $[r+1]$ follows.

Finally, to prove the desired result, consider any $x \leq a$ and let $r = x$. Then $[r]$ gives

$$\mathbf{Ack}gx = g^{r+1}x \leq h^{r+1}x = \mathbf{Ack}hx$$

which is the required comparison. ■

So far we have been concerned mostly with fast snakes. But earlier I said that one crucial result makes use of a slow snake. Here is how we produce that slow snake.

1.8 LEMMA. *For each fast snake h there is a slow snake k which does increase without bound, and such that*

$$x < ha \implies kx \leq 1 + ka$$

for each $a, x \in \mathbb{N}$.

Proof. Since h is a fast snake we have

$$0 < h0 < h^20 < \dots < h^r0 < \dots$$

so we use these intervals to produce k . For each $x \in \mathbb{N}$ let kx be the unique r with

$$h^r0 \leq x < h^{r+1}0$$

so that k is constant on this interval. We have

$$kx \leq r \iff x < h^{r+1}0$$

for each $x, r \in \mathbb{N}$.

Since $r \leq h^r 0$, we have $kx \leq x$. For $x \leq y$ with $s = ky$, we have

$$x \leq y < h^{s+1}0$$

so that $kx \leq s = ky$. Thus k is a slow snake.

For each $r \in \mathbb{N}$, with $x = h^r 0$ we have $kx = r$, so that k does increase without bound.

Finally, consider any $a, x \in \mathbb{N}$ with $x < ha$. Let $s = ka$, so that $a < h^{s+1}0$, to give

$$x < ha \leq h^{s+2}0$$

and hence $kx \leq s + 1 = 1 + ka$. ■

We apply this construction to $h = f_\omega$ where f is an arbitrary fast snake.

2 The interpolation technique

In this section I describe the technique from which we will obtain our main result.

Throughout the section we fix a fast snake f with its ladder

$$f = f_0 \sqsubseteq f_1 \sqsubseteq \cdots \sqsubseteq f_i \sqsubseteq \cdots \sqsubseteq f_\omega$$

of ever faster snakes. The aim is to insert below f_ω but above each f_i a long chain of snakes of increasing speed.

To motivate the idea let's recall how the ladder can be extended *above* f_ω .

We iterate the jump operator **Ack** through the ordinals. Thus we set

$$\mathbf{Ack}^0 f = f \quad \mathbf{Ack}^{\alpha+1} f = \mathbf{Ack}(\mathbf{Ack}^\alpha f) \quad \mathbf{Ack}^\lambda f x = \mathbf{Ack}^{\lambda[x]} f x$$

for each ordinal α , limit ordinal λ , and $x \in \mathbb{N}$. For each limit ordinal λ we require a fundamental sequence $\lambda[\cdot]$ for that ordinal. This, of course, puts a restriction on how far we can go. More importantly, the complexity of a function in the long chain is highly dependent on the system of ordinal notations we employ. We use the same idea here, but only for the limit ordinal ω .

Notice that

$$f_\omega = \mathbf{Ack}^\omega f$$

where $\omega[\cdot]$ is the identity function. A more standard choice of fundamental sequence is the successor function. (It generates better nesting properties of the fundamental sequences at higher limit ordinals.) With this version of $\omega[\cdot]$ we have

$$\mathbf{Ack}^\omega f x = \mathbf{Ack}^{x+1} f x = \mathbf{Ack}(\mathbf{Ack}^x f)x = \mathbf{Ack} f_\omega x$$

to exhibit a small jump in complexity.

We will use a whole family of rather tardy fundamental sequences for ω .

By Lemma 1.8 the fast snake $h = f_\omega$ gives us a slow snake k . We fix this function k for the remainder of this section.

Recall that \mathbb{I} is the set of reals p with $0 < p < 1$.

2.1 DEFINITION. For each real $p \in \mathbb{I}$ let

$$\omega_p[x] = \lfloor p(kx) \rfloor$$

for each $x \in \mathbb{N}$. ■

For some purposes this function $\omega_p[\cdot]$ would not be considered suitable to be a fundamental sequence for ω . It satisfies

$$\omega_p[x] < kx \leq x$$

for each $x \in \mathbb{N}$ and can be constant for long periods. However, it does eventually increase without bound. It doesn't go haring off like some fundamental sequences, and we will make good use of its tortoise-like qualities.

2.2 DEFINITION. For each real $p \in \mathbb{I}$ let

$$F_p = \mathbf{Ack}^{\omega_p} f$$

to obtain a fast snake. ■

In more detail we have

$$F_p x = \mathbf{Ack}^{\omega_p[x]} f x$$

for each $x \in \mathbb{N}$. Let $m = \omega_p[x]$. Then Lemma 1.4 gives

$$x < f x < \mathbf{Ack} f x < \cdots < \mathbf{Ack}^m f x = F_p x$$

to show that F_p is strictly inflationary. For $x \leq y$ let $n = \omega_p[y]$. Then $m \leq n$ and

$$F_p x = \mathbf{Ack}^m f x \leq \mathbf{Ack}^n f y = F_p y$$

to show that F_p is a snake.

By a similar reasoning we have

$$f \sqsubseteq F_p \sqsubseteq f_\omega$$

for $p \in \mathbb{I}$. What is more interesting is the way this whole family of snakes F_\bullet live together.

2.3 LEMMA. *We have*

$$\mathbf{Ack} F_p \sqsubseteq F_q$$

for all $p, q \in \mathbb{I}$ with $p < q$.

Proof. Consider two such reals $0 < p < q < 1$. Since the function k increases without bound we have

$$\frac{2}{q-p} \leq ka$$

for all sufficiently large a . We show that

$$\mathbf{Ack} F_p a \leq F_q a$$

and hence

$$\mathbf{Ack}F_p \sqsubseteq_a F_q$$

for each such a .

Consider any such a , and fix this for the remainder of the proof.

We have

$$2 + p(ka) \leq q(ka)$$

so that

$$\omega_q[a] = \lfloor a(ka) \rfloor \geq 2$$

and hence

$$\omega_q[a] = i + 1$$

for some $i \in \mathbb{N}$.

We also fix this i for the remainder of the proof.

We have

$$F_q a = \mathbf{Ack}^{\omega[a]} f a = \mathbf{Ack}^{i+1} f a = \mathbf{Ack} f_i a$$

so that

$$\mathbf{Ack}F_p a \leq \mathbf{Ack} f_i a$$

is our objective. In fact, we show

$$x \leq a \implies \mathbf{Ack}F_p x \leq \mathbf{Ack} f_i x$$

and then take $x = a$. To prove this implication we invoke Lemma 1.7. Thus

$$x < \mathbf{Ack} f_i a \implies F_p x \leq f_i x$$

is our new objective.

Consider any

$$x < \mathbf{Ack} f_i a$$

and fix this x for the remainder of the proof. We thus have a, i , and x fixed.

Recall that $f_j \sqsubseteq_j f_\omega$ for each $j < \omega$. Since

$$i + 1 = \omega_q[a] = \lfloor q(ka) \rfloor < ka \leq a$$

we have

$$\mathbf{Ack} f_i = f_{i+1} \sqsubseteq_{i+1} f_\omega$$

so that

$$\mathbf{Ack} f_i a \leq f_\omega a$$

(since $i + 1 \leq a$) to give

$$x < f_\omega a$$

from the position of x . Thus

$$kx \leq 1 + ka$$

by our choice of k .

From this comparison we have

$$p(kx) \leq p + p(ka) < 1 + p(ka)$$

so that

$$1 + p(ka) < 2 + p(ka) \leq q(ka)$$

to give

$$1 + \omega_p[x] \leq \omega_q[a] = i + 1$$

and hence

$$\omega_p[x] \leq i$$

for this i and x .

Finally, we recall that $f_j \sqsubseteq_0 f_i$ for $j \leq i$. This, with $j = \omega_p[x]$ we have

$$\mathbf{Ack}^{\omega_p[x]} f = f_j \sqsubseteq_0 f_i$$

and hence evaluation at x gives

$$F_p x = \mathbf{Ack}^{\omega_p[x]} f x \leq f_i x$$

for the required result. ■

Next we use an iterated version of eventual domination.

2.4 DEFINITION. For each pair f and g of fast snakes let

$$f \triangleleft g \quad \text{mean} \quad (\forall i)[f_i \sqsubseteq g]$$

using the ladder of f . ■

With this notation Lemma 2.3 can be strengthened as follows.

2.5 THEOREM. *For each fast snake f there is an \mathbb{I} -indexed family of snakes*

$$(F_p \mid p \in \mathbb{I})$$

such that

$$f \triangleleft F_p \triangleleft f_\omega \quad p < q \implies F_p \triangleleft F_q$$

for all reals $p, q \in \mathbb{I}$.

Proof. Consider first a pair of reals $0 < p < q < 1$. We require

$$\mathbf{Ack}^i F_p \sqsubseteq F_q$$

for an arbitrary $i < \omega$. But for any reals

$$p = p(0) < p(1) < p(2) < \dots < p(i) < q$$

we have

$$\mathbf{Ack} F_{p(0)} \sqsubseteq F_{p(1)} \quad \mathbf{Ack} F_{p(1)} \sqsubseteq F_{p(2)} \quad \dots \quad \mathbf{Ack} F_{p(i)} \sqsubseteq F_q$$

by Lemma 2.3, and hence

$$\mathbf{Ack}^i F_p \sqsubseteq F_q$$

by the monotone property of \mathbf{Ack} .

Since

$$f \sqsubseteq F_p \sqsubseteq f_\omega$$

the required left hand condition follows in the same way. ■

For reals $p < q$ the leap in rate of growth from F_p to F_q is quite large. We show that for rational $p < q$ this leap ensures that F_q is not primitive recursive in F_p .

3 Complexity of relations

The complexity of a function is determined by two components; its rate of growth and the complexity of its graph. In the previous two sections we have dealt with the rate of growth (or as much as we need here). In this and the next section we deal with the complexity of relations in general, and graphs of functions in particular.

Consider an n -placed function ϕ . The graph $\Gamma(\phi)$ of ϕ is the $1 + n$ placed relation

$$x_0 = \phi(x_1, \dots, x_n)$$

(for $x_0, x_1, \dots, x_n \in \mathbb{N}$). How can we measure the complexity of this relation?

More generally, how can we measure the complexity of an m -placed relation

$$R(x_1, \dots, x_m)$$

over \mathbb{N} ? Often the complexity of such a relation R is defined to be the complexity of its characteristic function $\chi(R)$ given by

$$\chi(R)(x_1, \dots, x_m) = \begin{cases} 1 & \text{if } R(x_1, \dots, x_m) \\ 0 & \text{if not} \end{cases}$$

for $x_1, \dots, x_m \in \mathbb{N}$. Here that is a bit too crude. We are measuring the complexity of functions only up to primitive recursion, but for relations we need a finer stratification.

As a motivating example let's look at the graph of the snake **Ack** f associated with a fast snake f . In fact, let's look at something more general, the 3-placed relation

$$F(r, x, y) \text{ given by } f^{r+1}x = y$$

associated with f . We then have

$$\mathbf{Ack}fx = y \iff F(x, x, y)$$

for $x, y \in \mathbb{N}$. To test whether or not

$$F(r, x, y)$$

we use the strictly inflationary property

$$x < fx < f^2x < \dots < f^{r+1}x$$

of f . With this we see that $F(r, x, y)$ unravels as follows.

There is a sequence

$$a_0, a_1, \dots, a_r$$

taken from \mathbb{N} where

$$a_0 = x \quad (\forall i < r)[a_{i+1} = fa_i]$$

and where $y = fa_r$.

To test whether or not $F(r, x, y)$ we must survey all those sequences from \mathbb{N} of a certain kind. We certainly know a bound on the length of such a sequence, namely $1 + r$ using the input r . Also, because of the nature of f we have

$$a_0 \leq a_1 \leq \cdots a_r \leq y$$

to give us a bound on the size of each component of the sequence. There are only finitely many sequences we have to look at, and we know what these are before we start the test.

To be more precise we use a result which more or less goes back to Gödel.

3.1 PROPOSITION. *There are two ‘very simple’ functions*

$$\text{seq} : \mathbb{N}^3 \longrightarrow \mathbb{N} \quad \text{bnd} : \mathbb{N}^2 \longrightarrow \mathbb{N}$$

such that for each sequence

$$a_0, a_1, \dots, a_r$$

from \mathbb{N} , if

$$(\forall i \leq r)[a_i \leq x]$$

then

$$(\forall i \leq r)[\text{seq}(s, i) = a_i]$$

for some $s < \text{bnd}(r, x)$.

We need not say what ‘very simple’ means. There are several examples of such pairs of functions, all of them in the lower reaches of the set of primitive recursive functions. In fact, all of them are Kalmár elementary, and some are even simpler. Thus, for our purposes the two functions have trivial complexity.

With these two functions we see that

$$(\exists s < \text{bnd}(r, x)) \left[\begin{array}{c} \text{seq}(s, 0) = x \\ \wedge \\ (\forall i \leq r)[\text{seq}(s, i') = f\text{seq}(s, i)] \\ \wedge \\ f\text{seq}(s, r) = y \end{array} \right]$$

is a rephrasing of $F(r, x, y)$.

Look how this formal description is built up. We certainly use the graph of f , but everything else is built up by boolean combinations and bounded quantification, and for our purposes these are modifications of trivial complexity.

3.2 DEFINITION. Let \mathcal{S} be a family of relations. A relation R is **rudimentary** in \mathcal{S} if R can be obtained from finitely many members of \mathcal{S} by boolean combinations and bounded quantification. Within this expression certain predetermined functions may occur, either in the body or as bounds in the quantifiers. These include the standard initial functions for primitive recursion and the chosen sequence coding functions **seq** and **bnd**. ■

This definition is a bit wordy. We could be more precise, but that would take longer to say. All we need here is a rough idea of the notion.

The idea and terminology comes from [10], and the notion is roughly the same as used there. However, the precise connection depends on the ‘predetermined functions’, in particular on **seq** and **bnd**. Here we don’t need to be so specific.

Using this notion the calculations above show the following.

3.3 LEMMA. For each fast snake f the 3-placed relation

$$f^{r+1}x = y$$

is rudimentary in the graph of f .

Our second example is not quite so straight forward. We look at the function k given by Lemma 1.8. We start from an arbitrary fast snake h and produce k by

$$kx = r \iff h^r 0 \leq x < h^{r+1} 0$$

for $x, r \in \mathbb{N}$. Thus an instance

$$kx = r$$

of the graph of k unravels as follows.

There is a sequence

$$a_0, a_1, \dots, a_r$$

taken from \mathbb{N} where

$$a_0 = 0 \quad (\forall i < r)[a_{i+1} = ha_i]$$

and where $a_r \leq x < ha_r$.

The input r gives us a bound on the length of the sequence, and the input x gives a bound on the size of the components. Thus we have a rephrasing of $kx = r$.

$$(\exists s < \text{bnd}(r, x)) \left[\begin{array}{c} \text{seq}(s, 0) = x \\ \wedge \\ (\forall i \leq r)[\text{seq}(s, i') = h\text{seq}(s, i)] \\ \wedge \\ \text{seq}(s, r) \leq x < h\text{seq}(s, r) \end{array} \right]$$

At first sight this seems to show that the graph of k is rudimentary in the graph of its parent snake h . But a closer look brings out a problem. We certainly use the graph of h in the rephrasing, but we also use the 2-placed relation

$$x < ha$$

which could be more complicated than the graph of h .

To obtain the result we need we have to be a bit more particular.

The next section is devoted to proving the following.

3.4 THEOREM. Let f be a fast snake with associated 3-placed Ackermann function A . Then the 4-placed relation

$$A(i, r, x) \leq y$$

is rudimentary in the graph of f .

Using this we can obtain the results we need.

3.5 LEMMA. Let f be a fast snake, and let $h = f_\omega$. Then each of the following 2-placed relations

$$hu \leq v \quad hu = v \quad v < hu$$

is rudimentary in the graph of f .

Proof. By Theorem 3.4 it is sufficient to show that each of the 2-placed relations is rudimentary in the 4-placed relation $A(i, r, x) \leq y$.

Since

$$hu = f_\omega u = f_u u = A(u, 0, u)$$

we have

$$hu \leq v \iff A(u, 0, u) \leq v$$

which deals with the left hand relation.

Next we have

$$v < hu \iff \neg[hu \leq v]$$

which deals with the right hand relation.

Finally we have

$$hu = v \iff (v \neq 0) \wedge (hu \leq v) \wedge \neg(hu \leq v - 1)$$

which deals with the graph of h . ■

With this we see that the second calculation above shows the following.

3.6 LEMMA. Let f be a fast snake, let $h = f_\omega$, and let k be the associated slow snake (as given by Lemma 1.8). Then the graph of k is rudimentary in the graph of f .

This result depends on Theorem 3.4 which is proved in the next section. Of course, we must make sure we don't use Lemma 3.5 in that proof.

4 The derivation technique

This section is devoted to a proof of Theorem 3.4. Of course, that result is not unknown, but the method of proof used here may have independent interest.

We fix a fast snake f and consider the relation

$$A(i, r, x) \leq y$$

which, at first sight, could be more complicated than the graph of A .

To prove the result we first set up a recursive description of the relation. The following result should be compared with the method of proof of Theorem 1.6.

4.1 LEMMA. Let f be a fast snake with associated 3-placed Ackermann function A . Then we have

$$(B) \quad A(0, 0, x) \leq y \iff fx \leq y$$

$$(O) \quad A(i', 0, x) \leq y \iff A(i, x, x) \leq y$$

$$(I) \quad A(i, r', x) \leq y \iff (\exists z < y)[A(i, 0, z) \leq y \text{ and } A(i, r, x) \leq z]$$

for all $i, r, x, y \in \mathbb{N}$.

Proof. Only (I) is not immediate.

(\Rightarrow) Suppose $A(i, r', x) \leq y$. Then with

$$z = A(i, r, x)$$

the inflationary property of $A(i, 0, \cdot)$ gives

$$z < A(i, 0, z) = A(i, r', x) \leq y$$

as required.

(\Leftarrow) Suppose there is such a z . Then

$$A(i, r', x) = A(i, 0, w)$$

where

$$w = A(imr, x) \leq z$$

so that the monotone property of $A(i, 0, \cdot)$ gives

$$A(i, r', x) = A(i, 0, w) \leq A(i, 0, z) \leq y$$

as required. ■

To test $A(i, r, x) \leq y$ we track through the recursive description given by Lemma 4.1. Our problem is to show that we can put a bound on the length of such a test before we start the tracking.

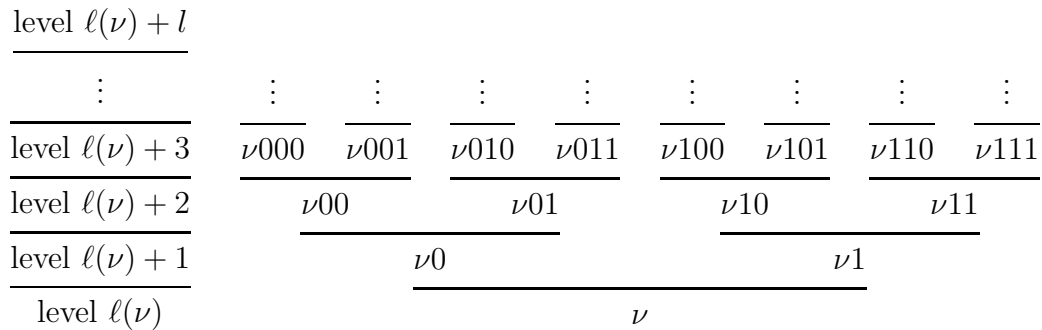
To carry out such a test we set up a rather primitive derivation system. This manipulates judgements

$$(\nu, l) \vdash (i, r, x, y)$$

with components as follows.

- (ν) This is a node of \mathbb{C} , the Cantor tree. We recall the details of \mathbb{C} in a moment. The job of ν is merely a hook on which to hang the 4-tuple (i, r, x, y)
- (l) This a control value, $l \in \mathbb{N}$. As we will see later, it puts a bound on the height (and hence size) of the whole derivation.
- (i, r, x, y) These are the inputs to the relation we wish to test.

Recall that \mathbb{C} , the Cantor tree, is the full binary splitting tree. It has a root \perp and starting with that each node ν splits into two immediate successors $\nu 0$ and $\nu 1$, each of which splits, and so on. The whole tree has infinite height, but we use only finite convex parts.



Starting with any node ν we may rise to a level l above that of ν . We may omit some of the nodes, but once omitted no higher node can appear. Thus the whole convex part has $< 2^{l+1}$ nodes.

A derivation

∇

of the system is a collection of judgements hung on the nodes of such a finite convex part of \mathbb{C} . Of course, ∇ is built up according to certain rules, which we look at later. We write

$$(\nabla) \quad (\nu, l) \vdash (i, r, x, y)$$

to indicate that ∇ is a derivation with its root, the indicated judgement, hung at node ν . The control l puts a bound on the convex region used by ∇ .

We prove two results about this derivation system. These provide a kind of soundness and adequacy, respectively.

(Sound) If

$$(\nabla) \quad (\nu, l) \vdash (i, r, x, y)$$

then $A(i, r, x) \leq y$.

(Adequate) If $A(i, r, x) \leq y$ then, for each node ν , there is a derivation

$$(\nabla) \quad (\nu, l) \vdash (i, r, x, y)$$

with $i + r \leq l \leq yi + r$.

This completeness (soundness+adequacy) shows how $A(i, r, x) \leq y$ can be tested. We look for a derivation

$$(\nabla) \quad (\perp, l) \vdash (i, r, x, y)$$

hung at the root of \mathbb{C} . We may restrict the control by $l \leq yi + r$. There are only finitely many such derivations, and we can put a bound on this number.

This doesn't quite prove Theorem 3.4, but we can begin to see the idea.

You may wonder why we allow a derivation to have its root at an arbitrary node of \mathbb{C} . This is just a technical convenience. It helps us to combine smaller derivations into larger ones. Any derivation

$$(\nabla_\nu) \quad (\nu, l) \vdash (i, r, x, y)$$

at a node ν can be replicated

$$(\nabla_\mu) \quad (\mu, l) \vdash (i, r, x, y)$$

at any node μ . It is the size and shape of a derivation that matters, not where we hang it.

Each derivation is generated from its leaves to its root. There are three rules of derivation. The (Base) rule says explicitly what can occur at a leaf. (I know the terminology is a bit upsidedown, but worse things have happened.) The other two rules

$$\begin{array}{c} \bullet \\ \hline \bullet \end{array} \text{ (Outer)} \qquad \begin{array}{c} \bullet \quad \bullet \\ \hline \bullet \end{array} \text{ (Inner)}$$

determine how one or two smaller derivations can be turned into a larger derivation.

The three rules, which correspond to the three clauses of Lemma 4.1, are listed in Table 1. Notice that before any rule can be applied a certain proviso must be met.

Let's first check that the system doesn't do too much.

Label	Rule	Proviso
(B)	$(\nu, l) \vdash (0, 0, x, y)$	$fx \leq y$
(O)	$\frac{(\nu 0, m) \vdash (i, x, x, y)}{(\nu, l) \vdash (i', 0, x, y)}$	$m < l$
(I)	$\frac{(\nu 0, n) \vdash (i, 0, z, y) \quad (\nu 1, m) \vdash (i, r, x, z)}{(\nu, l) \vdash (i, r', x, y)}$	$n, m < l$ and $z \leq y$

Table 1: Rules of derivation

4.2 LEMMA. *The derivation system is sound (in the sense explained above).*

Proof. Given a derivation

$$(\nabla) \quad (\nu, l) \vdash (i, r, x, y)$$

we must show that

$$A(i, r, x) \leq y$$

holds. We proceed by induction on the construction of ∇ .

(B) Here ∇ is a leaf

$$(\nu, l) \vdash (0, 0, x, y)$$

and this is allowed only if

$$A(0, 0, x, y) = fx \leq y$$

to give the required result.

(O) Here we have a root rule

$$\frac{(\nu 0, m) \vdash (i, x, x, y)}{(\nu, l) \vdash (i', 0, x, y)}$$

and the induction hypothesis applied to the numerator judgement gives

$$A(i', 0, x) = A(i, x, x) \leq y$$

for the required result.

(I) Here we have a root rule

$$\frac{(\nu 0, n) \vdash (i, 0, z, y) \quad (\nu 1, m) \vdash (i, r, x, z)}{(\nu, l) \vdash (i, r', x, y)}$$

so that

$$A(i, 0, z) \leq y \quad A(i, r, x) \leq z$$

by the induction hypothesis applied to the two numerator judgements. Remembering that $A(i, 0, \cdot)$ is monotone, these give

$$A(i, r', x) = A(i, 0, A(i, r, x)) \leq A(i, 0, z) \leq y$$

for the required result. ■

In this proof of soundness the control value is not used. However, adequacy is different. To achieve the best values we must be quite strict.

4.3 DEFINITION. A derivation is **tight** if at each node the control value is as small as possible. Thus

$$(B) \quad l = 0 \quad (O) \quad l = m + 1 \quad (I) \quad l = \max\{n + 1, m + 1\}$$

gives the required selection for each use of a rule. ■

We are going to show that if $A(i, r, x) \leq y$ then this comparison can be verified by a tight derivation. Furthermore, the size of the control value can be bounded before the test begins. To prove this we use a double induction analogous to that used in the proofs of Theorem 1.6 and Lemma 4.1.

For $i, r \in \mathbb{N}$ let

$$[i, r] \quad \text{abbreviate} \quad \left\{ \begin{array}{l} \text{For all } x, y \in \mathbb{N}, \text{ if} \\ \qquad \qquad \qquad A(i, r, x) \leq y \\ \text{then for each node } \nu \text{ there is a tight derivation} \\ \qquad \qquad \qquad (\nabla) \quad (\nu, l) \vdash (i, r, x, y) \\ \text{where } i + r \leq l \leq yi + r \text{ and } x < y. \end{array} \right.$$

and as before let $[i]$ abbreviate $(\forall i)[i, r]$.

4.4 LEMMA. *We have*

$$\begin{aligned} (B) \quad & [0, 0] \\ (O) \quad & [i] \implies [i', 0] \\ (I) \quad & [i, 0] \wedge [i, r] \implies [i, r'] \end{aligned}$$

for each $i, r \in \mathbb{N}$.

Proof. (B) We are given

$$x < fx = A(0, 0, x) \leq y$$

and then the leaf

$$(\nu, l) \vdash (0, 0, x, y)$$

is the required derivation.

(O) We are given

$$x < A(i, x, x) = A(i', 0, x) \leq y$$

and the hypothesis $[i]$ provides a derivation of

$$(\nu 0, m) \vdash (i, x, x, y)$$

where

$$i + x \leq m \leq yi + x$$

and $x < y$ (from above). A use of the O-rule gives

$$(\nu, l) \vdash (i', 0, x, y)$$

where $l = m + 1$. But now we have

$$i' \leq m + 1 \leq l$$

and, since $x + 1 \leq y$, we also have

$$l = m + 1 \leq yi + x + 1 \leq yi + x \leq yi'$$

as required.

(I) We are given

$$z < A(i, 0, z) = A(i, r', x) \leq y$$

where

$$z = A(i, r, x)$$

is the intermediate value. The two hypotheses $[i, 0]$ and $[i, r]$ provide derivations of

$$(\nu 0, n) \vdash (i, 0, z, y) \quad (\nu 1, m) \vdash (i, r, x, z)$$

where

$$i \leq n \leq yi \quad z < y \quad i + r \leq m \leq zi + r \quad x < z$$

hold. A use of the I-rule gives

$$(\nu, l) \vdash (i, r', x, y)$$

where

$$l = \max\{n + 1, m + 1\}$$

is the new control value. We have

$$i + r' \leq m + 1 \leq l$$

and both

$$n + 1 \leq yi + 1 \leq yi + r' \quad m + 1 \leq zi + r + 1 \leq yi + r'$$

to give

$$l \leq yi + r'$$

as required. ■

How does this prove Theorem 3.4? From the analysis and the extra information in Lemma 4.4 we have the following.

4.5 LEMMA. For each $i, r, x, y \in \mathbb{N}$ the relation

$$A(i, r, x) \leq y$$

holds precisely when there is a derivation

$$(\nabla) \quad (\perp, l) \vdash (i, r, x, y)$$

where $l \leq yi + r$ and where for each node

$$(\mu, m) \vdash (j, s, u, v)$$

of ∇ each of the comparisons

$$m \leq l \quad j \leq i \quad j + s \leq m \leq vj + s \quad u < v \leq y$$

holds.

Thus $A(i, r, x) \leq y$ holds precisely when there is a certain finite tree of 5-tuples (m, j, s, u, v) . We have a bound on the size of this tree given by $l \leq yi + r$. Also, with

$$b = \max\{yi + r, i, y\}$$

we have

$$m, j, s, u, v \leq b$$

for each 5-tuple in the tree. Each such 5-tuple can be coded as a single natural number, and then we know an upper bound to the size of these codes, say B where we may compute B from b . The whole tree can be coded as a sequence of natural numbers of length no more than $L = 2^{b+1}$, and where each component is no bigger than B . Thus

$$A(i, r, x) \leq y \iff (\exists t < \text{bnd}(L, B))[\dots]$$

where $[\dots]$ describes how the various components of a sequence $\text{seq}(t, \cdot)$ must fit together. The details are a bit messy, but essentially all we do is describe the notion of a derivation in coded form.

5 Dense chains of p.r. degrees

In this final section we at last get round to looking at the primitive recursive degrees. To explain what these are we need a bit of notation.

5.1 DEFINITION. For each set Θ of functions let

$$PR(\Theta)$$

be the clone of those functions that can be obtained from members of Θ by primitive recursion.

For each fast snake f let

$$ED(f)$$

be the set of those functions eventually dominated by f . ■

A function ϕ belongs to $PR(\Theta)$ precisely when ϕ can be constructed in a finite number of steps from the members of Θ (and the usual initial functions) by composition and primitive recursion. A primitive recursive degree is just a set of functions of the form $PR(\Theta)$ for some Θ . Such degrees are compared by inclusion.

We have

$$\phi \in ED(f) \iff \overline{\phi} \sqsubseteq f$$

for each fast snake f and functions ϕ . (Remember that $\overline{\phi}$ is the fast bounding function of ϕ .) The notation $ED(\cdot)$ is introduced here merely to make Theorem 5.2 and its Corollary 5.3 slightly easier to state.

There are several, perhaps many, proofs of Ackermann's result to be found in the literature. All the ones I know more or less prove the following.

5.2 THEOREM. *Let Θ be a family of functions and let f be a fast snake with $\Theta \subseteq ED(f)$. Then for each $\phi \in PR(\Theta)$ we have*

$$\phi \sqsubseteq \mathbf{Ack}^i f = f_i$$

for some $i < \omega$.

The proof of this gives a bound on the iteration level i . It is essentially the number of steps needed to get from Θ to ϕ .

Ackermann's result, Theorem 5.2, shows us how 'being primitive recursive in' and 'being eventually dominated by' are related.

For a fast snake f consider the 2-placed function F given by

$$F(r, x) = f^{r+1}x$$

(for $r, x \in \mathbb{N}$). We have $F \in PR(f)$, so that $\mathbf{Ack}f \in PR(f)$, and hence $f_i \in PR(f)$ for each $i < \omega$.

5.3 COROLLARY. *We have*

$$f \triangleleft g \iff PR(f) \subseteq ED(g)$$

for all fast snakes f and g .

Proof. (\Rightarrow) Suppose $f \triangleleft g$ and consider any $\phi \in PR(f)$. By Theorem 5.2 we have

$$\overline{\phi} \sqsubseteq f_i$$

for some $i < \omega$, and hence

$$\overline{\phi} \sqsubseteq g$$

since $f_i \sqsubseteq g$.

(\Leftarrow) Suppose $PR(f) \subseteq ED(g)$. For each $i < \omega$ we have $f_i \in PR(f)$, and hence $f_i \sqsubseteq g$, as required. ■

For a snake f let $\chi(f)$ be the characteristic function of the graph $\Gamma(f)$ of f . Thus, in any reasonable sense of the words, the function $\chi(f)$ and the relation $\Gamma(f)$ have the same complexity. In particular, for snakes f and g , if $\Gamma(f)$ is rudimentary in $\Gamma(g)$ then $\chi(f) \in PR(\chi(g))$.

There is a partial converse to Corollary 5.3 which we will need a couple of times.

5.4 LEMMA. *Let f and h be fast snakes with $f \sqsubseteq h$. Then $f \in PR(h, \chi(f))$.*

Proof. We have $f \sqsubseteq_a h$ for some $a \in \mathbb{N}$. With

$$b = \max\{f0, \dots, fa\}$$

we have

$$fx = (\text{least } y \leq \max\{b, hx\})[fx = y]$$

which is easily massaged into a primitive recursive construction of f from h and $\chi(f)$. ■

In fact, this proof shows that f is Kalmár elementary in $\{h, \chi(f)\}$, but we need not pursue that here.

With these observations we easily obtain a relativized version of the more usual form of Ackermann's result. To motivate what we do later, let's have a look at that.

For an arbitrary fast snake f let

$$\mathcal{C}_0(f) = PR(f) \quad \mathcal{C}_1(f) = PR(f_\omega, \chi(f))$$

to obtain two primitive recursive degrees. How are these related?

5.5 THEOREM. *For each fast snake f we have $\mathcal{C}_0(f) \subsetneq \mathcal{C}_1(f)$.*

Proof. We have $f \sqsubseteq f_\omega$ so that Lemma 5.4 gives $f \in \mathcal{C}_1(f)$, and hence $\mathcal{C}_0(f) \subseteq \mathcal{C}_1(f)$. By way of contradiction, suppose $\mathcal{C}_0(f) = \mathcal{C}_1(f)$. Then

$$f_\omega \in \mathcal{C}_1(f) = \mathcal{C}_0(f) = PR(f)$$

so that, by Theorem 5.2, we have

$$f_\omega \sqsubseteq \mathbf{Ack}^i f$$

for some $i < \omega$. With $h = \mathbf{Ack}^i f$ this gives

$$\mathbf{Ack}h \sqsubseteq f_\omega \sqsubseteq h$$

which is the contradiction. ■

We are going to strengthen this result by interpolating in the interval $\mathcal{C}_0(f) \subseteq \mathcal{C}_1(f)$ a dense chain of primitive recursive degrees. To do that we use the functions F_p of Section 2, but only for rational p . The following shows why we need this restriction.

5.6 LEMMA. *Let f be a fast snake with associated snake F_p for some real $p \in \mathbb{I}$. Then*

$$(F_p x = y) \iff (\exists i, u \leq x)[(i \leq pu < i + 1) \wedge (kx = u) \wedge (A(i, 0, x) = y)]$$

for each $x, y \in \mathbb{N}$.

Proof. (\Rightarrow) Suppose $F_p x = y$. Let

$$u = kx \quad i = \omega_p[x] = \lfloor pu \rfloor$$

so that

$$i \leq pu < i + 1 \quad i < u = kx \leq x$$

with

$$A(i, 0, x) = \mathbf{Ack}^i fx = F_p x = y$$

as required.

(\Leftarrow) This follows by more or less the same reasoning. ■

What is the complexity of the graph $\Gamma(F_p)$. From Sections 3 and 4 we know that each of $\Gamma(k)$ and $\Gamma(A)$ is rudimentary in $\Gamma(f)$, to show that $\Gamma(F_p)$ is rudimentary in $\Gamma(f)$ combined with the complexity of the real p . That can be quite high. Thus we stick to rational numbers since these have trivial complexity.

Recall that \mathbb{J} is the set of rational numbers p with $0 < p < 1$

5.7 COROLLARY. *Let f be a fast snake with associated snake F_p for rational $p \in \mathbb{J}$. Then $\Gamma(F_p)$ is rudimentary in $\Gamma(f)$ and $\chi(F_p) \in PR(\chi(f))$.*

With this we can add to the degrees $\mathcal{C}_0(f)$ and $\mathcal{C}_1(f)$.

5.8 DEFINITION. Let f be a fast snake. For each rational $p \in \mathbb{J}$ let

$$\mathcal{C}_p(f) = PR(F_p, \chi(f))$$

to obtain a family of primitive recursive degrees.

It now doesn't take too long to prove our main result, the extension of Theorem 5.5.

5.9 THEOREM. *For each fast snake f we have*

$$\mathcal{C}_0(f) \subsetneq \mathcal{C}_p(f) \subsetneq \mathcal{C}_1(f) \quad p < q \implies \mathcal{C}_p(f) \subsetneq \mathcal{C}_q(f)$$

for all rationals $p, q \in \mathbb{J}$.

Proof. Let us concentrate on the right hand implication.

Consider rational $p < q$ in \mathbb{J} . Since $F_p \sqsubseteq F_q$ we have

$$F_p \in PR(F_q, \chi(F_p))$$

by Lemma 5.4, and hence $F_q \in \mathcal{C}_q(f)$ by Corollary 5.7. This gives $\mathcal{C}_p(f) \subseteq \mathcal{C}_q(f)$.

By way of contradiction, suppose $\mathcal{C}_p(f) = \mathcal{C}_q(f)$. Then

$$F_q \in \mathcal{C}_q(f) = \mathcal{C}_p(f)$$

so that, By Theorem 5.2, we have

$$F_q \sqsubseteq \mathbf{Ack}_i F_p$$

for some $i < \omega$. With $h = \mathbf{Ack}^i F_p$, a use of $F_p \triangleleft F_q$ gives

$$\mathbf{Ack} h \sqsubseteq \mathbf{Ack}^i F_p \sqsubseteq F_q = h$$

which is the contradiction.

The left hand interpolation follows in the same way. ■

This completes our analysis.

References

- [1] W. Ackermann: Zum Hilbertschen Aufbau der reellen Zahlen, *Math. Annalen* 99 (1928) 118-133.
- [2] M.A. McBeth: *Combinatorial number theory*, The Edwin Mellor Press (1994), [ISBN 0-7734-9085-X].
- [3] P.G. Odifreddi: *Classical recursion theory*, Vols I and II, Elsevier (1999).
- [4] R. Péter: Konstruktion nichtrekursiver Funktionen, *Math. Annalen* 111 (1935) 42-60.
- [5] R. Péter: *Recursive functions* Academic Press (1967).
- [6] R.M. Robinson: Recursion and double recursion, *Bull. Amer. Math. Soc.* 54 (1948) 987-993.
- [7] H. Rose: *Subrecursion, functions and hierarchies*, Oxford U.P. (1984).
- [8] H. Simmons: *A density property of the primitive recursive degrees*, Dept. Of Computer Science, Victoria University of Manchester, Tech. Report UMCS-93-1-1.
- [9] C. Smoryński: *Logical number theory I*, Springer (1991).
- [10] R.M. Smullyan: *Theory of formal systems*, Annals of Mathematics Studies, No. 47, Princeton University Press (1961).