NEW FIXED POINT COMBINATORS FROM OLD

JAN WILLEM KLOP

Vrije Universiteit Amsterdam
e-mail address: jwk@cs.vu.nl

Abstract. There are two themes in this note. The first is how to derive new fixed point combinators from given ones. Here we extend the derivation principle leading to the Böhm sequence of fpc's. The second is the question how to prove that the new fpc's are indeed new. More general, we tentatively present a method to discriminate terms as to β-convertibility that goes beyond the classical Böhm-out technique, and exploits the clock behaviour that is inherent in a λ-term.

Dedicated to Henk Barendregt, in celebration of his 60th anniversary

> *The theory of sage birds (technically called* fixed point combinators*) is a fascinating and basic part of combinatory logic; we have only scratched the surface.*
>
> R. Smullyan, To Mock a Mockingbird, 1985.

1. How to make fixed point combinators

Let's start with reviewing some old recipe's to construct fixed point combinators, or in abbreviation, fpc's.

1.1. **Curry's fixed point combinator.** The simplest fpc is the one known as *Curry's fpc*. It is constructed as follows (See Figure 1). Let F be a λ-term. We want to have a 'fixed point' of F, i.e., a term X such that $FX = X$. We try to construct a term Y satisfying $YF = F(YF)$. For, then YF is a fixed point as desired. Our construction will even be 'uniform' for every F. We try to find a term Ω_F depending on F (as suggested by the subscript) such that $\Omega_F \twoheadrightarrow F(\Omega_F)$. Now suppose that $\Omega_F = \omega_F\omega_F$, with the first ω_F meant for the 'control' and the second ω_F meant for the replication of the original ω_F. So we are led to requiring $\omega_F x \twoheadrightarrow F(xx)$, and this is obtained by putting $\omega_F = \lambda x.F(xx)$. Summing up we can put $Y = \lambda f.\omega_f\omega_f = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, and this is Curry's fpc, that we will call Y_0 henceforth.

Definition 1.1. A fpc Y is *reducing*, if $\forall F : YF \twoheadrightarrow F(YF)$. So Curry's fpc Y_0 is not reducing. But the next one is reducing.

2000 ACM Subject Classification: F.4.1 [Mathematical logic and formal languages]: Lambda calculus and related systems.

Key words and phrases: fixed point combinators, lambda calculus, looping combinators, head reduction.

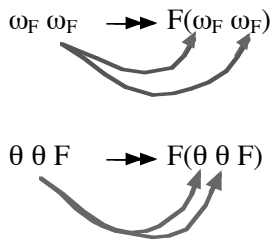

Figure 1: *The making of fpc Y_0 and Y_1.*

1.2. **Turing's fixed point combinator.** A slightly different construction yields *Turing's fpc* Y_1 having the advantage over Curry's fpc that it is a reducing one. The construction now proceeds as follows. Just as in the construction of Curry's fpc we split the task in two: $Y_1 = \eta\eta$. So we want $Y_1F = \eta\eta F \twoheadrightarrow F(\eta\eta F)$. Here the first η is the one for the control and the second η is for the replication. So we want ηxf $\twoheadrightarrow$ f(xxf). This is simple: put $\eta = \lambda xf.f(xxf)$ and finally $Y_1 = (\lambda xf.f(xxf))(\lambda xf.f(xxf))$, which is Turing's fpc.

Proposition 1.2. *The fpc's Y_0 and Y_1 are not β-convertible. Equivalently, $\neg(Y_0 \downarrow Y_1)$, that is, they have no common reduct.*

Proof. We give the simple proof in a coinductive fashion, that actually leads to an interesting question, formulated after this proof. In the following lines $'\ldots \Leftarrow - - -'$ means: we can prove $\ldots$ *only if* we can prove $---$. Now we have $(Y_0 \downarrow Y_1) = (\lambda f.\omega_f\omega_f \downarrow \eta\eta) \Leftarrow \lambda f.\omega_f\omega_f \downarrow \lambda f.f(\eta\eta f) \Leftarrow \omega_f\omega_f \downarrow f(\eta\eta f) \Leftarrow f(\omega_f\omega_f \downarrow f(\eta\eta f) \Leftarrow \omega_f\omega_f \downarrow \eta\eta f \Leftarrow f(\omega_f\omega_f) \downarrow f(\eta\eta f) \Leftarrow \omega_f\omega_f \downarrow \eta\eta f$ and we have the same task as two steps before. Therefore (!?) $\neg(Y_0 \downarrow Y_1)$. □

Problem 1.3. Intuitively, the 'principle' employed at (!?) in the proof of this proposition seems clear, but how can we make it explicit and justify it? It is certainly reminiscent to coinductive methods used in proving equations between recursively defined objects over a first order signature; but there one proves positive facts, namely equations, while here we conclude a negative fact of a more complicated nature than an equation. Anyway, one can easily give a proof using less questionable or more established principles. Moreover, in the sequel we give another proof along a very different line. Yet it seems that something interesting is at stake at this point, which we would like to understand better.

1.3. **One's own fixed point combinator.** A general recipe to construct one's own fpc is now emerging. Put $\Gamma = \gamma\gamma\cdots\gamma$ ($n \geq 2$ times). Then for $\gamma = \lambda a_1a_2\ldots a_{n-1}f.\,f(wf)$ where w is an arbitrary word of length n over the alphabet $\{a_1, a_2, \ldots, a_{n-1}\}$, we have a fpc. For instance one that proclaims its own identity:

$Y_{fpc} = (LLLLLLLLLLLLLLLLLLLLLLLLLL)$ where
$L = \lambda abcdefghijklmnopqstuvwxyzr.(r(thisisafixedpointcombinator))$

1.4. **Fpc's with dummy parameters.** Fpc's can be equipped with parameters $P_1, ..., P_m$: take $Y^{P_1,\ldots,P_m} = \zeta\zeta P_1 ... P_m$, where $\zeta = \lambda xp_1 ... p_m f.f(xxp_1 ... p_m f)$.

One of these fpc's is the following, with $\Omega = (\lambda x.xx)(\lambda x.xx)$, convenient for counterexamples and explanations in the sequel: $Y^\Omega = \zeta\zeta\Omega$, where $\zeta = \lambda xpf.f(xxpf)$. In infinitary lambda calculus λ^∞ it has the property WN^∞ (infinitary weak normalisation) but not the property SN^∞ (infinitary strong normalisation) (see Klop & de Vrijer [05]).

1.5. **Fpc's found by mechanical search.** In the computation below, as well as in the rest of these notes, we have gratefully made use of Freek Wiedijk's calculator for λ- and CL-terms. This tool can be obtained via the entrance page of http://web.mac.com/janwillemklop/iWeb.

The next fpc is due to McCune and Wos, and was found by mechanical search. Define $H = \lambda xyz.xyzy$ and $P = H(B(H(HB)B)B)(HH)$. Actually, the combinator H also appears in Smullyan [1985], as the 'Hummingbird'.

We compute using FLC (Freek's Lambda Calculator):

$Px \twoheadrightarrow$
$\boldsymbol{HH(B(Bx))(HH) \equiv C[x] \rightarrow}$
$(\lambda yz.Hyzy)(B(Bx))(HH) \rightarrow$
$(\lambda z.H(B(Bx))z(B(Bx)))(HH) \rightarrow$
$H(B(Bx))(HH)(B(Bx)) \rightarrow$
$(\lambda yz.B(Bx)yzy)(HH)(B(Bx)) \rightarrow$
$(\lambda z.B(Bx)(HH)z(HH))(B(Bx)) \rightarrow$
$B(Bx)(HH)(B(Bx))(HH) \rightarrow$
$(\lambda yz.Bx(yz))(HH)(B(Bx))(HH) \rightarrow$
$(\lambda z.Bx(HHz))(B(Bx))(HH) \rightarrow$
$Bx(HH(B(Bx)))(HH) \rightarrow$
$(\lambda yz.x(yz))(HH(B(Bx)))(HH) \rightarrow$
$(\lambda z.x(HH(B(Bx))z))(HH) \rightarrow$
$\boldsymbol{x(HH(B(Bx))(HH)) \equiv xC[x]}$.

1.6. **Weak fixed point combinators.** It makes sense to extend the class of fpc's to the *weak fpc's*, that happen to be known in foundational studies of type systems as *looping combinators*. (See e.g. Coquand & Herbelin [94], Geuvers & Werner [94].) One definition is that a wfpc is a term having the same BT as a fpc, namely $\lambda x.x^\omega$. An amusing coinductive definition is: Z is a wfpc if $Zx = x(Z'x)$ where Z' is a wfpc. So a wfpc Z is just like a fpc Y, delivering when applied on x an infinite iteration x^ω, but the generator that performs this process may change along the way.

Example 1.4. Define by double recursion, Z, Z' such that $Zx = x(Z'x)$,$Z'x = x(Zx)$. Then Z, Z' are both wfpc's, and $Zx = x^2(Zx)$. So Z delivers its output twice as fast as an ordinary fpc, but the generator flipflops. We will come back to this issue of 'production velocity' in the final section. By the way, how do we obtain double recursion? That can be done as follows:

1.7. **Multiple recursion.** Barendregt [84] gives two proofs of the double fixed point theorem. We are especially interested in the one for λI-calculus, since there it is easier to prove non-equations, a theme that will be pursued later in this note.

Theorem 1.5. $\forall F, G\, \exists A, B\ (A \twoheadrightarrow FAB\ \&\ B \twoheadrightarrow GAB)$.

Proof. We give four proofs, all for λI-calculus. (NB: this does not mean that they work *only* for λI-calculus and not for λK-calculus! It means that the constructions work *also* for λI-calculus.) The terms in those proofs will serve later as examples for methods to prove non-equations.

First proof. (Barendregt [84], p.141. Take $A_b = Y(\lambda a.Fab)$, $B_0 = Y(\lambda b.GA_bb)$, and $A_0 = A_{B_0}$. Then $A_0 \equiv A_{B_0} \equiv Y(\lambda a.FaB_0) \twoheadrightarrow (\lambda a.FaB_0)A_0 \to FA_0B_0$and $B_0 \twoheadrightarrow (\lambda b.GA_bb)B_0 \to GA_{B_0}B_0 \equiv GA_0B_0$.

Second proof, a slightly different and *prima facie* simpler variant of the preceding one. Start with the second objective $B \twoheadrightarrow GAB$ and take $B = Y(GA)$, then the first objective becomes $A \twoheadrightarrow FA(Y(GA))$. Now take $A = Y(\lambda a.Fa(Y(Ga)))$. In what respect is this solution simpler than the first? Writing out abbreviations, it turns out that the first solution in A, B respectively uses $3+2$ times Y, and the second solution $2+3$ times Y. However, the first solution uses 5 abstractions, the second only 2. Is there a 'sensible' measure deciding which solution is simpler? Below we will discuss such a measure.

Third proof (Smullyan [85], Exercise 6, p.196 and 198; also mentioned in Barendregt [90]), Theorem 2.2.17, p.334. A beautiful solution indeed: Take N such that $Ndfg \twoheadrightarrow d(Nffg)(Ngfg)$. Further, take $A = NFFG$ and $B = NGFG$. Then $A = NFFG = F(NFFG)(NGFG = FAB$, and $B = NGFG = G(NFFG)(NGFG) = GAB$.

Fourth proof. The previous proof is ingenious, but how does one come to such a construction? We give a proof starting from the heuristics as used above in the construction of Turing's fpc. It leads to a solution much resembling Smullyan's, yet presumably different. (The resemblance is only manifest if the fixed point definition of N is written out in full, using, e.g., Y_0 or Y_1.) Take $A = \alpha\alpha\beta FG \twoheadrightarrow F(\alpha\alpha\beta FG)(\beta\beta\alpha FG)$ and $B = \beta\beta\alpha FG \twoheadrightarrow G(\alpha\alpha\beta FG)(\beta\beta\alpha FG)$. To that end, take $\alpha = \lambda abfg.f(aabfg)(bbafg)$ and $\beta = \lambda abfg.g(aabfg)(bbafg)$. □

Remark 1.6. The second and fourth solution seem to lend themselves best for a generalization to arbitrary n-fold recursion.

2. Derived fixed point combinators

2.1. **The Böhm sequence.** It is well-known, as observed by C. Böhm and others, that the class of fpc's coincides exactly with the class of fixed points of the peculiar term $\delta = \lambda ab.b(ab)$, convertible with SI. The notation δ is convenient for calculations and stems from B. Intrigila [97]. This term also attracted the attention of R. Smullyan [85], in his beautiful fable about fpc's figuring as birds in an enchanted forest: "An extremely interesting bird is the owl O defined by the following condition: $Oxy = y(xy)$." (p.133, 134). It follows that starting with Y_0, we have an infinite sequence of fpc's $Y_0, Y_0\delta, Y_0\delta\delta, ..., Y_0\delta^{\sim n}, ...$ where we will indicate $Y_0\delta^{\sim n}$ by Y_n. Note that indeed Y_1, the notation that we had given to Turing's fpc, is correct in this naming convention. Now the question is whether all these 'derived' fpc's are really new, in other words, whether the sequence is free of duplicates. This is *Exercise 6.8.9 in Barendregt [84]. Note that we could also have started the sequence from another fpc than Curry's. Now for the sequence starting from an arbitrary fpc Y, it is actually an open problem whether that sequence of fpc's $Y, Y\delta, Y\delta\delta, ..., Y\delta^{\sim n}, ...$ is free of repetitions. All we know, applying Intrigila's theorem below, is that no two consecutive fpc's in this sequence are convertible. But let us first consider the Böhm sequence.

Theorem 2.1. *The Böhm sequence contains no duplicates.*

Proof. For the first two elements of the sequence we already have seen that they are unequal. Experimenting with the next one, $Y_2 \equiv \eta\eta\delta$, we see that the reduction graph is easy to determine in full. But for $Y_3 \equiv \eta\eta\delta\delta$, we soon find that its full reduction graph is very

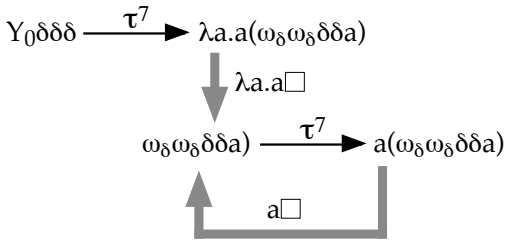

Figure 2: *Head reduction of* Y_3.

complicated. The head reduction of this term Y_3 is displayed in Figure 2.1, but this is by no means the whole reduction graph. For future reference we note that the head reduction diagram suggests a 'clock behaviour'. In order to facilitate the calculations to determine an initial part of the reduction graph of these fpc's, we remove the λ's in favour of the following applicative rules:

- $\delta x \to \gamma x$
- $\gamma xy \to y(xy)$
- $\eta x \to \gamma(xx)$
- $ax \Rightarrow x$
- $\gamma x \Rightarrow xa$ (only at the root)

Here γ is an auxiliary constant; γx can be seen as abbreviating $\lambda a.a(xa)$. Here the last two rules correspond to the removal of a hnf context; either a variable a as appearing in $\lambda a.a(xa)$ is removed (4th rule), or an abstraction is removed (5th rule). In the figure we have displayed the initial part of the reduction graph. Moreover, we indicated the passive occurrences of δ by the color red, in the color version of this paper. (An occurrence of δ is passive if it occurs as $(P\delta)$ for some P.) This yields the following invariant for the reducts of $Y_3 = \eta\eta\delta\delta$:

(1) every reduct has exactly 2 passive δ's;
(2) η has as first argument an η, as second and following arguments either a δ or an a, e.g. see $\eta\eta\delta\delta$, $\eta\eta\delta\delta a$;
(3) γ has as 2nd, 3rd and following argument always δ or a. The 1st may have various forms, as in $\gamma^n(\eta\eta)$.
(4) for δ the same clause.

Analogously for $Y_n = \eta\eta\delta^{\sim(n-1)}$; then every reduct has exactly $n-1$ passive δ's; the other clauses of the invariant are the same as above.

Note: in the invariant above clauses 2-4 cannot be omitted; they are induction loading, necessary to obtain clause 1. Establishing these invariants completes the proof that the Böhm sequence is free of duplicates. □

Smullyan [85] notes that not only does postfixing a δ to a fpc yields again a fpc, and likewise for a wfpc, prefixing it before a wfpc yields again a wfpc.

Proposition 2.2. *(Smullyan [85].) Let Z be a wfpc. Then both δZ and $Z\delta$ are wfpc's.*

The term δ looks fairly innocent, and one might think (as did the present author at some time) that all applicative combinations of this term, let's call them δ-terms, are SN. However, the term δ is a highly 'explosive' term (personal communication from Hans Zantema and

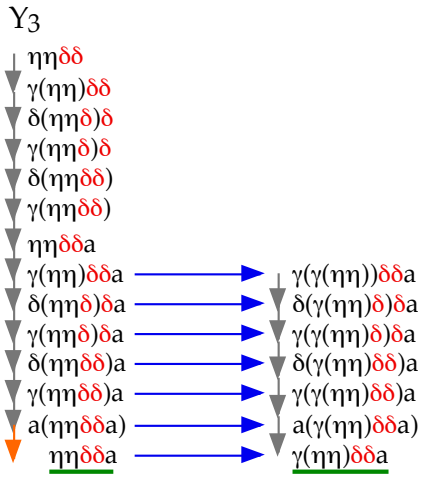

Figure 3: *Initial part of reduction graph of* Y_3.

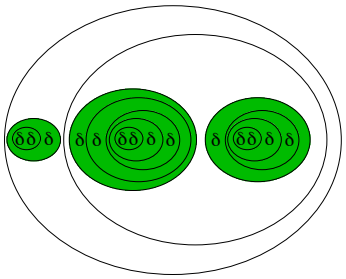

Figure 4: *δ-term with three centers.*

Johannes Waldmann.) Putting $t_n = \delta^{n-1}\delta$ we have $t_n t_m = t_m(t_{n-1}t_m)$, hence $t_2 t_k = t_k(t_1 t_k) = t_k(\delta t_k) = t_k t_{k+1}$. Now we have for $k \geqslant 3 : t_{k-1}t_k = t_k(t_{k-2}t_k) = t_k(t_k(t_{k-3}t_k)) = \ldots = t_k^p(t_2 t_k) = t_k^p(t_k t_{k+1})) = C[t_k t_{k+1}]$, which implies that $t_{k-1}t_k$ for all $k \geqslant 3$ has an infinite reduction, or in other words, is not SN. From this we obtain by induction on $n+m$ that $t_n t_m$ is not SN, for all $n, m \geq 2$. Here is the start of the 'explosion' of $\delta\delta(\delta\delta) = \delta\delta(\delta^2\delta) = t_2 t_3$, writing α for $\delta\delta$:

$$\alpha\alpha = \delta\delta\alpha \to \gamma\delta\alpha \to \alpha(\delta\alpha) \to \gamma\delta(\delta\alpha) \to \delta\alpha(\delta(\delta\alpha))$$
$$\to \gamma\alpha(\delta^2\alpha) \to (\delta^2\alpha)(\alpha(\delta^2\alpha)) \to \gamma(\delta\alpha)(\alpha(\delta^2\alpha)) \to \alpha(\delta^2\alpha)(\delta\alpha(\alpha(\delta^2\alpha)) \to \ldots$$

Proposition 2.2 provides us with the following idea. The proposition states that for a wfpc Z, any term $\delta\delta\delta\ldots\delta\delta Z\delta\delta\delta\ldots\delta\delta\delta$, with brackets not associated as usual to the left or the right, but *centered around* Z, is again a wfpc. So although δ is highly explosive material, if we apply it centering around a wfpc, it is innocent. What if we take applicative combinations of δ, centered around a δ itself? Would that also be innocent? Surprisingly it is:

Proposition 2.3. *Let t be a nontrivial δ-term, i.e. not a single δ. Then: t is SN* $\Leftrightarrow$ *t contains exactly one occurrence of* $(\delta\delta)$*, a 'center'.*

Furthermore, if δ-terms t, t' are SN, then they are convertible iff t, t' have the same length.

Problem 2.4. Is convertibility decidable for all δ-terms?

A very interesting theorem involving δ was proved by B. Intrigila, settling a conjecture by R. Statman in the negative. We will put it in a wider (conjectured) perspective at the end of this note.

Theorem 2.5. *(Intrigila [97].)*
There is no 'double' fixed point combinator.
I.e. For no fpc Y we have $Y\delta =_\beta Y$.

Problem 2.6. Is Intrigila's theorem also valid for wfpc's: for no wfpc Z we have $Z\delta = Z$?

Remark 2.7. (*Infinitary fixed point combinators.*) For a fpc Y, $Y\delta$ is again a fpc. Now we can compute in infinitary λ-calculus: $Y\delta \to_\omega (\lambda f.f^\omega)\delta \to \delta^\omega$. The infinite term δ^ω is also remarkable. Indeed it is a fpc: $\delta^\omega x \equiv \delta(\delta^\omega)x \to\!\!\to x(\delta^\omega x)$. It can also be normalized again: $\delta^\omega \to_\omega \lambda f.f^\omega$. There are many more infinitary fpc's, e.g. for every n, the infinite term $(SS)^\omega S^{\sim n} I$ is one. Why this is so, will be clear from the sequel.

Remark 2.8. The postfix $B\delta\delta$ turns every fpc Y into a wfpc delivering its output twice as fast as Y: $Y(B\delta\delta)x \twoheadrightarrow x^2(Y(B\delta\delta)x)$.

Conjecture 2.9. *The term δ is the only λI-term that uniformly transforms a given fpc into a new fpc by postfixing.*

Remark 2.10. (i) Hans Zantema (personal communication) obtained some further interesting information about δ-terms. Above we saw that $\delta\delta(\delta\delta)$ has an infinite head reduction, which makes it unsolvable in Barendregt's classical definition. Its BT therefore is trivial, namely Ω. But the Berarducci tree BeT of $\delta\delta(\delta\delta)$ is not trivial: Zantema proved that δ-terms are 'top-terminating'. (Strictly speaking, this does not yet imply that BeT$(\delta\delta(\delta\delta))$ is non-trivial, since Zantema restricted himself to the applicative rule for δ; we expect that Zantema's observation remains valid for the full λ-version.) The same top-termination holds according to Zantema for infinite δ-terms, of which an interesting example is $\delta^\omega(\delta^\omega)$. Cf. also the discussion of YY in the setting of BeT's in Dezani et al [03]). Zantema's results are reminiscent of the situation of S-terms, analysed by J. Waldmann. In particular Waldmann [98] showed that normalisation of S-terms is decidable, just as for δ-terms; we wonder whether there is a connection. Both for S-terms and for δ-terms the word problem is open. For S-terms, Waldmann also showed top-termination.

2.2. **The Scott sequence.**

Definition 2.11. (Arithmetical sequence of λ-terms)

(1) Let $A, B, C, D, E, ...$ be an arbitrary sequence of terms. Then we will call the sequence $A, AB, ABC, ABCD, ABCDE, ...$ an *applicative sequence* of terms.
(2) A particular case is a sequence of the form $M, MP, MPP, MPPP$, ... , that we will call an *arithmetical sequence.*

So the Böhm sequence of fpc's is an arithmetical sequence.

2.3. **The equation $BY = BYS$.** In Scott [75] the equation $BY = BYS$, with B and S as the usual combinators and Y a fpc, is mentioned as an interesting example of an equation not provable in $\lambda\beta$, while easily provable with Scott's Induction Rule.[1]Scott mentions that he expects that using 'methods of Böhm' the non-convertibility in $\lambda\beta$ can be established, but that he did not attempt a proof. On the other hand, with the induction rule the equality is easily established. Our first remark is that indeed the equation holds in the infinitary lambda calculus $\lambda\beta^\infty$, which seems to have Scott's Induction Rule (SIR) incorporated. (The relation of SIR versus infinitary lambda calculus $\lambda\beta^\infty$ should be determined more carefully, eventually.) Indeed, a straightforward calculation shows that in $\lambda\beta^\infty$, we have $BY = BYS = \lambda ab.(ab)^\omega$. How does one encounter this equation $BY = BYS$? Henk Barendregt mentioned in personal communication how the equation may originate: Suppose we want a term M satisfying, for given P, Q: $MP = QP(MP)$. There are two solutions. One is to take $MP = Y(QP) = BYQP$, which is obtained by taking $M = BYQ$. The other solution is, first writing $MP = SQMP$, and taking $M = Y(SQ) = BYSQ$. The question whether these solutions are the same, then amounts to the question whether the equation $BY = BYS$ holds. That the equation is not provable in $\lambda\beta$, is a nice one-line proof. Here we take for the fpc Y, Curry's fpc Y_0, just as in Scott [75].

Proposition 2.12. *$BY_0 \neq_\beta BY_0S$.*

Proof. Postfixing the combinator I yields BY_0I and BY_0SI. Now $BY_0I =_\beta Y_0$ and $BY_0SI =_\beta Y_0(SI) = Y_1$. Because (Proposition 1.2) $Y_0 \neq_\beta Y_1$, the result follows. In the same breath we can strengthen this non-equation to all fpc's Y, by the same calculation followed by an application of Theorem 2.5 stating that for no fpc Y we have $Y = Y\delta = Y(SI)$. □

Remark 2.13. The idea of postfixing an I is suggested by the BT $\lambda ab.(ab)^\omega$ of BY and BYS. Namely, in $\lambda\beta^\infty$ we calculate: $(\lambda ab.(ab)^\omega)I = \lambda b.(Ib)^\omega = \lambda b.b^\omega$ which is the BT of a fpc.

2.4. **A plethora of derived fixed point combinators.** Actually, the comparison between the terms BY and BYS has more in store for us than just providing an example that the extension from finitary lambda calculus $\lambda\beta$ to infinitary lambda calculus $\lambda\beta^\infty$ is not conservative. The BT-equality of BY and BYS suggests looking at the whole arithmetical sequence BY, BYS, $BYSS$, $BYSSS$, ..., $BYS^{\sim n}$, ..., that we will indicate as the *Scott sequence*. By the congruence property of BT-equality, they all have the same BT $\lambda ab.(ab)^\omega$; so the terms in the Scott sequence are not fpc's. But they are close to being fpc's, for the first two terms in the sequence we already saw above that postfixing an I turns them into fpc's Y_0, Y_1. How about postfixing an I to all the terms in the Scott sequence, yielding the sequence $BYI, BYSI, BYSSI, BYSSSI, ..., BYS^{\sim n}I, ...$? Surprisingly, all these terms are fpc's, the sequence concurs with the Böhm sequence of fpc's only for the first two elements, and then splits off with different fpc's. But there is a second surprise. In the proof that $BYSSI$ (and following) terms are indeed fpc's, we find a new derivation principle for fpc's, turning an old fpc into a new one. Let's call the derivation principle from Böhm, stating that postfixing a δ yields a new fpc: principle (δ). Now we have a second derivation principle, let's call it (σ), stating that postfixing a vector of terms $(SS)S^{\sim n}I$ yields a new fpc. We can arbitrarily apply derivation principles (δ) and (σ), and so obtain starting from a given fpc, a whole rooted tree of new fpc's.

[1]This equation is also discussed in Dezani et al. [03].

Theorem 2.14. *(i) If Y is a fpc, then $Y\delta$ is a fpc.*
(ii) If Y is a fpc, then $Y(SS)S^{\sim n}I$ is a fpc.

Example 2.15. $Y(SS)SIx \rightarrow$
$(\lambda x.SS(xx))(\lambda x.SS(xx))SIx \rightarrow$
$SS((\lambda x.SS(xx))\lambda x.SS(xx))SIx \rightarrow$
$(\lambda x.Sz(yz))((\lambda x.SS(xx))\lambda x.SS(xx))SIx \rightarrow$
$(\lambda z.Sz((\lambda x.SS(xx))(\lambda x.SS(xx))z))SIx \rightarrow$
$SS((\lambda x.SS(xx))(\lambda x.SS(xx))S)Ix \rightarrow$
$(\lambda yz.Sz(yz))((\lambda x.SS(xx))(\lambda x.SS(xx))S)Ix \rightarrow$
$(\lambda z.Sz((\lambda x.SS(xx))(\lambda x.SS(xx))Sz))Ix \rightarrow$
$SI((\lambda x.SS(xx))(\lambda x.SS(xx))SI)x \rightarrow$
$(\lambda yz.Iz(yz))((\lambda x.SS(xx))(\lambda x.SS(xx))SI)x \rightarrow$
$(\lambda z.Iz((\lambda x.SS(xx))(\lambda x.SS(xx))SIz))x \rightarrow$
$Ix((\lambda x.SS(xx))(\lambda x.SS(xx))SIx) \rightarrow$
$x((\lambda x.SS(xx))(\lambda x.SS(xx))SIx)$.

Remark 2.16. Another fpc 'generating vector' is obtained as follows. Start with the equation $Mab = ab(Mab)$; solutions all have the BT seen above, $\lambda ab.(ab)^\omega$. For every M satisfying this equation, we have that MI is a fpc. For: $MIx = Ix(MIx) = x(MIx)$. Now we can solve the equation in different ways. The first is: $Mab = Y(ab)$, so $M = \lambda ab.Y(ab) = (\lambda yab.y(ab))Y = BY$, as found before. The second is $Mab = ab(Mab) = Sa(Ma)b$, which is obtained by solving $Ma = Sa(Ma)$, leading to $Ma = Y(Sa) = BYSa$, so $M = BYS$. Also this solution was considered before. The third is $M = \lambda ab.(Mab) = (\lambda mab.(mab))M$, yielding $M = Y\varepsilon$ with $\varepsilon = \lambda abc.bc(abc)$. And this yields a new fpc generating vector, because for every fpc Y, $Y\varepsilon I$ is a fpc: $Y\varepsilon Ix = \varepsilon(Y\varepsilon)Ix = Ix(Y\varepsilon Ix) = x(Y\varepsilon Ix)$.

These three schemes for generating new fixed points from old, are by no means the only ones. There are in fact infinitely many of such schemes. They can be obtained analogously to the ones that we extracted above from the equation $BY = BYS = \lambda ab.(ab)^\omega$, or the equation $Mab = ab(Mab)$. We only treat the case for $n = 3$: consider the equation $Nabc = abc(Nabc)$. Then every solution N is again a 'pre-fpc', namely NII is a fpc: $NIIx) = IIx(NIIx) = x(NIIx)$. The first solution is $N = Y\xi$ with $\xi = \lambda nabc.abc(nabc)$, yielding the fpc generating vector $...\xi II$. The second solution is $Nabc = Y(abc)$, which yields $N = (\lambda yabc.y(abc)))Y = (\lambda yabc.BBByabc)Y$. We obtain $N = BBBY$. A different calculation gives $Nabc = abc(Nabc) = S(ab)(Nab)c$. So we take $Nab = S(ab)(Nab)$, which yields $Nab = Y(S(ab)) = BBBY(BS)ab$. So $N = BBBY(BS)$, and thus we find the equation $BBBY = BBBY(BS)$, in analogy with the equation above $BY = BYS$. Also this equation spawns lots of fpc's as well as fpc generating vectors. Let's abbreviate (BS) by A. First one forms the arithmetical sequence $BBBY, BBBYA, BBBYAA, BBBYAAA, ...$ These terms all have the BT $\lambda abc.abc(abc)^\omega$. They are not yet fpc's , but only 'pre-fpc's'. But after postfixing $...II$ we do again obtain a sequence of fpc's: $BBBYII, BBBYAII, BBBYAAII, BBBYAAAII, \ldots$. Again the first two coincide with Y_0, Y_1, but the the series deviates not only from the Böhm sequence but also from the Scott sequence above. As above, the proof that a term in this sequence is indeed a fpc, yields a fpc generating vector. Thus we find e.g. the following new fpc generating schemes, which we render in a self-explaining notation:

(1) $Y \Rightarrow Y(S(AI)I$
(2) $Y \Rightarrow Y(AAA)II$
(3) $Y \Rightarrow Y(AII)$

(4) $Y \Rightarrow Y(AAI)I$
(5) $Y \Rightarrow Y(AAA)A^{\sim n}II$

(Note: scheme 3 came up out of the general search; one may recognize that it is not a new scheme, because the term AII is actually the Owl δ). We can derive many more of these schemes by proceeding with solving the general equation $Na_1a_2...a_n = a_1a_2...a_n(Na_1a_2...a_n)$ in different ways as explained above, from which we will refrain here. One final fpc generating scheme we can't resist mentioning, since it ties up with the notion of a fpc with dummy parameters mentioned in 1.4:

$Y \Rightarrow YP_1P_2...P_n$ where $P_1 = \lambda yp_2...p_nx.\, x(yp_2 \dots p_nx)$ and $P_2, ..., P_n$ are arbitrary (dummy) terms.

This concludes our fabrication of building blocks for fpc's.

3. Clock behaviour of lambda terms

As we saw, there is vast space of fpc's and there are many ways to derive new fpc's. The question is whether all these fpc's are indeed new. So we have to prove that they are not β-convertible. For the Böhm sequence we did this by an ad hoc argument based on a syntactic invariant; and this method works fine to establish lots of non-equations between the alleged 'new' fpc's that we constructed above. Still, the question remains whether there are not more 'strategic' ways of proving such inequalities. In this final section we propose a more strategic way to discriminate terms with respect to β-conversion. The idea is to extract from a λ-term more than just its BT, but also how the BT was formed; one could say, in what tempo, or in what rhythm. A BT is formed from static pieces of information, but these are rendered in a clock-wise fashion, where the ticks of the internal clock are head reduction steps, that we will indicate as τ-steps henceforth. They are coexisting with another kind of internal steps, that we will call ι-steps; these are defined to be non-head reduction steps. Third, we employ γ-steps, indicating an observation in the BT, i.e. the removal of a head normal form context. First we make a notational remark.

Notation 3.1. (i). (*Applicative notation.*) In rendering BT's there are two notations suggesting themselves. The first is the 'applicative notation', where a BT is a unary-binary tree with unary abstraction nodes and binary application nodes.This notation is suitable when we apply infinite trees on each other, or an infinite tree to a finite (term) tree, e.g. as in $(\lambda abc.abc(abc)^\omega)II$.

(ii) (*Head normal form or hnf notation.*) This is the notation favoured in Barendregt [84], and it is especially suitable for BT's, which are 'stand-alone objects' when one is not concerned with infinitary lambda calculus. We also adopt this for rendering BT's below with one minor adaptation: in a hnf such as $\lambda abc.abcM$, the abstractions are three separate unary nodes in the BT and the variable of the a-vector $abcM$ is a ternary node in the BT, splitting off to $b, c, BT(M)$. See for example the BT's of BY and BYS displayed in Figure 6. In Barendregt [84] there would be just one ternary node $\lambda abc.a\Box\Box\Box$. The reason that in this section we employ the hnf notation is that we will consider 'enriched' BT's, with a natural number along the edges of the BT leading from one BT-node to a next BT-node; and these edges are not explicitly visible in the applicative notation.

Definition 3.2. (Clock reduction) (i) For fpc's Y and wfpc's Z. The clock reduction of Y consists of an infinite sequence of head reduction steps (τ-steps) and when no head step is possible because the term is in hnf $\lambda x.Y'$or $x(Yx)$, a γ-step that removes the head context

$\lambda x.\Box$ or $x\Box$ respectively. Example: the clock reduction of Y_0 is $(\gamma\tau)^\omega$. And for $Y_1 = \eta\eta$ we have $\eta\eta \to_\tau \gamma(\eta\eta) \to_\gamma \eta\eta x \to_\tau \gamma(\eta\eta)x \to_\tau x(\eta\eta x) \to_\gamma \eta\eta x$ so the clock is $\tau\gamma(\tau\tau\gamma)^\omega$.

(ii) For general terms with BT's that are branching, we have the same definition, but now the γ-steps also choose a direction in which to proceed. The clocks, or *clocked BT's* of the terms BY and BYS are as displayed in Figure 6. Here between the nodes of the tree, the number of necessary τ-steps have been indicated by the red numbers.

(iii) We are only interested in the tail of the clock reduction; the beginning does not count. More precisely: we consider clock sequences to be the same if they are *eventually concurrent*. Thus they may consist of different arbitrary long finite prefixes, followed by a common infinite tail. E.g. $\tau\gamma(\tau\tau\gamma)^\omega{=}(\gamma\tau\tau)^\omega$.

Proposition 3.3. *Clocks are accelerated under reduction, slowing down under expansion.*

Proof. By a straightforward parallel moves diagram construction. Put the clock reduction of the term M horizontally against a vertical reduction step $M \to M'$. If this is a τ-step (head reduction step), it will be absorbed by the upper reduction, and the projected reduction will soon coincide with the upper reduction. If it is a ι-step (non-head reduction step), it will propagate to the right as a complete development of ι-steps; some of these may absorb some of the horizontal τ-steps in the upper reduction, and this entails that the clock given by the lower projected reduction sequence is faster. The γ-steps in the upper reduction commute with both τ- and ι-steps. □

Definition 3.4. (*Simple terms*) A term M is simple, if in no reduction of M a redex is multiplied. So every redex $(\lambda x.A)B$ contracted in a reduct of M, has the property that x occurs at most once in A, or B is in normal form. An equivalent and useful reformulation is that in reduction diagrams involving reducts of M, *no splitting in elementary diagrams* occurs. Example: $Y_0\delta$ is not simple; it reduces to $\omega_\delta\omega_\delta$ and this term may duplicate the redex in the second ω_δ. But the reduct $\eta\eta \equiv Y_1$ is simple, and likewise all $\eta\eta\delta^{\sim n}$. This example illustrates that although sometimes the terms in consideration are not simple, with some luck they can be simplified by some reductions. Another example is $Y_1(SS)SI$ as in the Example above. Due to the presence of the redex (SS) this term is not simple. But it can easily be simplified, by reducing SS to its normal form $\underline{(SS)}$. (But there are also terms that have no simple reduct, i.e. cannot be simplified in this sense.)

Theorem 3.5. *For simple terms, clocks are invariant under reduction.*

Proof. (Sketch.) See Figure 5. The proof consists of an easy diagram chasing argument, just as in the proof of Proposition 3.3 about clock acceleration under reduction. The diagram consists of displaying M and its clock reduction vertically as in Figure 5; ι-steps are to the right. Now every reduct M' of M can be reached by descending a finite 'staircase' of the form, e.g., $\tau\tau\tau\iota\gamma\tau\iota\tau\iota\iota$. By repeated projection, starting from the initial vertical clock reduction of M, we obtain the vertically displayed clock reduction of M'. The point is now that in taking these projections, the ι-steps don't split. Now either they turn at some time into τ-steps and we are done, because then the projected sequence coincides henceforth with the previous one; or that never happens, e.g. because that redex is in a 'dark corner' of the term such as formed by a dummy parameter in a fpc Y^Ω, thus never coming up to the front; in that case the previous clock reduction is copied by the projection process in its entirety. □

Corollary 3.6. *Let M, N be simple terms with different clocks. Then $M \neq_\beta N$.*

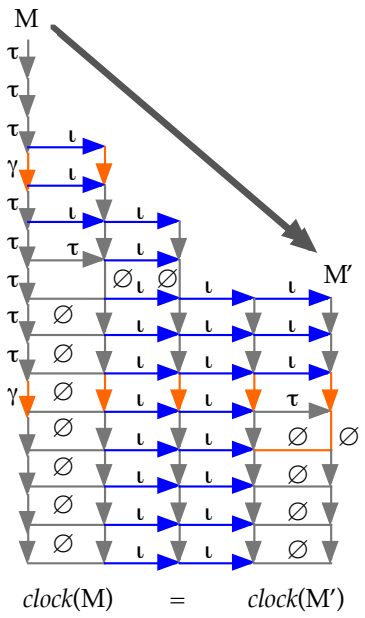

Figure 5: *Clock-invariance*

Proof. Let M, N be simple and have different clocks. Suppose $M =_\beta N$. Then these terms have a common reduct L. Now L must have two different clocks, which is impossible. Hence $M \neq_\beta N$. Note that if M, N are simple fpc's, then L is again so. □

Example 3.7. (1) The Böhm sequence of fpc's contains no duplicates. The proof consists first of noting that $\eta\eta\delta^{\sim n}$ is a simple term, and second computing the clock of this term. Above we did this already for $\eta\eta$; with every subsequent δ, two more head steps are introduced, so $clock(\eta\eta\delta^{\sim n}) = (\tau^{2n+2}\gamma)^\omega$, so all the clocks of fpc's in the Böhm sequence are different, so the sequence contains no duplicates.

(2) The Scott sequence of terms contains no duplicates, and likewise the sequence of fpc's derived from this sequence: first simplify the redex (SS) present, then the terms are simple, then compute their clocks, which turn out to be all different.

(3) The four double fixed point solutions for λI-calculus are different. Now we can state in what sense the first solution there is less efficient than the second: its clock is slower than that of the second.

(4) We also have at present an alternative proof that $BY \neq_\beta BYS$: the clock on the main branch of their BT, the only infinite branch, is different (Figure 6) for BY it is 022222..., while for BYS it is 404444... Also, these terms are simple.

Conjecture 3.8. *(For λI). Let $MP =_{BT} M$. Then MP has a slower clock than M. (In λK the clock can remain the same: $Y(KY) = Y$.)*

Problem 3.9. We mention some problems. (i) First, to prove that a term is simple may amount to just the sort of tediously establishing an invariant that we saw in the ad hoc proof for the Böhm sequence. So we would like to have techniques for proving terms simple. (ii) The speeding up of clocks under reduction, suggests for fpc's that they might be reduced

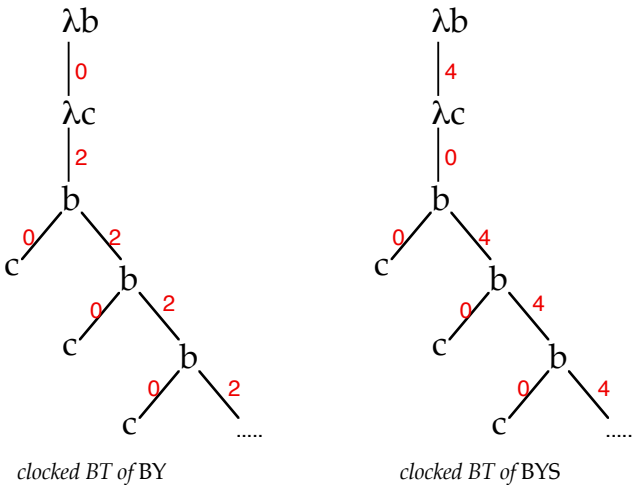

Figure 6: *BY and BYS.*

to a fpc with a fastest clock, i.e. minimal time between the γ's. This 'clock-minimalisation' would be very helpful; but we encounter a curious problem, namely that we do not know how to prove that the reduct of a fpc is again a fpc. For wfpc's it is easy to prove this 'subject-reduction' property.

Conjecture 3.10. *We conclude with a conjecture that gives a common perspective to all the non-equations that we have found or have endeavoured to find; including Intrigila's theorem. Namely,*

(1) that the space of fpc's is a free space, in the sense that no non-trivial equations hold. Furthermore,

(2) that fpc's enjoy a unique factorization property, in terms of a prime fpc as start, and building blocks as above as 'factors'. An fpc is called prime, if it is not the derivation (by postfixing a vector) of any other fpc. (Here we work in λI-calculus, otherwise the notion trivializes, by the equation $Y(KY') = Y'$.) Finally,

(3) we conjecture that the derivation relation is well-founded, i.e. there is no backward infinite sequence of derivations between fpc's.

Remark 3.11. As said, the notion of derivation is not interesting for λK-calculus without a further stipulation, which we will now make. That is, that we only are interested in derivation vectors that yield, after postfixing at a fpc, *ever new* fpc's. They must never loose their strength, as it were.

Acknowledgement. This paper has greatly benefitted from stimulating discussions with Henk Barendregt, so there is some 'reflection' in the dedication of this paper to him. He taught me lambda calculus and more. I also thank all members of the VU/CWI/UU-Infinity project, to wit, Jörg Endrullis, Clemens Grabmayer, Helle Hansen, Dimitri Hendriks, Ariya Isihara, Clemens Kupke, Vincent van Oostrom, Femke van Raamsdonk, Jan Rutten, Roel de Vrijer. Finally, thanks to Johannes Waldmann and Hans Zantema, for pointing out that δ-terms are not SN, contrary to my earlier conjecture.

References

[1] Barendregt, H.P. [1984], The Lambda Calculus, its Syntax and Semantics (North-Holland, Amsterdam, 1984).

[2] Barendregt, H.P. [1990], Functional Programming and Lambda Calculus, Chapter 7 in Handbook of Theoretical Computer Science, Volume B, Formal Models and Semantics, p.321-365, editor J. van Leeuwen. Elsevier 1990.Terese [2003], Term Rewriting Systems, Cambridge University Press.

[3] A. Berarducci, B. Intrigila. Church-Rosser-theories, infinite λ-terms and consistency problems. In W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, eds., Logic Colloquium '93. Oxford University Press, 1996.

[4] B. Intrigila [1997]. Non-existent Statman's Double Fixed Point Combinator Does Not Exist, Indeed. In: Information and Computation 137, 35-40, (1997).

[5] Coquand, T., and Herbelin, H.[1994], A-translation and Looping Combinators in Pure Type Systems. Journal of Functional Programming, 1994, Vol.4, Nr.1, p.77-88.

[6] M. Dezani-Ciancaglini, P. Severi, F.-J. de Vries [2003], Infinitary lambda calculus and discrimination of Berarducci trees, Theoretical Computer Science 298(2):275-302, 2003.

[7] Geuvers, H.; Werner, B., [1994], On the Church-Rosser property for expressive type systems and its consequences for their metatheoretic study. In: Proceedings Logic in Computer Science, 1994. LICS '94. 1994 p.320 - 329.

[8] J. R. Kennaway, J.W. Klop, M. R. Sleep, F.J. de Vries, [1997]. Infinitary lambda calculus, Theoret.Comput. Sci. 175, 93-125.

[9] J.R. Kennaway, F.-J. de Vries, Infinitary rewriting. Chapter 12 in Term rewriting systems, Terese [2003].

[10] J. R. Kennaway, J.W. Klop, M. R. Sleep, F.J. de Vries, Infinitary lambda calculus and Böhm models. In Proc. Conference on Rewriting Techniques and Applications, p.257-270, Springer LNCS 914, 1995.

[11] J. R. Kennaway, J.W. Klop, M. R. Sleep, F.J. de Vries, Transfinite reductions in Orthogonal Term rewriting Systems. Inf. and Comp. 119, Nr.1, 18-38 (1995)

[12] Klop, J.W. [1980] Combinatory Reduction Systems, Mathematical Centre Tracts 127, Mathematisch Centrum Amsterdam. Also available via http://web.mac.com/janwillemklop/iWeb/Site/Bibliography.html.

[13] J.W. Klop and R.C. de Vrijer [2005]. Infinitary Normalization. In: We Will Show Them: Essays in Honour of Dov Gabbay, volume 2, editors S. Artemov, H. Barringer, A.S. d'Avila Garcez, L.C. Lamb and J. Woods, pages 169-192, College Publications, 2005.

[14] [McCune1991] McCune, W., and Wos, L., "The Absence and the Presence of Fixed Point Combinators". Theoretical Computer Science 87:221-228, 1991.

[15] D.S. Scott [1975], Some philosophical issues concerning theories of combinators. In: λ-Calculus and Computer Science Theory. Proceedings of the Symposium Held in Rome, March 25-27, 1975, ed. C. Böhm, .346-366. Springer LNCS 37.

[16] Smullyan, R. [1985], To Mock a Mockingbird, And Other Logic Puzzles Including an Amazing Adventure in Combinatory Logic. Alfred A. Knopf, New York 1985.

[17] Terese [2003], Term Rewriting Systems, Cambridge University Press.

[18] J. Waldmann [1998]. Normalisation of S-Terms is Decidable, in: 9th Conf. Rewriting Techniques and Applications (RTA), Tsukuba 1998, ed. T. Nipkow, LNCS 1379, pp 138-150.

[19] F. Wiedijk [2005]. Lambda and CL calculator. Obtainable via http://web.mac.com/janwillemklop/iWeb/Site/.

[20] [Wos1993] Wos, L., "The Kernel Strategy and Its Use for the Study of Combinatory Logic". Journal of Automated Reasoning 10(3):287-343, 1993.