Limiting Partial Combinatory Algebras*

Yohji Akama

March 2, 2001

Abstract

We will construct from every partial combinatory algebra (PCA, for short) $\mathcal{A}$ a PCA a-lim($\mathcal{A}$) such that (1) every representable numeric function $\varphi(n)$ of a-lim($\mathcal{A}$) is exactly of the form $\lim_t \psi(t, n)$ with $\psi(t, n)$ being a representable numeric function of $\mathcal{A}$, and (2) $\mathcal{A}$ can be embedded into a-lim($\mathcal{A}$) which has a synchronous application operator. Here, a-lim($\mathcal{A}$) is $\mathcal{A}$ equipped with a limit structure in the sense that each element of a-lim($\mathcal{A}$) is the limit of a countable sequence of $\mathcal{A}$-elements. We will discuss various limit structures for $\mathcal{A}$ in terms of Barendregt's range property. Moreover, we will repeat the construction lim($-$) transfinite times to interpret infinite λ-calculi. Finally, we will attempt to interpret type-free $\lambda\mu$-calculus by introducing another partial applicative structure which has an asynchronous application operator and allows a parallel limit operation.

1 Introduction

A partial combinatory algebra (PCA, for short) is a partial algebra axiomatized by the same axioms as a combinatory algebra, except that application is only a partial operator.

The realizability interpretations extract the computational content from intuitionistic logic's proofs as programs. By carrying out the realizability interpretations abstractly over the PCA's, we can obtain 'realizability' models of typed term calculi and constructive set theories in which we can do mathematics. Cf. tripos construction(Hyland-Johnstone-Pitts[6]).

Recently, a new realizability interpretation was introduced by Nakata-Hayashi[8] to extract the computational content of semi-classical logic's proofs as approximation algorithms.

They first noticed that Gold's limiting recursive functions[5], which was originally introduced to formulate learning process of machines, serve as approximation algorithms. Then, they proved that some limiting recursive functions approximate a realizer of a semi-classical principle $\neg\neg\exists x \forall y.\, f(x, y) = 0 \to \exists x \forall y.\, f(x, y) = 0$. Also, they showed impressive usages of the semi-classical principle for mathematics and for software verification.

In this way, Nakata-Hayashi opened up the possibility that limiting operations provide realizability interpretation of semi-classical logical systems.

They formulated the set of the limiting recursive functions as a *Basic Recursive Function Theory*(BRFT, for short. Wagner[16] and Strong[13]), and carried out their realizability interpretation over the BRFT.

If we can formulate the set of limiting algorithms as a PCA $\mathcal{L}$, then by carrying out Nakata-Hayashi realizability interpretation abstractly over $\mathcal{L}$, we may be able to construct 'realizability' models of

1. semi-classical typed term calculi. For example, typed Parigot's $\lambda\mu$-calculus, typed term calculi with control operators.
2. semi-classical constructive set theory.

So, we will introduce a construction from a given PCA $\mathcal{A}$ another PCA a-lim($\mathcal{A}$). In a-lim($\mathcal{A}$), every allowed limit is exactly of the form $\lim_t a_t$ such that the infinite sequence $\langle\, a_t \in \mathcal{A}\, \rangle_t$

1. is indexed by $\mathbb{N}$; and

*The author acknowledges Susumu Hayashi and Mariko Yasugi for stimulating discussion on limiting computation, limiting realizability interpretation, and Gauss function. The author acknowledges Ken-etsu Fujita for stimulating discussion on $\lambda\mu$-calculi and for continual encouragement.

2. is generated inside the PCA $\mathcal{A}$.

We call this $\lim_t a_t$ an autonomous limit. Owing to the above (1,2), we will be able to prove that every representable function of a-lim($\mathcal{A}$) is exactly a limiting recursive function guessed by some representable functions of $\mathcal{A}$ (see Section 3).

So, we have only to take $\mathcal{L}$ as a-lim($\mathbb{N}$), in order to find realizability models of both semi-classical typed term calculi and a semi-classical constructive set theory.

In order to strengthen much more the computation power of a PCA $\mathcal{A}$, we will first consider the following two forms of limits $\lim_\lambda a_\lambda$.

(R) $\langle\, a_\lambda \in \mathcal{A}\,\rangle_\lambda$ is indexed by the whole $\mathcal{A}$.

(N) $\langle\, a_\lambda \in \mathcal{A}\,\rangle_\lambda$ is any countable sequence of $\mathcal{A}$-elements.

We will prove that (R) is unable to strengthen the computation power of $\mathcal{A}$. However, (N) strengthens extremely the computation power of $\mathcal{A}$. We will introduce another construction from any PCA $\mathcal{A}$ to another PCA n-lim($\mathcal{A}$) where only allowed limit is of the form (N). Then, the set of representable functions in n-lim($\mathcal{A}$) is the set of all partial numeric functions. Moreover, it can represent/compute Gauss function! See Section 4.

By using our results on limits over PCA's, we aim to interpret the following infinite λ-calculi

1. Tait's typed calculus of infinitely long terms[14]: An infinite sequence of type A terms is again a type A term.
2. S. Feferman's typed calculus of infinitely long terms (For details, see Schwichtenberg and Wainer [10]): An infinite sequence $\langle P_1, P_2, \ldots\rangle$ of type A terms is again a type A term. However, there should be a term that calculates for each i the code of P_i.
3. two infinite type-free λ-calculi of Kennaway-Klop-Sleep-de Vries[7] and Berarducci-Dezani[2]: Both have terms representing infinite Böhm-(Lévy-Longo-, or Berarducci-)trees.
4. The type-free $\lambda\mu$-calculus (Parigot [9]): The typed version corresponds to the classical logic, and a typed/type-free version relates to a typed/type-free functional programming language with control operators such as `call/cc`. It can be an infinite λ-calculus, by the following translation: μ-application $[\alpha]P$ and μ-abstraction $\mu\alpha.\, M$ will be respectively translated to is infinite applications $P'\alpha_0\alpha_1\ldots$ and $\lambda\alpha_0\alpha_1\ldots.\, M'$.

   Then, the μ-reduction will be a β-reduction.

The relationship between Tait's infinite typed λ-calculus and Feferman's typed λ-calculus is comparable to the relationship between n-lim($-$) and a-lim($-$).

The infinite λ-calculi (1,2,3) have an infinite term consisting of infinite terms. Therefore, we will repeat our constructions a-lim($-$) and/or n-lim($-$) in Section 5.

However, Parigot's type-free $\lambda\mu$-calculus is very difficult to interpret with a-lim and/or n-lim. We will introduce another construction n-LIM($-$) which extends a given PCA $\mathcal{A}$ to a partial applicative structure n-LIM($\mathcal{A}$) such that

- every allowed limit in n-LIM($\mathcal{A}$) is a parallel limit. So, we need not iterate the construction n-LIM($-$).
- the application operator is asynchronous.

This construction n-LIM($-$) was found by try-and-error. We will remark this structure by using the vocabulary of concurrency theory, and we will clarify the hidden parallelism of parallel limits. Then, we will interpret the type-free $\lambda\mu$-calculus in Section 6

2 Limiting Recursion

The set of partial functions from A to B is denoted by $A \rightharpoonup B$. We write $\overset{\infty}{\forall}\, x \in A.\, \varphi(x)$ to express that "$\{x \in A \mid \neg\varphi(x)\ \}$ is a finite set."

Definition 2.1 We say a partial numeric function $\varphi(n_1, \ldots, n_k)$ is *guessed* by a partial numeric function $\psi(t, n_1, \ldots, n_k)$ as t goes to infinity, if $\forall n_1, \ldots, n_k \exists t_0 \forall t > t_0.\ \varphi(n_1, \ldots, n_k) \simeq \psi(t, n_1, \ldots, n_k)$. We write $\varphi(n_1, \ldots, n_k) \simeq \lim_t \psi(t, n_1, \ldots, n_k)$.

Here, the symbol "$\simeq$" means "if one-side is defined, then the other side is defined as having the same value."

For every class $\mathcal{F}$ of partial numeric functions, $\lim(\mathcal{F})$ denotes the set of partial numeric functions guessed by a partial numeric function in $\mathcal{F}$.

In calibrating the computational power of $\lim(\mathcal{F})$, we recall the *limiting recursive functions* introduced by Gold[5]. We assume the knowledge about the arithmetical hierarchy of sets and complete sets with respect to many-one reducibility. The standard reference is Soare's book[11].

Fact 2.2 (Gold[5]) 1. A total function guessed by a partial recursive function can be guessed by a primitive recursive function.

2. A (partial) function guessed by a total recursive function is exactly a (partial) recursive function in the halting set $\mathcal{K}$ (called a *limiting recursive function* by Gold.)

We denote the set of partial recursive functions by **PRF**. $\lim(\mathbf{PRF}) \supseteq \mathbf{PRF}$, because every partial recursive function $\varphi(n_1, \ldots, n_k)$ is guessed by the partial recursive function $\varphi(\pi_2^{k+1}(t, n_1, \ldots, n_k), \ldots, \pi_{k+1}^{k+1}(t, n_1, \ldots, n_k))$, where each π_i^{k+1} returns the i-th argument. Below, we will show that $\lim(\mathbf{PRF})$ strictly includes the set of Gold's limiting recursive functions.

Fact 2.3 (Takeshi Yamazaki, or folklore) 1. For every total function, it is guessed by a total recursive function if and only if the graph of the function is in a class Δ_2^0 of the arithmetical hierarchy.

2. For every partial function, if it is guessed by a partial recursive function, then the graph and the domain are both in a class Σ_3^0 of the arithmetical hierarchy. Moreover, there is a partial function f such that f is guessed by a partial recursive function and the graph of f and the domain of f are both complete with respect to many-one reducibility.

A class of partial numeric functions is called a *basic recursive function theory*(BRFT, for short), if it has for each n the enumeration functions $\varphi_n(e, x_1, \ldots, x_n)$, the S_n^m-functions, the successor, the constant functions, the projections, and the equality, and is closed under the composition.

Theorem 2.4 (Nakata-Hayashi[8]) If $\mathcal{F}$ is a BRFT, then so is $\lim(\mathcal{F})$.

Proof. The enumeration functions of $\lim(\mathcal{F})$ actually enumerates the guessing functions.

3 Autonomous Limit

A *partial combinatory algebra(* PCA*, for short)* is $\mathcal{A} = \langle |\mathcal{A}|,\ \cdot,\ s,\ k\rangle$ such that $\cdot$ is a binary partial operator(*application operator*) on a set $|\mathcal{A}|$, and $s, k \in |A|$ are distinct elements subject to

$$(k \cdot a) \cdot b = a, \quad (s \cdot a) \cdot b \downarrow, \quad ((s \cdot a) \cdot b) \cdot c \simeq (a \cdot c) \cdot (b \cdot c).$$

Here, the symbol "$\downarrow$" means "is defined." And the symbol "$=$" means that "both sides are defined as having the same value." So, $t \downarrow$ is equivalent to an equation $t = t$. Examples of PCA are the set $\mathbb{N}$ of natural numbers and the set of values of the call-by-value λ-calculus. $\mathcal{A}, \mathcal{B}, \ldots$ range over PCA's. The application operator is often omitted if the context is clear.

Given a PCA, we can simulate a λ-abstraction; for a "polynomial" $t[x]$, there is a "polynomial" $\lambda x.\, t[x]$ such that $(\lambda x.\, t[x]) \cdot a \simeq t[a]$. $\lambda x.\, a = k \cdot a$, $\lambda x.\, x = s \cdot k \cdot k$. $\lambda x.\, t[x] \cdot t'[x] \simeq s \cdot (\lambda x.\, t[x]) \cdot (\lambda x.\, t'[x])$. The *Church numeral* of a natural number n, denoted by $\overline{n}$, is a polynomial $\lambda y \lambda x.\, y \cdot (\cdots (y \cdot x) \cdots)$ with the y successively applied n-times to x.

We say a partial numerical function $\psi(t, n_1, \ldots, n_k)$ is *represented* by an element $a \in \mathcal{A}$, whenever $\big(\cdots((a \cdot \overline{t}) \cdot \overline{n_1}) \cdots\big) \cdot \overline{n_k} = \overline{m}$ if and only if $\psi(t, n_1, \ldots, n_k) = m$. The set of partial numerical functions representable in $\mathcal{A}$ is denoted by $\mathrm{RpFn}(\mathcal{A})$. It is well-known that $\mathrm{RpFn}(\mathcal{A})$ contains **PRF** and is a BRFT.

Given a PCA $\mathcal{A}$, we will construct a PCA $\text{a-lim}(\mathcal{A})$ such that $\mathrm{RpFn}(\text{a-lim}(\mathcal{A})) = \lim(\mathrm{RpFn}(\mathcal{A}))$.

Definition 3.1 (Autonomous Limit of PCA) Given a PCA $\mathcal{A} = \langle |\mathcal{A}|,\ \cdot,\ s,\ k\rangle$. For $a, b \in \mathcal{A}$, define

$$a \sim b \iff \overset{\infty}{\forall}\, t \in \mathbb{N}.\, (a \cdot \overline{t} \simeq b \cdot \overline{t}). \qquad (1)$$

The extension $\text{a-lim}(\mathcal{A})$ *of a* PCA $\mathcal{A}$ *by the autonomous limits* is a PCA $\text{a-lim}(\mathcal{A}) = \langle |\text{a-lim}(\mathcal{A})|, *, \mathbf{s}, \mathbf{k}\rangle$, where

- $|\text{a-lim}(\mathcal{A})|$ is a quotient set $\{a \in |\mathcal{A}| \mid \overset{\infty}{\forall} t.\, a \cdot \overline{t} \downarrow\}/\sim$;
- $\mathbf{s} = [k \cdot s]_\sim$ ("s for any value t", t being "time");
- $\mathbf{k} = [k \cdot k]_\sim$ ("k for any value t", t being "time"); and

- $[a]_\sim * [b]_\sim = [(s\cdot a)\cdot b]_\sim$ ("synchronous application").

As $\sim$ is an equivalence relation on $|\mathcal{A}|$, the quotient set $|\text{a-lim}(\mathcal{A})|$ is well-defined. The element is an equivalence class $[a]_\sim$ with $a \in \mathcal{A}$. When a is undefined, so will $[a]_\sim$ be undefined.

The operation $*$ is well-defined; Suppose $a \sim a'$ and $b \sim b'$ and $s\cdot a\cdot b\cdot \overline{n} \simeq (a\cdot\overline{n})\cdot(b\cdot\overline{n})$ is defined for values of n which are large enough. Then, because $a\cdot\overline{n} \simeq a'\cdot\overline{n}$ and $b\cdot\overline{n}\simeq b'\cdot\overline{n}$, we have $s\cdot a\cdot b\cdot\overline{n} \simeq (a\cdot\overline{n})\cdot(b\cdot\overline{n}) \simeq (a'\cdot\overline{n})\cdot(b'\cdot\overline{n}) \simeq s\cdot a'\cdot b'\cdot\overline{n}$. Therefore, $[a]*[b] \simeq [a']*[b']$.

Theorem 3.2 a-lim$(\mathcal{A})$ is a PCA such that RpFn(a-lim$(\mathcal{A})$) $=$ lim(RpFn$(\mathcal{A})$).

Proof. Let $t \in \mathbb{N}$ be sufficiently large.

- $\mathbf{s}*[a]*[b]*[c] \simeq [s(s(s(ks)a)b)c]$ while $[a]*[c]*([b]*[c]) \simeq [s(sac)(sbc)]$. An axiom for s implies $s(s(s(ks)a)b)c\overline{t} \simeq (ks\overline{t})(a\overline{t})(b\overline{t})(c\overline{t})$. The last is, by using the axiom for k with $\overline{t}\downarrow$, $\simeq s(a\overline{t})(b\overline{t})(c\overline{t})$. It is, by an axiom for s, $\simeq s(sac)(sbc)\overline{t}$. So, $\mathbf{s}*[a]*[b]*[c] \simeq [a]*[c]*([b]*[c])$.

- $\mathbf{s}*[a]*[b] \simeq [s(s(ks)a)b]$. An axiom of s implies $s(s(ks)a)b\overline{t} \simeq ks\overline{t}(a\overline{t})(b\overline{t})$. It is, by using the axiom of k with $\overline{t}\downarrow$, $\simeq s(a\overline{t})(b\overline{t})$. It is always defined because $a\overline{t}\downarrow$, $b\overline{t}\downarrow$ and an axiom of s. So, $\mathbf{s}*[a]*[b]\downarrow$.

- $\mathbf{k}*[a]*[b] = [s(s(kk)a)b]$ and $s(s(kk)a)b\overline{t} \simeq (kk\overline{t})(a\overline{t})(b\overline{t})$. It is, by using the axiom of k with $\overline{t}\downarrow, \simeq k(a\overline{t})(b\overline{t})$. It is $\simeq a\overline{t}$ as $b\overline{t}\downarrow$. So, $\mathbf{k}*[a]*[b] = [a]$.

Therefore, a-lim$(\mathcal{A})$ is indeed a PCA.

Before proving the latter part of the theorem, we note that a Church numeral $\overline{\mathbf{n}}$ of a-lim$(\mathcal{A})$ is $[k\overline{n}]_\sim$ with the latter Church numeral $\overline{n}$ being in $\mathcal{A}$.

Both of RpFn(a-lim$(\mathcal{A})$) and lim(RpFn$(\mathcal{A})$) have the nowhere defined partial functions of any arity. Let φ be a unary partial function somewhere defined. Then, $\varphi \in$ RpFn(a-lim$(\mathcal{A})$) $\iff$

$$\begin{aligned}
&\exists a\in\mathcal{A}\Big(\overset{\infty}{\forall}\, t.\, a\overline{t}\downarrow \;\;\&\\
&\forall n,m\ (\ [a]_\sim * \overline{\mathbf{n}} = \overline{\mathbf{m}} \text{ iff } \varphi(n)=m\ )\Big)\\
&\iff\\
&\exists a\in\mathcal{A}\Big(\overset{\infty}{\forall}\, t.\, a\overline{t}\downarrow \;\;\&\\
&\forall n,m\\
&\Big(\overset{\infty}{\forall}\, t\in\mathbb{N}.\, sa(k\overline{n})\overline{t} = (k\overline{m})\overline{t}\Big) \text{ iff } \varphi(n)=m\Big)\\
&\iff\\
&\exists a\in\mathcal{A}\Big(\overset{\infty}{\forall}\, t.\, a\overline{t}\downarrow \;\;\&\\
&\forall n,m\Big(\overset{\infty}{\forall}\, t\in\mathbb{N}.\, a\overline{t}\overline{n} = \overline{m}\Big) \text{ iff } \varphi(n)=m\ \Big)
\end{aligned}$$

We will refer above formula by (¶). On the other hand, $\varphi \in$ lim(RpFn$(\mathcal{A})$) is equivalent to

$$\exists\psi\in\mathrm{RpFn}(\mathcal{A})$$
$$\forall n,m\Big(\lim_t \psi(t,n) = m \text{ iff } \varphi(n)=m\Big) \qquad (2)$$

As φ is somewhere defined, we have $\varphi(l)\downarrow$ for some l. Therefore, $\overset{\infty}{\forall}\, t.\, \psi(t,l)\downarrow$. So, every $a \in \mathcal{A}$ representing ψ satisfies $\overset{\infty}{\forall}\, t.\, a\overline{t}\downarrow$. Thus, (¶) if and only if (2). Therefore, $\varphi \in$ RpFn(a-lim$(\mathcal{A})$) if and only if $\varphi \in$ lim(RpFn$(\mathcal{A})$). For φ with the arity being greater than 1, it is similarly proved.

Definition 3.3 (homomorphism) A function f from a PCA $\mathcal{A}$ to a PCA $\mathcal{B}$ is a *homomorphism*, if f preserves the operators as relations. That is,

- $f(s) = \mathbf{s}$, $f(k) = \mathbf{k}$, and
- $ab = c$ implies $f(a)f(b) = f(c)$.

We denote by $\mathbb{PCA}$ the category of PCA's and homomorphisms between them. An injective, surjective homomorphism is called an *isomorphism*. We abbreviate a homomorphism by homo.

Theorem 3.4 Define the function $\iota_\mathcal{A}$ by

$$\iota_\mathcal{A} : \mathcal{A} \to \text{a-lim}(\mathcal{A});\ a \ \mapsto\ [k\cdot a]_\sim\ .$$

Then,

1. $\iota_\mathcal{A}$ is an injective, non-surjective homo.
2. $[a]_\sim = \iota_\mathcal{A}(b) \iff \overset{\infty}{\forall}\, t\in\mathbb{N}.\ a\cdot\overline{t} = b$ $(\iff$ "$\lim_t\ a\cdot\overline{t} = b$")
3. $a\cdot a' = b \iff \iota_\mathcal{A}(a) * \iota_\mathcal{A}(a') = \iota_\mathcal{A}(b)$.

Proof. (1) $\iota_{\mathcal{A}}$ is clearly an injective homo. To see that it is not surjective, assume that $[skk]_\sim = \iota_{\mathcal{A}}(b)$. Then, $\overset{\infty}{\forall}\, t.\, skk\overline{t} \simeq kb\overline{t}$, which is $\overset{\infty}{\forall}\, t.\, \overline{t} \simeq b$. Contradiction. (2) and (3) are trivial.

4 Various Limitations

Each element of the PCA a-lim($\mathcal{A}$) is of the form $\lim_t a_t$, where the

1. parameter t can be any natural number; and
2. the sequence $\langle a_t\rangle_t$ is of the form $\langle a\cdot\overline{t}\rangle_t$ for some $a\in\mathcal{A}$. In this case, the sequence is "autonomously tracked" by an $\mathcal{A}$-element a.

To justify the necessity of the two conditions, we will discuss following alternative $\lim_t a_t$ for $\mathcal{A}$:

1. t of $\lim_t a_t$ can be any element of $\mathcal{A}$; or
2. the sequence $\langle a_t\rangle_t$ is any countable sequence.

Range property and limit. We consider the following semantics of lim operation: $\lim_a f(\vec{x},a)\simeq g(\vec{x}) \iff$

$$\overset{\infty}{\forall}\, a\in|\mathcal{A}|.\ g(\vec{x})\simeq f(\vec{x},a). \tag{3}$$

Although this limit is meaningful in a PCA $\mathbb{N}$, it is not the case for the set $\mathcal{M}$ of closed λ-terms modulo β-equality. $\mathcal{M}$ is a model of $\lambda\beta$-calculus. Even though we can construct from $\mathcal{M}$ another PCA $A(\mathcal{M})$ with the limit being (3), we have $\mathbf{PRF} = \mathrm{RpFn}(A(\mathcal{M})) = \mathrm{RpFn}(\mathcal{M}) \neq \lim(\mathrm{RpFn}(\mathcal{M})) = \lim(\mathbf{PRF})$. This is because of the range property studied by Barendregt([1, p.517]): In the closed term model of $\lambda\beta$-calculus, the range of any combinator is either a singleton or an infinite set.

Indeed, for any definable function $f:\mathcal{M}\times\mathcal{M}\to\mathcal{M}$, so is the function $F_x(a)=f(x,a)$. When $\lim_a f(x,a)$ in the above sense has a value, F_x has the finite range $\{F_x(a)\mid a\in\mathcal{M}\}$. However, it must be a singleton because of the range property. Therefore, the lim of (3) will be useless.

Limit along all the countable sequences. Given a partial applicative structure $\mathcal{M} = \langle|\mathcal{M}|,\ \cdot\rangle$. Let n-lim($\mathcal{M}$) $=\langle|\text{n-lim}(\mathcal{M})|,*\rangle$ such that (1) $|\text{n-lim}(\mathcal{M})| = (\mathbb{N}\rightharpoonup|\mathcal{M}|)\,/\,\sim$, where $f\sim g$ means $\overset{\infty}{\forall}\, t\in\mathbb{N}.\, f(t)\simeq g(t)$; and (2) $*$ is the synchronous application, that is, $[f]_\sim * [g]_\sim = [h]_\sim$ where $h(t)\simeq f(t)\cdot g(t)$.

We do not have $\mathbf{k}\in$ n-lim($\mathcal{M}$) such that $\mathbf{k}*[g]_\sim*[h]_\sim = [g]_\sim$. This can be clearly seen when $f:\mathbb{N}\rightharpoonup|\mathcal{M}|$ is nowhere defined. Then, $x*[f]_\sim = [f]_\sim$. Moreover, $\mathbf{k}*[g]_\sim*[h]_\sim=[f]_\sim$ when t is an even natural number iff $g(t)\downarrow$ iff not $h(t)\downarrow$. These two anomalies are because p of $[p]_\sim\in$ n-lim($\mathcal{M}$) is not so 'densely' defined.

Theorem 4.1 Let $\mathcal{M}$ be a partial algebra of a signature $\langle f_1,\ldots,;\ u_1=u_1',\ldots;v_1\simeq v_1',\ldots\rangle$. Here $f_1,\ldots$ are (partial) operators, and $u_1=u_1',\ldots;v_1\simeq v_1',\ldots$ are axioms. Each u_i,u_i',v_i,v_i' is built up from the variables and the operators $f_1,\ldots$.

The extension n-lim($\mathcal{M}$) *of* $\mathcal{M}$ *by the non-constructive limits* is $\langle|\text{n-lim}(\mathcal{M})|,\ \mathbf{f},\ldots\rangle$ such that

1. $|\text{n-lim}(\mathcal{M})| = \{a:\mathbb{N}\rightharpoonup\mathcal{M}\mid \overset{\infty}{\forall}\, t.\, a(t)\downarrow\}/\sim$, where
$$a\sim b \iff \overset{\infty}{\forall}\, k\in\mathbb{N}.\, a(k)\simeq b(k). \tag{4}$$
2. For $\mathbf{c}^{(k)} = [\ c^{(k)}\ ]_\sim \in |\text{n-lim}(\mathcal{M})|$, define $\mathbf{f}(\mathbf{c}^{(1)},\ \mathbf{c}^{(1)},\ldots)\simeq[\ \varphi\ ]_\sim$ with $\varphi(i)\simeq f(\ c^{(1)}(i),\ c^{(2)}(i),\ldots)$.

Then n-lim($\mathcal{M}$) is a partial algebra of the same signature.

A homomorphism is a function preserves the operators as relations.

Theorem 4.2 Define the function

$$\iota_{\mathcal{M}}:\mathcal{M}\to\text{n-lim}(\mathcal{M});\ b\ \mapsto\ [a]_\sim,\quad \text{where } a(t)=b.$$

Then,

1. $\iota_{\mathcal{M}}$ is an injective homo. Moreover, it is non-surjective if $\mathcal{M}$ has at least two elements.
2. $[a]_\sim = \iota_{\mathcal{M}}(b) \iff (\overset{\infty}{\forall}\ t\in\mathbb{N}.\ a(t)=b)\quad (\iff \text{"}\lim_t\ a(t)=b\text{"})$
3. $\mathbf{f}(b,\ldots)=b'$ in $\mathcal{M}$ $\iff f(\iota_{\mathcal{M}}(b),\ldots)=\iota_{\mathcal{M}}(b')$ in n-lim($\mathcal{M}$).

Proof. Suppose $\iota_{\mathcal{M}}$ is surjective. Then, let $a:\mathbb{N}\rightharpoonup\mathcal{M}$ such that $a(t)=b_{t \bmod 2}$ with distinct $b_0,b_1\in\mathcal{M}$. We have $\overset{\infty}{\forall}\ t\in\mathbb{N}\,(a(t)=b)$. Contradiction. The other claims are clear.

Theorem 4.3 If $\mathcal{A}$ is a PCA, so is n-lim($\mathcal{A}$) such that RpFn(n-lim($\mathcal{A}$)) is the set of all partial numeric functions.

Proof. The first part follows from the previous Theorem. We will prove only the second part. Since any partial numeric function of finite domain is representable in any PCA, every partial numeric m-ary function φ is represented by $[f]_\sim$ such that $f(t)$ represents $\varphi \restriction \{0, 1, \ldots, t-1, t\}^m$ in $\mathcal{A}$.

Remark 4.4 There are other PCA's where every partial numeric function is representable. For instance, D_∞ introduced by Scott and the partial continuous operations (PCO, for short. See [15, Ch.9, Sect.4]) introduced by Kleene. However, no n-lim($\mathcal{A}$) is isomorphic to them. This fact is explained by comparing the three extensional collapses[15] of n-lim($\mathcal{A}$), D_∞ and PCO.

On the one hand, the extensional collapse of n-lim($\mathcal{A}$) contains a discontinuous functional over $\mathbb{N}$. One such example is the Gauss function, a function given a real number x returning the smallest integer not greater than x. The Gauss function is actually a functional over $\mathbb{N}$; every real number may be regarded as a converging sequence of rational numbers, and thus as a function from $\mathbb{N}$ to $\mathbb{Q}$. Because $\mathbb{Q}$ is encoded to $\mathbb{N}$, every real number is a function from $\mathbb{N}$ to $\mathbb{N}$. The Gauss function is not Kleene-Kreisel continuous, because the output needs infinite precision on the input real number to determine whether the input is an integer or not. As Gauss function is rewritten as the limit according to Yasugi-Brattka[17], it is in the extensional collapse of n-lim($\mathcal{A}$).

On the other hand, the extensional collapses of both D_∞ and PCO are the same; a type structure consisting of all total Kleene-Kreisel continuous functionals over $\mathbb{N}$. By Kleene-Kreisel continuous functionals, we mean that the output of the functional depends only on finite subinformation of the input functionals.

So, n-lim($\mathcal{A}$) is isomorphic to neither D_∞ nor PCO.

Remark 4.5 An element of n-lim($\mathcal{A}$) is an equivalence class of sequences of $\mathcal{A}$-elements, while an element of a-lim($\mathcal{A}$) is an equivalence class of "autonomously tracked" sequences of $\mathcal{A}$-elements. This kind of *autonomous enumeration* also appears in defining by infinite terms a large class of calculable functionals of finite types.

In [14], Tait introduced a typed λ-calculus of infinitely long terms, which has the following formation rule of infinite terms:

> if $\{a_t\}_{t\in\mathbb{N}}$ is an enumeration of type A terms, then $\langle a_t\rangle_{t\in\mathbb{N}}$ is a type A term.

After Solomon Feferman developed the system in a proof-theoretic context, he introduced in unpublished notes another typed λ-calculus T_0, where the above formation rule is applicable to only *autonomous enumerations*:

> if for each $t \in \mathbb{N}$, $f(t)$ codes a type A term a_t and f is itself defined by a term, then $\langle a_t\rangle_{t\in\mathbb{N}}$ is a type A term.

He analyzed T_0 in a recursion-theoretic context (For details for higher-type recursion-theoretic expansion, see Schwichtenberg and Wainer [10]).

The relationship between Tait's typed λ-calculus and Feferman's typed λ-calculus T_0 is comparable to the relationship between n-lim($\mathcal{A}$) and a-lim($\mathcal{A}$).

5 Repeating Limits

In functional programming, we use infinite lists as streams(for input/output) and non-terminating recursive calls. These objects are usually the unwinding("limit") of finite objects. To analyze such infinite objects, of interest are *infinite* λ-calculi which were introduced by Kennaway-Klop-Sleep-de Vries[7] and Berarducci-Dezani[2]. Both calculi admit a term like $\lambda x.y^\omega x$, $\lambda x.\, y^\omega(x^\omega)$, and have terms with the limit operation $(-)^\omega$ being nested.

Interpretations of these infinite λ-calculi may be constructed by our limit constructions(Definition 3.1 and Theorem 4.1) of PCA's. In such cases, it is necessary to repeat the limit constructions ω-times.

Definition 5.1 (α-times repeated limits) For every PCA $\mathcal{A}$ and every ordinal number α, let us define a PCA $\lim^\alpha(\mathcal{A})$ and the canonical injective homo. $\iota^\alpha_\beta : \lim^\beta(\mathcal{A}) \to \lim^\alpha(\mathcal{A})$ for each $0 < \beta \le \alpha$.

- a-$\lim^0(\mathcal{A}) = \mathcal{A}$, and ι^β_β is the identity for each ordinal number β.
- a-$\lim^{\beta+1}(\mathcal{A}) = $ a-lim $\big($a-$\lim^\beta(\mathcal{A})\big)$, and $\iota^{\beta+1}_\gamma = \iota_{\text{a-}\lim^\beta(\mathcal{A})} \circ \iota^\beta_\gamma$.
- When α is a limit ordinal number, a-$\lim^\alpha(\mathcal{A})$ is an inductive limit of $\langle \iota^\delta_\gamma \rangle_{0<\gamma<\delta<\alpha}$, and each ι^α_δ is a natural injection of the inductive limit.

We similarly define n-$\lim^{\alpha}(\mathcal{A})$.

Theorem 5.2 For each ordinal number $\alpha > 0$,

1. $\lim^{\alpha}(\mathcal{A})$ is indeed a PCA, and ι^{α}_{β} is an injective, non-surjective homo. for $\beta < \alpha$; and
2. RpFn(a-$\lim^{\alpha}(\mathcal{A})$) is a-$\lim^{\alpha}$(RpFn($\mathcal{A}$)) for α finite, and is $\bigcup_{\beta\in\mathbb{N}} \lim^{\beta}(\mathrm{RpFn}(\mathcal{A}))$ otherwise.

Proof. The first part will proved by transfinite induction on α. In proving the second part of the theorem, note that RpFn(a-$\lim^{\alpha}(\mathcal{A})$) is $\bigcup_{\beta<\alpha}$ RpFn(a-$\lim^{\beta}(\mathcal{A})$). In view of Theorem 3.2, it is OK for finite α. For $\alpha = \omega + 1$, RpFn(a-$\lim^{\omega+1}(\mathcal{A})$) is $\lim\left(\bigcup_{\beta<\omega} \mathrm{RpFn}(\text{a-}\lim^{\beta}(\mathcal{A}))\right)$, which is $\lim\left(\bigcup_{\beta<\mathbf{N}} \text{a-}\lim^{\beta}(\mathrm{RpFn}(\mathcal{A}))\right)$. So, $\varphi(n) \in \mathrm{RpFn}(\lim^{\omega+1}(\mathcal{A}))$ iff $\exists t'_0 \forall t_0 > t'_0\, \varphi(n) \simeq \xi(t_0, n)$ for some $\xi \in \lim^{\beta}(\mathrm{RpFn}(\mathcal{A}))$ with $\beta < \omega$. Hence, $\varphi(n) \in \mathrm{RpFn}(\lim^{\omega+1}(\mathcal{A}))$ iff $\exists t'_0 \forall t_0 > t'_0 \cdots \exists t'_\beta \forall t_\beta > t'_\beta\, \varphi(n) \simeq \xi'(t_0, \cdots, t_\beta, n)$ for some $\xi' \in \mathrm{RpFn}(\mathcal{A})$ with $\beta < \omega$. Thus, $\mathrm{RpFn}(\lim^{\omega+1}(\mathcal{A}))$ is $\mathrm{RpFn}(\lim^{\omega}(\mathcal{A}))$. In this way, we can prove the second part of the theorem.

Theorem similar to Theorem 3.4 also holds

We do not know whether there is an ordinal number α such that $\lim^{\alpha}(\mathcal{A})$ is isomorphic to $\lim(\lim^{\alpha}(\mathcal{A}))$. $\iota^{\alpha+1}_{\alpha}$ cannot be such an isomorphism, according to above Theorem.

Although the construction lim is an endofunctor of a category $\mathbb{PCA}$, it is difficult to employ a category-theoretic version of Tarski's fixpoint theorem arguments; because two homo's $\lim(\iota^{\beta+1}_{\beta})$ and $\iota^{\beta+2}_{\beta+1}$ are not equal but their equalizer can be proved to be $\iota^{\beta+1}_{\beta}$.

Theorem 5.3 For each ordinal number $\alpha > 0$,

1. n-$\lim^{\alpha}(\mathcal{A})$ is indeed a PCA, and ι^{α}_{β} is an injective, non-surjective homo. for $\beta < \alpha$; and
2. RpFn(a-$\lim^{\alpha}(\mathcal{A})$) is the set of partial numeric functions.

An infinitely long term $\langle P_1, P_2, \ldots\rangle$ of Tait's or Feferman's λ-calculi can be interpreted with a-$\lim^{\alpha+1}(\mathcal{A})$ as follows:

$$\langle a_1, a_2, a_3, \ldots, \rangle \simeq [t \mapsto \langle a_1, a_2, a_3, \ldots, a_t\rangle_t]_{\sim},$$

where $a_i \in$ a-$\lim^{\alpha}(\mathcal{A})$ is an interpretation of the term P_i, and the ordinal number α depends on the complexity of P_i's. Here, $\langle a_1, a_2, a_3, \ldots, a_t\rangle = \mathbf{cons}\ a_1\ (\mathbf{cons}\ a_2\ (\cdots(\mathbf{cons}\ a_t\ \mathbf{i})\cdots))$, for a pairing $\mathbf{car}\ (\mathbf{cons}\ a\ b) = a$, and $\mathbf{cdr}\ (\mathbf{cons}\ a\ b) = b$. For example, we can define

$$\mathbf{cons} = \lambda xyz.\, xyz, \qquad \mathbf{car} = \lambda x.\, x\mathbf{k}, \quad \mathbf{cdr} = \lambda x.\, x(\mathbf{ki}). \tag{5}$$

6 An Interpretation of Type-free $\lambda\mu$-calculus

In [9], Parigot introduced the typed $\lambda\mu$-calculus which corresponds to classical propositional logic via Curry-Howard isomorphism. By forgetting the types in the $\lambda\mu$-calculus, we obtain a *type-free $\lambda\mu$-calculus.* Both calculi are related to type-free/typed programming languages with control operators (call/cc, $\mathcal{C}$ introduced by M. Felleisen and D. Friedman in [3], raise-handle, ...).

Type-free $\lambda\mu$-calculus is specified by defining $\lambda\mu$-terms, and the reduction rules (the β-reduction rule and the μ-reduction rule).

$\lambda\mu$-Terms are generated by

$$\begin{array}{lllll} M & ::= c & \mid x & \mid MM & \mid \lambda x.\, M \\ & & \mid [\alpha]M & & \mu\texttt{-application} \\ & & \mid \mu\alpha.\, M. & & \mu\texttt{-abstraction} \end{array}$$

An occurrence of α in $\mu\alpha. \cdots$ will be called a *bound* occurrence of α. An occurrence of α which is not bound is called *free.* $[\alpha]M$ is regarded as application of M to the continuation bound to α. $\mu\alpha.\, M$ is regarded as functional abstraction over the continuation variable α on the level of continuation semantics.

Here, $x, y, z, \ldots$ range over term-variables. $\alpha, \beta, \gamma, \ldots$ range over *μ-variables* which are distinguished from the term-variables.

Mixed substitution. A *context* is generated by the above grammar with a special constant ( ). By replacing the occurrences of ( ) of a context $C(\ )$ with a $\lambda\mu$-term M, we obtain a $\lambda\mu$-term $C(M)$.

For any context $C(\ )$ and a $\lambda\mu$-term N, we define a *mixed substitution* $\vartheta = \Big[[\alpha](\ ) := C(\ )\Big]$ as follows; If N is a $\lambda\mu$-term, so is a $N\vartheta$. If N does not have a free μ-variable α, then $N\vartheta$ is N. So, if N is a variable or $\mu\alpha.\,M$, then $N\vartheta \equiv N$. The mixed substitution commutes with a λ-abstraction and an application. The mixed substitution $\vartheta = \Big[[\alpha](\ ) := C(\ )\Big]$ satisfies $([\beta]M)\vartheta \equiv [\beta](M\vartheta)$ and $(\mu\beta.\,M)\vartheta \equiv \mu\beta.\,(M\vartheta)$, provided $\beta \not\equiv \alpha$. Finally,

$$([\alpha]M)\Big[[\alpha](\ ) := C(\ )\Big] \equiv C\Big(M\Big[[\alpha](\ ) := C(\ )\Big]\Big)$$

μ-reduction is specified by the following rule:

$$(\mu\alpha.\,M)N \to_\mu \mu\alpha.\Big(M\Big[[\alpha](\ ) := [\alpha]((\ )N)\Big]\Big).$$

In graphical notation, the rule is

$$\begin{aligned} &(\mu\alpha.(\cdots([\alpha]P)\cdots))N \\ \to_\mu\ &\mu\alpha.(\cdots([\alpha](P'N))\cdots). \end{aligned} \tag{6}$$

For instance, $\Big(\mu\alpha.\,[\alpha]\,(y[\alpha]x)\Big)\,z \to_\mu \mu\alpha.\,[\alpha]\,(y\,([\alpha](xz))\,z)$. Of course, the type-free $\lambda\mu$-calculus has the usual β-reduction.

6.1 Informal Semantics of $\lambda\mu$

μ-application(abstraction,reduction) = infinite application(abstraction,β-reduction). Consider an informal translation from the type-free $\lambda\mu$-calculus to (infinitely long) type-free λ-calculus:

$$\begin{cases} [\alpha]P & \mapsto\quad P'\alpha_0\ldots\alpha_m\ldots \\ \mu\alpha.\,M & \mapsto\quad \lambda\alpha_0\ldots\alpha_m\ldots.\,M' \end{cases} \tag{7}$$

Then, the above rewriting rule (6) is translated to

$$\begin{aligned} &(\lambda\alpha_0\alpha_1\ldots(\cdots(P\alpha_0\alpha_1\ldots)\cdots))N \\ \to_\beta\ &\lambda\alpha_0\alpha_1\ldots(\cdots(P'N\alpha_0\alpha_1\ldots)\cdots), \end{aligned} \tag{8}$$

which turns out to be a β-reduction between infinite terms, if we can rename bound variables infinite times

$$\begin{aligned} &\lambda\alpha_1\ldots.\ (\cdots(P'N\alpha_1\ldots.)\cdots) \\ \equiv\ &\lambda\alpha_0\alpha_1\ldots(\cdots(P'N\alpha_0\alpha_1\ldots)\cdots). \end{aligned}$$

This idea of Parigot is being studied by Fujita[4]; with type-regime, the translation above precisely corresponds to Gödel translation and the length of $\alpha_0, \alpha_1, \ldots$ is finite.

The η-like reduction on continuation

$$\mu\alpha.\,[\alpha]P \to P \qquad \text{if } \alpha \text{ is not free in } P.$$

turns out to be just *infinite* η-reductions(see Theorem 6.15).

μ-variables = infinite streams. The idea of (7) leads us to interpret

$$[\alpha]P \simeq \lim_t P\alpha_0\alpha_1\ldots\alpha_t \simeq [t \mapsto P\alpha_0\alpha_1\ldots\alpha_t]_\sim$$

$$\alpha_i \simeq \mathbf{car} * \overbrace{(\mathbf{cdr} * \cdots (\mathbf{cdr}}^{i} * \alpha)\cdots)$$

Then, we have a *Swap rule* of Streicher-Reus's version of type-free $\lambda\mu$-calculus[12]:

$$[\mathbf{cons} *\ M\ * N]P \simeq [N](P * M),$$

if we allow more general *continuation terms* than mere μ-(continuation) variables namely pure λ-terms stacked to a μ-variable $\mathbf{cons} * M_1 * (cons * M_2 * \ldots * (\mathbf{cons} * M_n * \alpha)\ldots)$.

Before we consider the interpretation of μ-abstraction, we will make (8) rigorous.

μ-reduction causes delay in a stream. The translation result of the rewriting rule (6) is $f \to g$ such that

$$\begin{aligned} f(t) &\simeq (\lambda a_0\ldots a_t.\,(\cdots(Pa_0\ldots a_t)\cdots))N \\ &\simeq \lambda a_1\ldots a_t.\,(\cdots(PNa_2\ldots a_t)\cdots), \end{aligned}$$

$$g(t) \quad \simeq \lambda a_0\ldots a_t.(\cdots(PNa_0\ldots a_t)\cdots)$$

But,

$$f(t+1) \simeq g(t).$$

Because it takes 1 'clock time' to compute(by β-reduction) g from f, a delay will occur because of the extra computational time required. Anyway, $[f]_\sim \neq [g]_\sim$ in n-lim($\mathcal{A}$). So, n-lim($\mathcal{A}$) does not interpret the calculus. Neither does a-lim($\mathcal{A}$).

To equate f and g above, let us replace the equivalence relation $\sim$ with the smallest equivalence relation $\approx$ containing $\sim$ and the 'delay' rule $f \approx (t \mapsto f(t+1))$. Unfortunately, a quotient set $(\mathbb{N} \rightharpoonup |\mathcal{A}|)\ /\ \approx$ cannot have the synchronous application operator $[f]_\approx * [g]_\approx \simeq [t \mapsto f(t)\cdot g(t)]_\approx$ well-defined.

6.2 Asynchronous Applicative Structure and Parallel Limit

Given a PCA $\mathcal{A}$, we introduce another partial algebra n-LIM($\mathcal{A}$) where an appropriate application operator can be defined.

An element of n-LIM($\mathcal{A}$) is an equivalence class $[f]_\sim$ where $f : \mathbb{N}^n \rightharpoonup \mathcal{A}$ such that $n \geq 0$ and $\{\vec{x} \in \mathbb{N}^n \mid f(\vec{x}) \text{ is undefined }\}$ is finite. Here, $\sim$ is an equivalence relation over $\bigcup_{n\geq 0}(\mathbb{N}^n \rightharpoonup |\mathcal{A}|)$ defined by the usual equational rules(reflexivity, symmetricity, and transitivity) plus the following two rules.

1. The 'delay' rule.

   $$\mathbb{N}^n \xrightarrow{f} \mathcal{A} \sim \mathbb{N}^n \xrightarrow{id\times\cdots\times id\times suc\times id\cdots\times id} \mathbb{N}^n \xrightarrow{f} \mathcal{A}$$

   where id is the identity function on $\mathbb{N}$ and suc is the successor function.

   As for $\times$, it is an associative operator and for all $f_i : A_i \rightharpoonup B_i$ we have $f_1 \times f_2 : A_1 \times A_2 \rightharpoonup B_1 \times B_2$ such that $(f_1 \times f_2)(a_1, a_2)$ is $(f_1(a_1), f_2(a_2))$ if each $f_i(a_i)$ is defined, and it is undefined otherwise.

   The rule is necessary to have $\lim_t f(t) \simeq \lim_t f(t+1)$.

2. 'Exchange' and 'weakening' rule.

   $$\mathbb{N}^n \xrightarrow{f} \mathcal{A}$$
   $$\sim \mathbb{N}^m \xrightarrow{(\pi^m_{\sigma(1)},\pi^m_{\sigma(2)},\dots,\pi^m_{\sigma(n)})} \mathbb{N}^n \xrightarrow{f} \mathcal{A}$$

   where $m \geq n$, σ is a permutation on $\{1,\dots,n\}$, and for each $k = 1,\dots,m$ the function π^m_k returns the k-th argument.

   For all $f_i : A \rightharpoonup B_i$ we have $(f_1,\dots,f_n) : A \rightharpoonup B_1 \times \cdots \times B_n$ such that $(f_1,\dots,f_n)(a)$ is $(f_1(a),\dots,f_n(a))$ if each $f_i(a)$ is defined, and it is undefined otherwise.

   The rule will make the following application operator well-defined.

Lemma 6.1 Let $i, j, k \geq 0$. Suppose we have

$$\overset{\infty}{\forall}\, (u_1,\dots,u_k) \in \mathbb{N}^k.\ \forall s_1,\dots,s_i,t_1,\dots,t_j \in \mathbb{N}.$$
$$f_1(u_1,\dots,u_k,s_1,\dots,s_i) \simeq f_2(u_1,\dots,u_k,t_1,\dots,t_j).$$

Then $f_1 \sim f_2$,

Proof. Assume that if $\vec{t} > t_0$, then $f_1(\vec{t},\vec{t_1}) \simeq f_2(\vec{t},\vec{t_2})$. Let $g_i(\vec{t},\vec{t_i}) \simeq f_i(t_1+t_0,\dots,t_k+t_0,\vec{t_i})$. Then, $f_i \sim g_i$ by kt_0 times repeated applications of the 'delay' rule. We can derive $g_1 \sim g_2$ by 'exchange' and 'weakening' rule from that $g_1(\vec{t},\vec{t_1}) \simeq g_2(\vec{t},\vec{t_2})$. By the transitivity of $\sim$, we have $f_1 \sim f_2$.

Remark 6.2 For each partial algebra X = a-lim($\mathcal{A}$), n-lim($\mathcal{A}$), or n-LIM($\mathcal{A}$), we mention the relationship between the equivalence relation $\sim$ for X and the limit structure of X.

- For a-lim($\mathcal{A}$), we have always $i = j = 0$ and $k = 1$ and for an autonomous sequence of $\mathcal{A}$-elements the sequential limit $\lim_t a \cdot \overline{t}$ corresponds to $[a]_\sim \in$ a-lim($\mathcal{A}$).
- For n-lim($\mathcal{A}$), we have always $i = j = 0$ and $k = 1$ and for a $f : \mathbb{N} \rightharpoonup \mathcal{A}$ the sequential limit $\lim_t f(t)$ corresponds to $[f]_\sim \in$ n-lim($\mathcal{A}$).
- For n-LIM($\mathcal{A}$), for a $f : \mathbb{N}^{k+i} \rightharpoonup \mathcal{A}$ the parallel limit $\lim_{t_1,\dots,t_{k+i}} f(t_1,\dots,t_{k+i})$ corresponds to $[f]_\sim \in$ n-LIM($\mathcal{A}$).

An asynchronous application operator.

For $f : \mathbb{N}^n \rightharpoonup \mathcal{A}$ and $g : \mathbb{N}^m \rightharpoonup \mathcal{A}$, define

$$[f]_\sim * [g]_\sim \ \simeq\ [h]_\sim\ ,$$

with $h : \mathbb{N}^{n+m} \rightharpoonup \mathcal{A}$ being defined as

$$\mathbb{N}^n \times \mathbb{N}^m \xrightarrow{f\times g} \mathcal{A} \times \mathcal{A} \xrightarrow{(-)\cdot(-)} \mathcal{A}$$

where $(-)\cdot(-)$ is the application operator of a given PCA $\mathcal{A}$. The operator $*$ is 'asynchronous' in the sense that $f \times g$ is involved. The operator, as well as the equivalence relation $\sim$, permits 'delay' in the arguments (as streams). Therefore,

Lemma 6.3 $(-) * (-)$ is well-defined.

We say n-LIM($\mathcal{A}$) is *the extension of a* PCA $\mathcal{A}$ *by the non-constructive parallel limits and the asynchronous application.*

Remark 6.4 We can explain the application operator with the vocabulary of the concurrency theory:

1. f and g (respectively) is a process having at most n and m (respectively) independent clocks. For all time slices[1] except for finite numbers, we can observe $\mathcal{A}$-elements.

[1] $(t_1,\dots,t_n)$ where each t_i is the value of the clock.

2. h is a process having the clocks of both f and g. Given a time slice, let a be the observation of f at a given time slice and let b be the observation of g at a given time slice. Then the observation of h at a given time slice is $a \cdot b$.

Lemma 6.5 In n-LIM($\mathcal{A}$),

1. $[k]_\sim * x * y = x$.
2. It is not the case that $[s]_\sim * x * y * z \simeq x * z * (y * z)$. But, it is the case if $x = [h]_\sim$ with some $h \in \mathcal{A}$.

Proof. Let $x = [f]_\sim, y = [g]_\sim, z = [h]_\sim$, and the arities be n, m, l.

1. The left hand side is $[h]_\sim$ such that $h = f \circ (\pi_1^{n+m} \times \cdots \times \pi_n^{n+m})$. Therefore, $h \sim f$ by the definition.
2. The left hand side is $[u]_\sim$ with u being naturally an $(n+m+l)$-ary partial function, while the right hand side is $[v]_\sim$ with v being naturally $(n+l+m+l)$-ary. So, it is difficult to have the equation unless $l = 0$ (i.e., $h \in \mathcal{A}$).

Definition 6.6 For every polynomial $t[x]$, define the polynomial $\lambda x.\, t[x]$ as follows. Let x not occur in u.

- $\lambda x.\, a \simeq [k]_\sim * a$. $\quad \lambda x.\, x \simeq [\lambda x.\, x]_\sim$.
- $\lambda x.\, u * x \simeq u$.
- $\lambda x.\, t[x] * u \simeq [\lambda xyz.\, xzy]_\sim * (\lambda x.\, t[x]) * u$.
- $\lambda x.\, u * t[x] \simeq [\lambda xyz.\, x(yz)]_\sim * u * (\lambda x.\, t[x])$.
- $\lambda x.\, t[x] * t'[x] \simeq [s]_\sim * (\lambda x.\, t[x]) * (\lambda x.\, t'[x])$.

Lemma 6.7 For every $a \in$ n-LIM($\mathcal{A}$) we have

$$(\lambda x.\, t[x])a \simeq t[a],$$

if x occurs in $t[x]$ at most once, or if $a = [h]_\sim$ with some $h \in \mathcal{A}$.

The *Church numeral* $\overline{t}$ below is defined with the λ-abstraction of Definition 6.6.

Lemma 6.8 n-LIM($\mathcal{A}$) has

1. a pairing **cons**, **car**, **cdr**; and
2. **nth** such that $\mathbf{nth} * \langle x_0, \ldots, x_n \rangle * \overline{t} \simeq x_t$.

Proof. (1) By Lemma 6.6 and equation (5). As (2), let $\mathbf{nth} \simeq \lambda xy.\, \mathbf{car} * (y * \mathbf{cdr} * x)$.

We can define similarly *the extension* a-LIM($\mathcal{A}$) *of a* PCA $\mathcal{A}$ *by the autonomous parallel limits and the asynchronous application.*

6.3 The Interpretation

Convention 6.9 In every $\lambda\mu$-term M,

1. every bound μ-variable will be distinct. That is, $\alpha \neq \beta$, if M is $\cdots(\mu\alpha.\, \cdots\alpha\cdots)\cdots(\mu\beta.\, \cdots\beta\cdots)\cdots$.
2. every bound variable is different from any free μ-variables.

Definition 6.10 Given a PCA $\mathcal{A}$. Given a type-free $\lambda\mu$-term M. Let $\{\alpha, \beta, \ldots\}$ be the set of μ-variables in M, then a set $\{t_\alpha, t_\beta, \ldots\}$ of numeric variables is denoted by M^p.

Given a $\lambda\mu$-term M, we will define the partial function M^g which denotes at most one $\mathcal{A}$-element, when the values of M^p is determined.

$$([\alpha]M)^g \quad\simeq\quad M^g(\mathbf{nth}\ \alpha\ \overline{0})\ldots(\mathbf{nth}\ \alpha\ \overline{t_\alpha});$$

$$(\mu\alpha.\, M)^g \quad\simeq\quad \lambda\alpha_0\ldots\alpha_{t_\alpha}.\, M^g\Big[\alpha := \langle\alpha_0, \ldots, \alpha_{t_\alpha}\rangle\Big].$$

$x^g \simeq x$ ranges over the elements of $\mathcal{A}$. $(MN)^g \simeq M^g N^g$ and $(\lambda x.\, M)^g \simeq \lambda x.\, M^g$.

The interpretation $[\![M]\!]$ of M in n-LIM($\mathcal{A}$) is $[M^g]_\sim$.

We will prove that this interpretation indeed works well for the type-free $\lambda\mu$-calculus.

Lemma 6.11 $(P[x := Q])^g \simeq P^g[x := Q^g]$.

Proof. By induction on P. We abbreviate $[x := Q]$ as θ, $[x := Q^g]$ as θ^g.

1. Case P is $[\alpha]M$. Then the left hand side is $([\alpha](M\theta))^g \simeq (M\theta)^g$ $(\mathbf{nth}\ \alpha\ \overline{0})$ $\ldots$ $(\mathbf{nth}\ \alpha\ \overline{t_\alpha})$. By induction hypothesis, it is $M^g\theta^g$ $(\mathbf{nth}\ \alpha\ \overline{0})$ $\ldots$ $(\mathbf{nth}\ \alpha\ \overline{t_\alpha})$ $\simeq$ $\Big(M^g\ (\mathbf{nth}\ \alpha\ \overline{0})\ \ldots\Big)\theta^g$, the right hand side.
2. Case P is $\mu\beta.\, M$. Then the left hand side is $(\mu\beta.\, (M\theta))^g \simeq \lambda\beta_0\ldots\beta_{t_\beta}.\, (M\theta)^g$ $[\beta := \langle\beta_0, \ldots, \beta_{t_\beta}\rangle]$. By induction hypothesis, it

is $\lambda\beta_0 \dots \beta_{t_\beta}. M^g\theta^g\ [\beta := \langle\beta_0, \dots, \beta_{t_\beta}\rangle]$. Because we can assume that β is not in Q without loss of generality, it is $\left(\lambda\beta_0 \dots \beta_{t_\beta}. M^g\ [\beta := \langle\beta_0, \dots, \beta_{t_\beta}\rangle]\right)\ \theta^g$, the right hand side.

When P is of the other forms, then it is trivial.

Lemma 6.12 $\left(\ P\Big[[\alpha](\ ) := (\ )x_0 \dots x_{t_\alpha}\Big]\ \right)^g \simeq P^g[\ \alpha := \langle x_0, \dots, x_{t_\alpha}\rangle\ ]$.

Proof. By induction on P. ϑ stands for $[\ [\alpha](\ ) := (\ )x_0 \dots x_{t_\alpha}\ ]$ and θ for $[\ \alpha := \langle x_0, \dots, x_{t_\alpha}\rangle\ ]$.

1. Case P is $[\alpha]M$. Then the left hand side is $(M\vartheta\ x_0 \dots x_{t_\alpha})^g$. By induction hypothesis, it is $M^g\theta x_0 \dots x_{t_\alpha}$, the right hand side.
2. Case P is $\mu\alpha. M$. Then the left hand side is $(\mu\beta. M\vartheta)^g$, which is $\lambda\beta_0 \dots \beta_{t_\beta}. (M\vartheta)^g\ [\beta := \langle\beta_0,\ \dots\ \beta_{t_\beta}\rangle]$. By induction hypothesis, it is $\lambda\beta_0 \dots \beta_{t_\beta}. M^g\theta\ [\beta := \langle\beta_0,\ \dots\ \beta_{t_\beta}\rangle]$. It is $\lambda\beta_0 \dots \beta_{t_\beta}. M^g[\beta := \langle\beta_0,\ \dots\ \beta_{t_\beta}\rangle]\theta$, the right hand side.

When P is of the other forms, then it is trivial.

Theorem 6.13 If x occurs free at most once in M, or if neither μ-application nor μ-abstraction occurs in N, then $((\lambda x. M)N)^g \simeq (M[x := N])^g$. Hence, $[\![(\lambda x. M)N]\!] \simeq [\![M[x := N]]\!]$.

Proof. The left hand side is $(\lambda x. M^g)N^g$, where x occurs free at most once in M^g, or $N \simeq [h]_\sim$ for some $h \in \mathcal{A}$. So, the left hand side is $M^g[x := N^g]$, by Lemma 6.7. It is the right hand side by Lemma 6.11.

Theorem 6.14 Let α occur free at most once in P, or let neither μ-abstraction nor μ-application occur in Q. Then

$$\Big[\!\!\Big[(\mu\alpha. P)Q\Big]\!\!\Big] \simeq \Big[\!\!\Big[\mu\alpha. P\Big[[\alpha](\ ) := [\alpha]((\ )Q)\Big]\Big]\!\!\Big].$$

Proof. We will claim:
if $f(t_\alpha, t_{\vec{\beta}}) \simeq \Big((\mu\alpha. P)Q\Big)^g$, then

$$f(t_\alpha + 1, t_{\vec{\beta}}) \simeq \Big(\mu\alpha. P\Big[[\alpha](\ ) := [\alpha]((\ )Q)\Big]\Big)^g.$$

The left hand side $f(t_\alpha + 1, t_{\vec{\beta}})$ is

$$\Big(\lambda\alpha_0 \dots \alpha_{t_\alpha+1}. P^g[\alpha := \langle\alpha_0, \dots, \alpha_{t_\alpha+1}\rangle\ ]\Big)Q^g,$$

which is, by Lemma 6.12,

$$\Big(\lambda\alpha_0 \dots\ \Big(\ P\Big[\ [\alpha](\ ) := (\ )\alpha_0 \dots \alpha_{t_\alpha+1}\ \Big]\ \Big)^g\Big)Q^g,$$

which is, by Lemma 6.11 and the premise,

$$\lambda\alpha_1 \dots \alpha_{t_\alpha+1}. \Big(\ P\Big[\ [\alpha](\ ) := (\ )Q\alpha_1 \dots \alpha_{t_\alpha+1}\ \Big]\ \Big)^g,$$

which is, by renaming bound variables,

$$\lambda\alpha_0 \dots \alpha_{t_\alpha}. \Big(\ P\Big[\ [\alpha](\ ) := (\ )Q\alpha_0 \dots \alpha_{t_\alpha}\Big]\ \Big)^g$$

Because of Convention 6.9, the mixed substitution above is a composition of two mixed substitutions:

$$\Big[\ [\alpha](\ ) := [\alpha]((\ )Q)\ \Big]\Big[\ [\alpha](\ ) := (\ )\alpha_0 \dots \alpha_{t_\alpha}\ \Big].$$

By Lemma 6.12, $f(t_\alpha + 1, t_{\vec{\beta}}) \simeq$

$$\lambda\alpha_0 \dots \alpha_{t_\alpha}. \Big(P\Big[[\alpha](\ ) := [\alpha]((\ )Q)\Big]\Big)^g\ \Big[\alpha := \langle\alpha_0, \dots, \alpha_{t_\alpha}\rangle\Big],$$

which is the right hand side

$$\Big(\mu\alpha. P\Big[[\alpha](\ ) := [\alpha]((\ )Q)\Big]\Big)^g.$$

This completes the proof of the theorem.

Theorem 6.15 **(η_{cont})** If α is not free in M, then $(\ \mu\alpha. [\alpha]M\ )^g \simeq M^g$, hence $[\![\ \mu\alpha. [\alpha]M\ ]\!] \simeq [\![M]\!]$.

Proof. The translation $(-)^g$ unwinds each occurrence α^i of α to the same sequence of usual variables $\alpha_0, \alpha_1, \dots, \alpha_{t_\alpha}$.

When we validate the μ-reduction but not η_{cont}-reduction, we can replace $(-)^g$ with another translation $(-)^G$ satisfying the following two conditions[2]. (1) for the same μ-variable α, we will

[2]These two conditions may be expressed along the line of Remark 6.4 with the vocabulary of the concurrency theory (like, "$\mu\alpha$ waits the α's inside the scope.....").

Because we comply Convention 6.9, we assign to a given $\lambda\mu$-term the largest number of clocks. Renaming of bound μ-variables of a $\lambda\mu$-term M can save the number of clocks which is assigned to M. Renaming of bound μ-variables may correspond to a certain scheduling among the parallel execution. But, I don't know.

distinguish the different occurrences by $\alpha^1, \alpha^2, \ldots$. let $(-)^G$ unwind α^i to $\alpha_0, \alpha_1, \ldots, \alpha_{t_{\alpha^i}}$

(2) For the μ-reduction with the following graphical notation

$$\begin{aligned} &(\mu\alpha^1.(\cdots([\alpha^2]P)\cdots([\alpha^3]Q)\cdots))N \\ \to_\mu\ &\mu\alpha^1.(\cdots([\alpha^2](P'N))\cdots([\alpha^3]Q')\cdots), \end{aligned} \tag{9}$$

we have $t_1 \geq t_2, t_3, \ldots$.

7 Conclusion

We hope that there is a constructive set theory such that if we construct non-autonomous limits inside it, then outside the theory the resulting limits are actually are autonomous limits.

If $\lim_t f^{(t)}(a)$ is convergent, then it is a fixpoint of f. Moreover the fixpoint may be controlled by the initial value a. So, we hope that limit may be useful to produce various interesting fixpoints.

References

[1] H.P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*. North-Holland, 2nd edition, 1984.

[2] A. Berarducci and M. Dezani-Ciancaglini. Infinite lambda-calculus and types. *TCS*, 212:29–75, 1999.

[3] M. Felleisen and D.P. Friedman. Control operators, the SECD machine, and the λ-calculus. In M. Wirsing, ed., *Formal descriptions of programming concepts III*, pages 193–217. North-Holland, 1986.

[4] K. Fujita. On proof terms and embeddings of classical substructural logics. *Studia Logica*, 61(2):199–221, 1998.

[5] E. Mark Gold. Limiting recursion. *J. S. L.*, 30:28–48, 1965.

[6] J. M. E. Hyland, P. T. Johnstone, and A. M. Pitts. Tripos theory. *Math. Proc. Cambridge Philos. Soc.*, 88(2):205–231, 1980.

[7] J.R. Kennaway, J.W. Klop, M.R. Sleep, and F.J. de Vries. Infinitary lambda calculus. *TCS*, 175(1):93–125, 1997.

[8] M. Nakata and S. Hayashi. Limiting first order realizability interpretation. *Sci. Math. Japonicae*, 2000. submitted.

[9] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Logic programming and automated reasoning*, pages 190–201. Springer, 1992.

[10] H. Schwichtenberg and S. S. Wainer. Infinite terms and recursion in higher types. In J. Diller and G. H. Müller, eds., pages 341–364. LNM, Vol. 500, 1975. Springer.

[11] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer, 1987.

[12] Th. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *J. F. P.*, 8(6):543–572, 1998.

[13] H.R. Strong. Algebraically generalized recursive function theory. *IBM J. of Research and Development*, 12:465–475, 1968.

[14] W.W. Tait. Infinitely long terms of transfinite type. In Crossley and Dummet, eds., *Formal Systems and Recursive Functions*, pages 176–185. North-Holland, 1965.

[15] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics*, vol. 123 of *Studies in Logic and the foundations of Mathematics*. North-Holland, 1988.

[16] E. G. Wagner. Uniformly reflexive structures: On the nature of Gödelizations and relative computability. *Trans. of the Amer. Math. Soc.*, 144:1–41, 1969.

[17] M. Yasugi, V. Brattka, and M. Washihara. Computability properties of Gaußian function. available via http://www.kyoto-su.ac.jp/~yasugi/Recent/gaussjuly19.ps, 1998. Revized version of an ariticle in Proc. of CCA'98,Brno.