# P Time, A Bounded Arrows Category, & Entailments (NYC CTS Talk)

Jim Otto, 3/29/23 (10:00 pm)

## 1 Introduction

In revisiting the P Time functions characterization from my thesis [CD1], the Bill Lawvere words

| | |
|---|---|
| Doctrine | Comprehension |

are key.

### 1.1 Doctrines

Our doctrines are roughly as in Kock & Reyes [KR]. While we may eventually wish them to be higher dimensional, for now they are 1-dimensional categories whose objects are small categories with chosen structure. Further they are either [AR]

| |
|---|
| locally finitely presentable |
| or locally finitely multi-presentable |

Here we consider doctrines for

| | |
|---|---|
| PR | primitive recursion |
| PTime | P Time functions |
| ZC~ | toposes with numbers, choice, & precisely 2 truth values |

The 1st 2 are locally finitely presentable, and thus have initial categories. The 3rd is likely only locally finitely multi-presentable, lacks an initial category, but instead has an initial family of categories. Keep in mind that the locally finitely presentable category of small commutative rings has the initial ring

$$\boxed{\mathbb{Z}}$$

while the locally finitely multi-presentable category of small fields has the initial family of fields

$$\boxed{\{\mathbb{Q}, \cdots \mathbb{Z}_p, \cdots\}}$$

Locally finitely presentable categories can be specified using strong models of small sets of entailments, while locally finitely multi-presentable categories can be specified using strong models of small sets of multi-entailments [MES1]. For a locally finitely presentable category, this specification by entailments allows its initial object to be constructed using answer over deduction fractions [CD3, CD2]. Here our entailments present finitely presentable arrows between our structures. While our structures are sets with a small graph acting on the left. They are inspired by Makkai's sketches [Mak], as well as by presheaves. Our multi-entailments are (modulo presentation) finite discrete base cones of our entailments, with a leg for each alternative. Strong models of small sets of multi-entailments are enough to capture classical multi-sorted 1st order logic [MEN2, Joh2]. Here strong modeling is by cone orthogonality [AR].

### 1.2 Comprehensions

Joaquín Díaz Boils [DB1] emphasizes the use of comprehensions in cutting recursion down to complexity. I learned of comprehensions from Duško Pavlović [Pav]. I use 2-comprehensions in passing from PR to PTime. They come from thinking of the ordinal 2 as a category, and then looking at its endo-functors and the natural transformations between them. 3- & V- comprehensions also appear in my thesis [CD1].

## 2 Genealogy

Our P Time functions characterization has the genealogy

| | |
|---|---|
| A Cobham (65) | L Román (89) |
| Bellantoni & Cook (92) | Me (95) |

L Román [Rom] used the PR doctrine to characterize the primitive recursive functions. A Cobham [Cob] used explicitly bounded recursion to cut down primitive recursion to the P Time functions. Bellantoni & Cook [BC2] used 2 tiers of numbers to magically make the Cobham bounds become implicit! I abstracted from a numeric arrows category to modify the PR doctrine to the PTime doctrine, which I used to characterize the P Time functions. This numeric arrows category has the tier 0 & tier 1 numbers

| | |
|---|---|
| $- : \mathbb{N} \to 1$ | $\mathrm{id} : \mathbb{N} \to \mathbb{N}$ |

Here $\mathbb{N}$ is the set $\{0,\ 1,\ 2,\ \cdots\}$ of natural numbers, 1 is the singleton set $\{0\}$, and id is the identity function.

## 3 The Primitive Recursion Characterization

The PR doctrine consists of small categories with chosen structure, finite products, and product stable natural numbers objects. Entailments for this doctrine are in [AH1]. Let Num be the category with

| | |
|---|---|
| objects | the finite products of $\mathbb{N}$ |
| arrows | the functions between these objects |

Let $I$ be the initial category in PR. Since Num is in PR, there is a unique PR functor

$$\boxed{c : I \to \mathsf{Num}}$$

L Román showed that the primitive recursive functions are precisely those functions in the image of this functor $c$.

## 3.1 Base 1

A base 1 natural numbers object in a category C, with terminal object 1, is an initial object

$$1 \xrightarrow{z} N \xleftarrow{s} N$$

in the category with objects the C objects & arrows

$$1 \xrightarrow{f} Y \xleftarrow{g} Y$$

and arrows the C commuting squares

$$\begin{array}{ccccc} 1 & \xrightarrow{f} & Y & \xleftarrow{g} & Y \\ \downarrow & & \downarrow r & & \downarrow r \\ 1 & \xrightarrow{h} & Z & \xleftarrow{k} & Z \end{array}$$

The PR doctrine is abstracted from the numeric category Num. In particular, there we can choose $1 = \{0\}$ and then have the base 1 natural numbers object

| | |
|---|---|
| $z : 1 \to \mathbb{N}$ | $\mathbb{N} \leftarrow \mathbb{N} : s$ |
| inclusion | $s\,x = x + 1$ |

## 3.2 Base 2

For (dyadic) base 2, rather than base 1, replace the initial $\mathsf{C}$ iteration diagram

$$1 \xrightarrow{z} N \xleftarrow{s} N$$

with the initial $\mathsf{C}$ iterations diagram

$$\begin{array}{ccccc} 1 & \xrightarrow{z} & N & \xleftarrow{s} & N \\ & & \uparrow{\scriptstyle t} & & \\ & & N & & \end{array}$$

In $\mathsf{Num}$ we have such a base 2 natural numbers object by

| $z : 1 \to \mathbb{N}$ | $\mathbb{N} \leftarrow \mathbb{N} : s$ | $\mathbb{N} \leftarrow \mathbb{N} : t$ |
|---|---|---|
| inclusion | $s\,x = 2x + 1$ | $t\,x = 2x + 2$ |

This is dyadic, rather than binary, since it uses digits 1 & 2 rather than 0 & 1. This avoids leading zero issues. For primitive recursion, it doesn't matter whether we use base 1 or base 2. To get the P Time functions we do need to use base 2. If we instead used base 1, we would end up with the Linear Space (in the sense of computational complexity!) functions [Bel, Rit, CD1].

## 3.3 Product Stability

For a category $C$ with object $X$, we use the polynomial category $C\,[X]$ [LS] to specify product stability. It is the full category in the slice category $C/X$ of the left projections

$$\boxed{\pi_{\mathrm{L}} : X \times Y \to X}$$

We have the functor

| $\mathrm{pb}_{-X} : C \to C[X]$ |
|---|
| $Y \mapsto$ pull back $Y \to 1$ along $X \to 1$ |

A $C$ natural numbers object is stable under products with $X$ when $\mathrm{pb}_{-X}$ takes it to a $C[X]$ natural numbers object. This last natural numbers object then sees $X$ as read only parameters. It sees the iteration vector $Y$ as read write.

## 3.4 Smallness

We consider the small sets to be the elements of a Zermelo universe $U$. For example

| $U = \bigcup_{i \in \mathbb{N}} P^i \,\mathrm{HF}$ |
|---|
| $\mathrm{HF} = \bigcup_{i \in \mathbb{N}} P^i \{\}$ |

where $P$ is power set of and HF consists of the hereditarily finite sets [MEN3, MES2].

# 4 The PTime Doctrine

The PTime doctrine consists of small categories with chosen structure, finite products, 2-comprehensions respecting the finite products, base 2 flat recursion, and base 2 safe recursion. The PTime Doctrine is abstracted from the numeric arrows category $\mathsf{Num}^2$, which has

| | |
|---|---|
| objects | the $\mathsf{Num}$ arrows |
| arrows | the $\mathsf{Num}$ commuting squares |

## 4.1 2-Comprehensions

On a category $C$, a 2-comprehension consists of endo-functors

$$\boxed{G,\, T : C \to C}$$

and natural transformations

$$G \xrightarrow{\ \epsilon\ } \mathrm{id} \xrightarrow{\ \eta\ } T$$

such that

| | |
|---|---|
| $G^2 = G$ | $G\,T = T$ |
| $T\,G = G$ | $T^2 = T$ |

| |
|---|
| $G\,\eta = T\epsilon = \eta\,\epsilon$ |
| $\epsilon\,G = G\,\epsilon = \eta\,G = \mathrm{id}$ |
| $\eta\,T = T\,\eta = \epsilon\,T = \mathrm{id}$ |

Here the id are identity functors or natural transformations. We have the functors

| | | |
|---|---|---|
| $d_0 : \mathsf{Num}^2 \to \mathsf{Num}$ | $s_0 : \mathsf{Num} \to \mathsf{Num}^2$ | $d_1 : \mathsf{Num}^2 \to \mathsf{Num}$ |
| $x : X_0 \to X_1 \mapsto X_1$ | $X \mapsto \mathrm{id} : X \to X$ | $x : X_0 \to X_1 \mapsto X_0$ |

From these we get a 2-comprehension on $\mathsf{Num}^2$ by

| | |
|---|---|
| $G = s_0\,d_1$ | $T = s_0\,d_0$ |

with $\epsilon$, $\eta$ by

$$\begin{array}{ccccc}
X_0 & \xrightarrow{\ \mathrm{id}\ } & X_0 & \xrightarrow{\ x\ } & X_1 \\
{\scriptstyle \mathrm{id}}\big\downarrow & & \big\downarrow{\scriptstyle x} & & \big\downarrow{\scriptstyle \mathrm{id}} \\
X_0 & \xrightarrow[\ x\ ]{} & X_1 & \xrightarrow[\ \mathrm{id}\ ]{} & X_1
\end{array}$$

Again, as we noted in the introduction, this all comes from actions on the ordinal 2.

## 4.2 Tier 0

We define the tier 0 subcategory $C_T$ of $C$ to be full & have

| | |
|---|---|
| objects | $C$ objects taken by $T$ to 1 |

Notice the isomorphism

$$\begin{array}{|c|}\hline \mathsf{Num} \cong \left(\mathsf{Num}^2\right)_T \\ \hline Y \mapsto - : Y \to 1 \\ \hline \end{array}$$

## 4.3 Flat Recursion

$C$ has base 2 flat recursion when $C_T$ has a sum cocone

$$\begin{array}{ccccc} 1 & \xrightarrow{z} & N & \xleftarrow{s} & N \\ & & \uparrow{\scriptstyle t} & & \\ & & N & & \end{array}$$

which is stable under products with $C$ objects $X$ in the sense that the product endo-functor

$$\boxed{X \times \_ : C \to C}$$

takes this sum cocone to a sum cocone. $\left(\mathsf{Num}^2\right)_T$ has such a sum cocone. Namely, modulo the isomorphism at the end of the last subsection,

| $z : 1 \to \mathbb{N}$ | $\mathbb{N} \leftarrow \mathbb{N} : s$ | $\mathbb{N} \leftarrow \mathbb{N} : t$ |
|---|---|---|
| inclusion | $s\,x = 2x + 1$ | $t\,x = 2x + 2$ |

The multi-stack machines in the Completeness subsection use such sum cocones. The arrows $s$, $t$ push a digit onto a stack of digits. The sum property makes decisions based on whether a stack is empty or on what digit it has on top. And pops that digit if it is there. Flat recursion [Lei] as sums was observed by R Cockett [Coc].

## 4.4 Safe Recursion

$C_T$ objects $Y$ are considered safe. While $C$ objects $X$ can, in general, be unsafe. In base 2 safe recursion, the unsafe object $G\,N$ clocks a safe read write iteration object $Y$ while having read only access to a possibly unsafe parameters object $X$. $C$ has base 2 safe recursion when for any $C$ object $X$, $C_T$ object $Y$, and $C\,[X]$ maps

$$\begin{array}{ccccc}
\mathrm{pb}_{-X}\,1 & \xrightarrow{f} & \mathrm{pb}_{-X}\,Y & \xleftarrow{g} & \mathrm{pb}_{-X}\,Y \\
 & & \uparrow{\scriptstyle h} & & \\
 & & \mathrm{pb}_{-X}\,Y & &
\end{array}$$

there exists unique $C\,[X]$ commuting

$$\begin{array}{ccccc}
\mathrm{pb}_{-X}\,G\,1 & \xrightarrow{\mathrm{pb}_{-X}\,G\,z} & \mathrm{pb}_{-X}\,G\,N & \xleftarrow{\mathrm{pb}_{-X}\,G\,s} & \mathrm{pb}_{-X}\,G\,N \\
{\scriptstyle \mathrm{id}}\downarrow & & \downarrow{\scriptstyle R} & & \downarrow{\scriptstyle R} \\
\mathrm{pb}_{-X}\,1 & \xrightarrow[f]{} & \mathrm{pb}_{-X}\,Y & \xleftarrow[g]{} & \mathrm{pb}_{-X}\,Y
\end{array}$$

$$\begin{array}{ccc}
\mathrm{pb}_{-X}\,G\,N & \xleftarrow{\mathrm{pb}_{-X}\,G\,t} & \mathrm{pb}_{-X}\,G\,N \\
\downarrow{\scriptstyle R} & & \downarrow{\scriptstyle R} \\
\mathrm{pb}_{-X}\,Y & \xleftarrow[h]{} & \mathrm{pb}_{-X}\,Y
\end{array}$$

The read only access to the possibly unsafe $X$ allows defining # (smash), the base 2 analogue of base 1 multiplication, which we sill need in the Completeness subsection.

# 5 The P Time Characterization

Let $I$ be the initial category in the PTime doctrine. Since $\mathsf{Num}^2$ is in PTime, there is a unique PTime functor

$$\boxed{c : I \to \mathsf{Num}^2}$$

Think of the $\mathsf{Num}^2$ objects as arrows going down. Then the bottoms of the commuting squares which are the $\mathsf{Num}^2$ arrows in the image of $c$ are the P Time functions.

## 5.1 Completeness

Completeness is that we so get all the P Time functions. The PTime doctrine uses base 2 numbers with digits 1 & 2 so that a numeral is just a string of 1s & 2s. Code the tapes of a multi-tape Turing machine with these numerals. Split these tapes into pairs of stacks of the digits 1 & 2. Then we have a multi-stack machine, similar Weihrauch's stack machines [Wei]. The $I$ arrows include the operations of this machine. In $I$, the analogue of base 1 addition is concatenation of strings of 1s & 2s. And the analogue of base 1 multiplication is the iterated concatenation # (smash). # allows constructing big enough functions to run the multi-stack machine, using base 2 safe recursion, for polynomial time.

## 5.2 Soundness

Soundness is that we so only get P Time functions. For this we use a subcategory $B$, with explicit time and output bounds, of the numeric arrows category $\mathsf{Num}^2$. The $B$ objects are finite products of

| the tier 0 numbers | the tier 1 numbers |
|---|---|
| $- : \mathbb{N} \to 1$ | $\mathrm{id} : \mathbb{N} \to \mathbb{N}$ |

The $B$ arrows have the form

$$\begin{array}{ccc} \mathbb{N}^i \times \mathbb{N}^j & \xrightarrow{\langle a\pi_{\mathrm{L}}, b\rangle} & \mathbb{N}^{i'} \times \mathbb{N}^{j'} \\ {\scriptstyle \pi_{\mathrm{L}}}\downarrow & & \downarrow{\scriptstyle \pi_{\mathrm{L}}} \\ \mathbb{N}^i & \xrightarrow[a]{} & \mathbb{N}^{i'} \end{array}$$

with $\pi_{\mathrm{L}}$ the left projection and $\langle a\, \pi_L,\ b\rangle$ a tuple function. In particular we have a function

$$\mathbb{N}^i \times \mathbb{N}^j \xrightarrow{b} \mathbb{N}^{j'}$$

Set

$$\boxed{|x| = \sum_{k\in i} |x_k|}$$

the total number of digits (1 or 2) in a numeric vector $x \in \mathbb{N}^i$. And set

$$\boxed{|y|_\infty = \max_{k\in j} |y_k|}$$

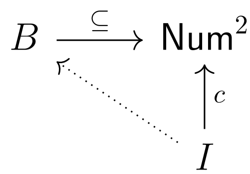
the maximal number of digits (1 or 2) in a numeric vector $y \in \mathbb{N}^j$. Also do similar with $i',\ j'$ replacing $i,\ j$. For $B$ arrows we require the explicit bounds

| | |
|---|---|
| $a\,x$ runs in time | $\leq p_a\,\lvert x\rvert$ |
| with output bound | $\lvert a\,x\rvert \leq p_a\,\lvert x\rvert$ |
| $b\,x\,y$ runs in time | $\leq q_b\,(\lvert x\rvert + \lvert y\rvert)$ |
| with output bound | $\lvert b\,x\,y\rvert_\infty \leq r_b\,\lvert x\rvert + \lvert y\rvert_\infty$ |

Here $x \in \mathbb{N}^i$, $y \in \mathbb{N}^j$ and the $p_a$, $q_b$, $r_b$ are non-negative coefficients polynomials. Since the inclusion

$$B \xrightarrow{\subseteq} \mathsf{Num}^2$$

is a PTime doctrine arrow, the unique arrow $c$ factors as




which shows soundness.

# 6 Structures

Our structures [MES1] are inspired by M Makkai's sketches [Mak], as well as by presheaves. However for us a (theory) signature is (now) any small graph. And our structures are sets with the signature acting on the left. This action is best specified using unique lifting as in discrete fibrations [Rie2].

## 6.1 Graphs

A graph $\Sigma$ consists of sets

| $\Sigma_0$ of objects | $\Sigma_1$ of arrows |
|---|---|

together with to & from functions

$$\boxed{d_0,\ d_1 : \Sigma_1 \to \Sigma_0}$$

A graph arrow takes objects to objects, arrows to arrows, and preserves to & from functions.

## 6.2 Unique Lifting

Fix a small graph $\Sigma$. A (left) $\Sigma$ structure is a graph arrow $\tau : T \to \Sigma$ satisfying the following left unique lifting property (for which we write LULP). As pictured in

$$\begin{array}{ccccc} x & \overset{\overline{f}}{\dashrightarrow} & & & T \\ & & & & \downarrow{\scriptstyle \tau} \\ A & \underset{f}{\longrightarrow} & B & & \Sigma \end{array}$$

for any $T$ object $x$ over $\Sigma$ object $A$ (in the sense that $\tau_0\, x = A$) & $\Sigma$ arrow $f : A \to B$, there exists a unique $T$ arrow $\overline{f}$ with $d_1\, \overline{f} = x$ & over $f$ (in the sense that $\tau_1\, \overline{f} = f$). We write $f \cdot x$ for this LULP arrow $\overline{f}$, as it is uniquely determined by $f$, $x$. Notice that (by LULP) any $T$ arrow $g$ has this form! Thus a $\Sigma$ structure $\tau : T \to \Sigma$ takes

| | |
|---|---|
| $\Sigma$ object $A$ | to the set $\tau_0^{-1} A = \{x \in T_0 \mid \tau_0\, x = A\}$ |
| $\Sigma$ arrow $f : A \to B$ | to the function $x \mapsto f \cdot x$ |

A $\Sigma$ structure arrow from $\Sigma$ structure $\tau : T \to \Sigma$ to $\Sigma$ structure $\upsilon : \Upsilon \to \Sigma$ is a graph arrow $\mu : T \to \Upsilon$ such that

$$\begin{array}{ccc} T & \xrightarrow{\quad \mu \quad} & \Upsilon \\ & {\scriptstyle \tau}\searrow \quad \swarrow{\scriptstyle \upsilon} & \\ & \Sigma & \end{array}$$

commutes.

## 6.3 Entailments

Fix a small graph $\Sigma$ and a $\Sigma$ structure $\tau$. A $\Sigma$ entailment is built from

| $\Sigma$ declarations | $\Sigma$ constraints |
|---|---|
| $x : A$ | $p\, x = q\, y$ |

and has the form

$$\exists! \text{ conclusion} \leftarrow \text{premise} .$$

The mode $\exists!$ is omitted when strong and weak modeling coincide, or when weak modeling is intended. The premise is a comma separated finite list of $\Sigma$ declarations and $\Sigma$ constraints. In a declaration

$$x : A$$

the $x$ is a variable, which is so declared of sort $A$, where $A$ is a $\Sigma$ object. This declaration is interpreted in $\tau$ as an element

$$\overline{x} \in \tau_0^{-1} A$$

In a constraint

$$p\, x = q\, y$$

the $p$, $q$ are $\Sigma$ paths, the $x$, $y$ are variables declared in the premise, and everything must be well-sorted. This constraint is interpreted in $\tau$ as the required equality

$$p \cdot \overline{x} = q \cdot \overline{y}$$

Here the paths act by iterating the signature action. The conclusion is a comma separated finite list of additional $\Sigma$ declarations and $\Sigma$ constraints.

## 6.4 Models

The structure $\tau$ strongly models an entailment when any interpretation of the premise in $\tau$ uniquely extends to an interpretation of the premise + conclusion in $\tau$. The modeling is instead weak when the extension exists, but is not necessarily unique.

## 6.5 Entailments for Categories

As a 1st example, we give entailments for categories. Our signature graph $\Sigma$ is that for 2D simplicial sets:

$$
\begin{array}{c}
[2] \\
{\scriptstyle d_k}\downarrow\uparrow{\scriptstyle s_j} \\
[1] \\
{\scriptstyle d_j}\downarrow\uparrow{\scriptstyle s_0} \\
[0]
\end{array}
$$

Here $j \in 2 = \{0, 1\}$, $k \in 3 = \{0, 1, 2\}$. For a $\Sigma$ structure $\tau$, the geometrical intention is

| $\tau_0^{-1}[0]$ | $\tau_0^{-1}[1]$ | $\tau_0^{-1}[2]$ |
|---|---|---|
| objects | arrows | triangles |

We need the simplicial entailments

$$
\begin{gathered}
\%\ \text{loop} \\
d_1\, s_0\, X = X,\ d_0\, s_0\, X = X \\
\leftarrow X : [0]\,.
\end{gathered}
$$

$$
\begin{gathered}
\%\ \text{triangle} \\
d_1\, d_1\, \alpha = d_1\, d_2\, \alpha,\ \%\, X \\
d_1\, d_0\, \alpha = d_0\, d_2\, \alpha,\ \%\, Y \\
d_0\, d_0\, \alpha = d_0\, d_1\, \alpha\ \ \%\, Z \\
\leftarrow \alpha : [2]\,.
\end{gathered}
$$

$$
\begin{gathered}
\%\ \text{degenerate} \\
d_2\, s_1\, f = f,\ d_1\, s_1\, f = f,\ d_0\, s_1\, f = s_0\, d_0\, f, \\
d_2\, s_0\, f = s_0\, d_1\, f,\ d_1\, s_0\, f = f,\ d_0\, s_0\, f = f \\
\leftarrow f : [1]\,.
\end{gathered}
$$

$$
\begin{gathered}
\%\ \text{doubly degenerate} \\
s_1\, s_0\, X = s_0\, s_0\, X \\
\leftarrow X : [0]\,.
\end{gathered}
$$

For the strong models to be categories, we also need the entailments

$$
\begin{gathered}
\%\ \text{composition} \\
\exists!\, \alpha : [2]\,,\ d_2\, \alpha = f,\ d_0\, \alpha = g, \\
g \circ f \Rightarrow d_1\, \alpha\ \ \%\ \text{functional sugar} \\
\leftarrow f,\ g : [1]\,,\ d_1\, f = d_0\, g.
\end{gathered}
$$

$$
\begin{gathered}
\%\ \text{associative} \\
(h \circ g) \circ f = h \circ (g \circ f) \\
\leftarrow f,\ g,\ h : [1]\,,\ d_0\, f = d_1\, g,\ d_0\, g = d_1\, h.
\end{gathered}
$$

Here % starts a comment line segment. And ∃! indicates strong modeling. Also we use functional sugar [MEN4] to make the associative entailment more readable. Dependent type sugar is also possible [MES2].

## 6.6 Folding

For an entailment $\alpha$, both its premise and its premise + conclusion can be closed up under the signature action to become finitely presented structures. Similarly $\alpha$ itself finitely presents a structure arrow $\overline{\alpha}$ between those structures [MES1]. Then for a small structure $\tau$ [AR]

| | |
|---|---|
| $\tau$ strongly models $\alpha$ | when $\tau$ is orthogonal to $\overline{\alpha}$ |
| $\tau$ weakly models $\alpha$ | when $\tau$ is injective relative to $\overline{\alpha}$ |

The folding $!\alpha$ is (modulo finite presentation) the codiagonal from pushing out $\alpha$ along itself. $\tau$ strongly models $\alpha$ when it weakly models $\{\alpha, !\alpha\}$.

## 6.7 Initial Models

Fix a small graph $\Sigma$ and a small set $\mathsf{Ax}$ of $\Sigma$ entailments. $\mathsf{Ax}$ has an initial strong model $I$ [CD3, CD2]. The $\mathsf{Ax}$ deductions close up $\mathsf{Ax}$ (modulo finite presentation) under folding, push out along structure arrows between finitely presented structures, identities, and composition. Suppose $\overline{X}$ is a $\Sigma$ premise closed up (under the signature action) to be a $\Sigma$ structure. Then any structure arrow $\overline{X} \to I$ is represented by a fraction [Bor1]

| | |
|---|---|
| $d : 0 \to \overline{Y}$ | $\overline{Y} \leftarrow \overline{X} : a$ |

with the denominator $d$ an $\mathsf{Ax}$ deduction, the numerator $a$ a structure arrow between finitely presented structures, and 0 the initial structure. Such fractions represent the same arrow when they can be put under a common denominator. $a$ is, in the sense of logic programming, an answer.

# 7 Entailments for PR

We summarize the additional entailments, beyond those for categories, for the PR doctrine as in [AH1].

## 7.1 Finite Products

Having a terminal object translates fairly easily to entailments. We expand our signature graph $\Sigma$ to

$$\begin{array}{ccccc} [2] & & & & \\ {\scriptstyle d_k}\downarrow\uparrow{\scriptstyle s_j} & & & & \\ [1] & & & & \\ {\scriptstyle d_j}\downarrow\uparrow{\scriptstyle s_0} & & & & \\ [0] & \xleftarrow{\;-\;} & \mathrm{I} & \xleftarrow{\;-\;} & \widetilde{\mathrm{I}} \end{array}$$

To the entailments for a category we add

$$\begin{gathered} \%\text{ unique arrow } !X \cdot Y \\ \exists!\, !X \cdot Y \Rightarrow t : [1]\,,\; d_1\, t = X,\; d_0\, t = -Y \\ \leftarrow X : [0]\,,\; Y : \mathrm{I}. \\ \%\text{ chosen terminal object } 1 \\ \exists!\, 1 \Rightarrow Y : \widetilde{\mathrm{I}} \leftarrow . \end{gathered}$$

To have finite products, we further expand our signature graph $\Sigma$ to

$$
\begin{array}{ccccc}
[2] & & & & \\
d_k \downarrow \uparrow s_j & & & & \\
[1] & \xleftarrow{\pi_j} & \mathrm{P} & \xleftarrow{-} & \widetilde{\mathrm{P}} \\
d_j \downarrow \uparrow s_0 & & & & \\
[0] & \xleftarrow{-} & \mathrm{I} & \xleftarrow{-} & \widetilde{\mathrm{I}}
\end{array}
$$

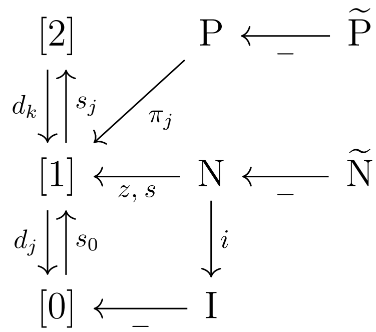
And we add the entailments

$$
\begin{gathered}
\%\ \text{cone} \\
d_1\, \pi_0\, \gamma = d_1\, \pi_1\, \gamma \leftarrow \gamma : \mathrm{P}.
\end{gathered}
$$

$$
\begin{gathered}
\%\ \text{unique tuple } \ (a,\ b) \cdot \gamma \\
\exists !\ (a,\ b) \cdot \gamma \Rightarrow f : [1]\,, \\
d_1\, f = d_1\, a,\ d_0\, f = d_1\, \pi_0\, \gamma, \\
\pi_0\, \alpha \circ f = a,\ \pi_1\, \alpha \circ f = b \\
\leftarrow X,\ Y : [0]\,,\ \gamma : \mathrm{P}, \\
d_0\, \pi_0\, \gamma = X,\ d_0\, \pi_1\, \gamma = Y, \\
a,\ b : [1]\,,\ d_1\, a = d_1\, b, \\
d_0\, a = X,\ d_0\, b = Y. \\
\%\ \text{chosen product } X \times Y \\
\exists !\, X \times Y \Rightarrow \alpha : \widetilde{\mathrm{P}}, \\
d_0\, \pi_0 - \gamma = X,\ d_0\, \pi_1 - \gamma = Y \\
\leftarrow X,\ Y : [0]\,.
\end{gathered}
$$

## 7.2 NNO

Finally we add entailments for a product stable base 1 NNO. We expand, and contract, our signature graph $\Sigma$ to



So we must eliminate the chosen terminal object entailment. Instead we will get a chosen terminal object as part of a chosen NNO. Then the base 1 stable NNO entailments are

$$\begin{gathered}\%\text{ iteration diagram}\\ d_1\, s\, \nu = N,\ d_0\, s\, \nu = N\\ \leftarrow \nu : \mathrm{N},\ N : [0]\,,\ d_0\, z\, \nu = N.\end{gathered}$$

$$
\begin{gathered}
\%\ \text{unique recursor}\ \mathcal{R}\, f\, g \cdot \_ \\
\exists!\, \mathcal{R}\, f\, g \cdot (\alpha,\ \beta,\ \nu) \Rightarrow r : [1]\,, \\
d_1\, r = d_1\, \pi_0\, \alpha,\ d_0\, r = Y, \\
r \circ (s_0\, X,\ z\, \nu \circ (!X \cdot i\, \nu)) \cdot \alpha = f, \\
r \circ (\pi_0\, \alpha,\ s\, \nu \circ \pi_1\, \alpha) \cdot \alpha = g \circ (\pi_0\, \alpha,\ r) \cdot \beta \\
\leftarrow f,\ g : [1]\,,\ \alpha,\ \beta : \mathrm{P},\ \nu : \mathrm{N}, \\
X,\ Y,\ N : [0]\,, \\
d_0\, \pi_0\, \beta = X,\ d_0\, \pi_1\, \beta = Y,\ d_0\, z\, \nu = N, \\
d_0\, \pi_0\, \alpha = X,\ d_0\, \pi_1\, \alpha = N, \\
d_1\, f = X,\ d_0\, f = Y, \\
d_1\, g = d_1\, \pi_0\, \beta,\ d_0\, g = Y.
\end{gathered}
$$

$$
\begin{gathered}
\%\ \text{chosen NNO} \\
\exists!\, \mathcal{N} \Rightarrow \nu : \widetilde{\mathrm{N}} \leftarrow .
\end{gathered}
$$

# 8 Smash

For the Completeness subsection, we need # (smash). We proceed with equations in $\mathsf{Num}$, with safe & unsafe typing from the tier 1 & tier 0 numbers in $\mathsf{Num}^2$, which can be solved using base 2 safe recursion. Here we write 0 rather than $z$.

## 8.1 $+ \rightarrow \cdot$

Base 1 addition becomes base 2 concatenation by

$$y \cdot 0 = y$$
$$y \cdot (s\,n) = s\,(y \cdot n)$$
$$y \cdot (t\,n) = t\,(y \cdot n)$$

| $y$ | $n$ |
|---|---|
| safe | unsafe |

## 8.2 $* \to \#$

Base 1 multiplication becomes base 2 smash by

$$x\#0 = x$$
$$x\#\,(s\,n) = (x\#n) \cdot x$$
$$x\#\,(t\,n) = (x\#n) \cdot x$$

| $(x\#n)$ | $x$ |
|---|---|
| safe iteration vector | unsafe parameter |

## 8.3 $\uparrow$

Base 1 exponential is iterated multiplication in the PR doctrine. Smash can not be iterated in the PTime doctrine because _#_ has no safe inputs.

# 9 The $B$ Inclusion

For the Soundness subsection, we need to check (which here we do in some haste) the inclusion

$$B \xrightarrow{\subseteq} \mathsf{Num}^2$$

for identities, composition, finite products, flat recursion, and safe recursion. Again $B$ arrows have the form

$$\begin{array}{ccc} X \times Y & \xrightarrow{\langle a\pi_{\mathrm{L}}, b\rangle} & X' \times Y' \\ {\scriptstyle \pi_{\mathrm{L}}}\downarrow & & \downarrow{\scriptstyle \pi_{\mathrm{L}}} \\ X & \xrightarrow[a]{} & X' \end{array}$$

| | | | |
|---|---|---|---|
| $X = \mathbb{N}^i$ | $Y = \mathbb{N}^j$ | $X' = \mathbb{N}^{i'}$ | $Y' = \mathbb{N}^{j'}$ |

and we are concerned with the time & output bounds

| | |
|---|---|
| $a\,x$ runs in time | $\leq p_a\,\lvert x\rvert$ |
| with output bound | $\lvert a\,x\rvert \leq p_a\,\lvert x\rvert$ |
| $b\,x\,y$ runs in time | $\leq q_b\,(\lvert x\rvert + \lvert y\rvert)$ |
| with output bound | $\lvert b\,x\,y\rvert_\infty \leq r_b\,\lvert x\rvert + \lvert y\rvert_\infty$ |

## 9.1 Multi-Stack Machines

For the time bounds we will use our multi-stack machines as sketched in the Completeness subsection. They have a finite number of stacks of digits. These stacks are numbered. The instruction types are

| | | |
|---|---|---|
| push | pop | halt |

The instruction lines have the forms & actions

| |
|---|
| label push digit stack next |
| push digit on stack; go to next |
| label pop stack next next' next" |
| try to pop stack; if digit none, 1, 2 go to next, next', next" |
| label halt |
| halt |

Time is the number of instructions executed. Notice that the $a$ output bound follows from its time bound. However the $b$ output bound is tighter than implied by its time bound.

## 9.2 Identities

The input stacks may need to be copied to output stacks, which takes linear time. But the outputs are the inputs, so that the $b$ output bound follows.

## 9.3 Composition

Composition gives

| | |
|---|---|
| $a'\,(a\,x)$ | $b'\,(a\,x)\,(b\,x\,y)$ |

Thus

$$|b'\,(a\,x)\,(b\,x\,y)|_\infty \le r_{b'}\,(p_a\,|x|) + r_b\,|x| + |y|_\infty$$

The remaining bounds compose non-negative coefficients polynomials.

## 9.4 Products

The argument for the projections is similar to that for identities. id : $\{0\} \to \{0\}$ is the terminal object in $\mathsf{Num}^2$. So going to it may need zeroing out a stack. Tuples add run times, and have fairly clear output bounds.

## 9.5 Flat Recursion

Base 2 flat recursion selects an alternative. So it bounds are fairly clear.

## 9.6 Safe Recursion

This, and composition, are the main items we need to look at. We bring back base 2 safe recursion from $C[X]$ to $C$, and then, with $\mathsf{Num}^2$ objects viewed as downward arrows, look at the tops of $\mathsf{Num}$ commuting squares. Then base 2 safe recursion in $B$ is that for any $\mathsf{Num}$ object $X$, $\left(\mathsf{Num}^2\right)_T$ object $Y$ (viewed as a $\mathsf{Num}$ object), and $\mathsf{Num}$ maps with the required bounds

$$\begin{array}{ccccc} X & \xrightarrow{\;b\;} & Y & \xleftarrow{\;b'\;} & X \times Y \\ & & \uparrow{\scriptstyle b''} & & \\ & & X \times Y & & \end{array}$$

there exists unique $\mathsf{Num}$ commuting

$$\begin{array}{ccccc} X & \xrightarrow{\langle \mathrm{id},\, z\,- \rangle} & X \times \mathbb{N} & \xleftarrow{\mathrm{id} \times s} & X \times \mathbb{N} \\ {\scriptstyle \mathrm{id}}\downarrow & & \downarrow{\scriptstyle \overline{b}} & & \downarrow{\scriptstyle \langle \pi_L, \overline{b} \rangle} \\ X & \xrightarrow[\;b\;]{} & Y & \xleftarrow[\;b'\;]{} & X \times Y \end{array}$$

$$\begin{array}{ccc} X \times \mathbb{N} & \xleftarrow{\mathrm{id} \times t} & X \times \mathbb{N} \\ \downarrow{\scriptstyle \overline{b}} & & \downarrow{\scriptstyle \langle \pi_L, \overline{b} \rangle} \\ Y & \xleftarrow[\;b''\;]{} & X \times Y \end{array}$$

whose bounds we need to check. (Stretching our bounds notation, here $X$ can be a tier 1, tier 0 hybrid, which we need to define concatenation & smash as in the Smash section. The arguments in $\overline{b}\,x\,n$ is another stretch.) On a multi-stack machine, this runs as an initialization followed by looping compositions, with the iteration vector $y$ evolving as

$$b\,x,\; b^? x\;(b\,x)\,,\; b^? x\;\left(b^? x\;(b\,x)\right),\;\cdots$$

Here ? is $'$ or $''$ as needed. Thus we have the output bound

$$\left|\bar{b}\,x\,n\right|_\infty \leq |n| \max\left(r_{b'}\,|x|\,,\ r_{b''}\,|x|\right) + r_b\,|x|$$

and thus the $\bar{b}\,x\,n$ time bound

$$\leq |n| \max\left(q_{b'}\left(|x| + j\left|\bar{b}\,x\,n\right|_\infty\right),\ q_{b''}\left(|x| + j\left|\bar{b}\,x\,n\right|_\infty\right)\right) + q_b\,|x|$$

Here $j$ is the length of the iteration vector $y$.

# 10 Research Gate

Many of my writings are at

https://www.researchgate.net/profile/Jim-Otto

# References

[AH1] Jim Otto, *Entailments for FP Categories with Stable NNO : AH1,* 2021.

[AR] Jiří Adámek & Jiří Rosický, *Locally Presentable and Accessible Categories*, Cambridge, 1994.

[BC2] S Bellantoni & S Cook, *A new recursion-theoretic characterization of the polytime functions*, in *STOC Proceedings*, ACM, 1992.

[Bel] S Bellantoni, *Predicative Recursion and Computational Complexity*, Thesis, University of Toronto, 1992.

[Bor1] Francis Borceux, *Handbook of Categorical Algebra, Volume 1,* Cambridge, 1994.

[CD1] James R Otto Jr, *Complexity Doctrines,* Thesis, McGill University, 1995.

[CD2] J Otto, *From Horn formula to Makkai sketch resolution,* 1996.

[CD3] J R Otto, *Update to Complexity Doctrines,* 1998.

[Cob] A Cobham, *The intrinsic computational difficulty of functions*, in Y Bar Hillel, *Proceedings of the 1964 International Congress of Logic, Methodology, and the Philosophy of Science*, North Holland, 1965.

[Coc] R Cockett, Conversation.

[DB1] Joaquín Díaz Boils, *Categorical Comprehensions and Recursion,* Journal of Logic and Computation, 2015.

[Joh2] Peter T Johnstone, *Sketches of an Elephant, Volumes 1–2,* Oxford, 2002.

[KR] A Kock & G E Reyes, *Doctrines in Categorical Logic*, in Jon Barwise, *Handbook of Mathematical Logic*, North Holland, 1977.

[Lei] D Leivant, *Ramified recurrence and computational complexity I*, in P Clote & J Remmel, *Feasible Mathematics II*, Birkhäuser, 1994.

[LS] J Lambek and P J Scott, *Introduction to higher order categorical logic,* Cambridge, 1986.

[Mak] M Makkai, *Generalized sketches as a framework for completeness theorems*, Preprint, 1994.

[MEN2] Jim Otto, *Classical 1st Order Logic: ME Notes 2,* 2021.

[MEN3] Jim Otto, *MacLane Universes: ME Notes 3,* 2021.

[MEN4] Jim Otto, *Categories & Path Rewriting: ME Notes 4,* 2021.

[MES1] Jim Otto, *Deconstructing Structures: ME Studies 1,* 2022.

[MES2] Jim Otto, *HF Sets & the Ackermann Relation: ME Studies 2*, 2022.

[Pav] Duško Pavlović, *Predicates and Fibrations*, Thesis, Rijksuniversiteit Utrecht, 1990.

[Rie2] Emily Riehl, *Category Theory in Context,* Dover, 2016.

[Rit] R Ritchie, *Classes of predictably computable functions*, Transactions of the AMS, 1963.

[Rom] L Román, *Cartesian Categories with Natural Numbers Object,* Journal of Pure and Applied Algebra, 1989.

[Wei] Klaus Weihrauch, *Computability*, Springer, 1987.