# A calculus from outer space!

Lucius Gregory Meredith[1]

Managing Partner, RTech Unchained 9336 California Ave SW, Seattle, WA 98103, USA
lgreg.meredith@gmail.com

**Abstract.** We describe a discrete calculus built from the intuitions informing the Einstein field equations.

## 1 Introduction

One of the notable things about the Einstein field equations [12] is their mutual recursion. The stress energy tensor describing the distribution of matter and energy informs the metric tensor, which in turn informs the stress energy tensor. Any red blooded computer scientist sees this and immediately wants to express it as a couple of mutually recursive functions and/or data structures.

Meanwhile, in computer science a revolution in the notion of location has happened and all but gone unnoticed. On the one hand, Huet's zipper [7] generalizes the intuitions of the Dedekind cut for all differentiable data structures. Specifically, for a data structure described by a differentiable functor, $T$, an element of the data type $\partial T \times T$ describes a location in an element of $T$ [8].

On the other, Milner's $\pi$-calculus transforms the $\lambda$-calculus by reifying the site of interaction as a channel, and introducing parallel composition [11]. This change simultaneously gives computations autonomy, allowing them to roam far outside the bounds of mere textual juxtaposition, and yet a means of finding each other to effect data exchange and computational evolution. Meredith's rho-calculus takes Milner's insight a step further [9], making the reified site of interaction into a fully first class entity with a complexity of structure equivalent to computations.

A question worth exploring is whether these two notions of location can be combined, yielding a site of interaction corresponding to a location in a computation. The thought is that such a reconciliation of the two ideas would allow for a computational calculus enjoying something like the intuitions underlying the mutually recursive dynamics in Einstein's field equations. Specifically, locations say where computation is distributed, computation changes where things are located.

In this paper we describe a calculus that does effect a reconciliation of the two notions of location. It provides a sandbox for exploring a wide range of discrete space-like phenomena in a fairly traditional, compact and tractable calculus. In particular, it enjoys a notion of bisimulation, allowing one to reason equationally about computational behavior, and by extension about spatial phenomena.

### 1.1 Summary of contributions and outline of paper

**Summary of contributions** We present a calculus, dubbed the space calculus (from outer space!) whose primitives are channels built from locations in computations reified as data structures. This gives us the ability to model a variety of space-like phenomena. One important feature of the calculus is that space *emerges* from computation. This makes it very distinct from work like Cardelli and Gardner's processes in space [4] where they embed more standard geometric primitives in a process calculus setting.

**Outline of paper** We provide the source of inspirations for the calculus and in so doing provide a more colorful review of related work. Then we present the calculus and a few examples of calculating with it. We present our main theorem demonstrating a fully abstract encoding of the ambient calculus into the space calculus, providing what we believe to be the first fully abstract encoding of an ambient style calculus into a substitution-based calculus. Finally, we conclude with some directions for future work.

## 2 Chemical intuitions

Huet's original insight was that we can describe a location in a tree via a pair consisting of a context (a tree with a hole in it) and a tree (think of it as a subtree to be plugged into the hole). McBride's analysis of the zipper begins by with the brilliant observation that contexts can be calculated for a wide range of data types, namely those expressible as differentiable functors. The two ideas in combination give a calculus for computing locations in data structures.

Meanwhile, Milner proposed the radical idea of turning functions into processes by liberating computational interaction from textual juxtaposition [10]. In the $\lambda$-calculus $\beta$-reduction only happens when an abstraction is applied to a term. That is, when they are *placed next to each other in a reduction context*. The $\pi$-calculus refines this idea, first by providing a symmetric, associative term constructor, namely parallel composition, that allows terms to freely mix (hence Berry and Boudol's intuitions regarding chemical abstract machines [2] were so apt a semantic framework for understanding Milner's calculus).

Having liberated terms from mere juxtaposition, Milner needed to provide a way for data exchange and computational evolution to occur. This is accomplished in the $\pi$-calculus by reifying the site of interaction in the form of a channel. These freer, more mobile computations are called processes, as opposed to the $\lambda$-calculus' *functions*.

Milner's calculus, however, suffers a flaw in that a theory of names must simultaneously be countably infinite, enjoy an effective equality, and *have no internal structure*. Taking all three together makes them unrealizable in modern computers. Meredith's reflective, higher-order calculus (rho-calculus, for short) remedies the situation by making channels be *quoted processes*, thus supplying internal structure and higher order capabilities in a single feature [9].

In a semantic setting the shift from $\pi$ to rho-calculus can be seen in terms of a domain equation and its fixpoint.

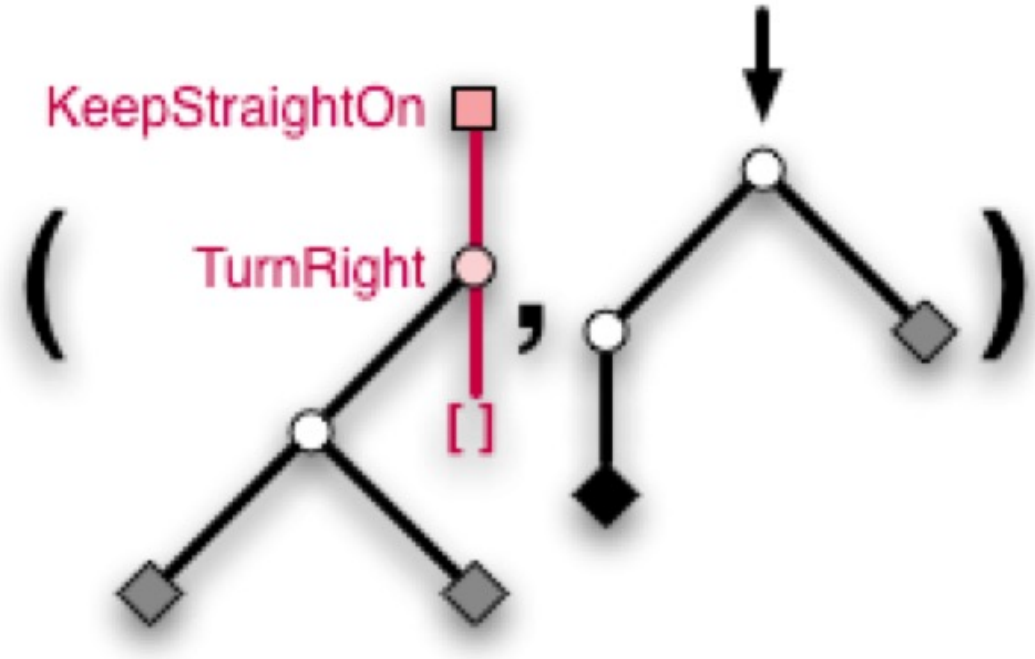


**Fig. 1.** Tree location

$$P[X] = 1 + (X \times P[X]) + (X \times X) \times P[X] + (P[X] \times P[X]) + X$$

$$R = P[R]$$

The domain equation is indeed a differentiable functor, and this we can provide a notion of location in a process term, that is $L[X] := P[X] \times \partial P[X]$. This allows us to reconcile channels as locations in a computation.

$$LP = P[P[LP] \times \partial P[LP]]$$

Your author wrote down this equation in 2008 and could not find a calculus that solved it until 2018. The calculus presented below is the solution he discovered in 2018.

The key idea returns again to *chemical* intuitions. The first step is to introduction a *catalyst* that must be present in order for redexes to reduce. In the rho-calculus a term of the form $\mathsf{for}(y \leftarrow x)P|x!(Q)$ would ordinarily reduce to a term of the form $P\{@Q/y\}$. In the space calculus (from outer space!) that term is inert. It must have a catalyst of the form $\mathsf{COMM}(K)$ mixed in, i.e. $\mathsf{COMM}(K)|\mathsf{for}(y \leftarrow x)P|x!(Q)$, before it can reduce. This feature makes the space calculus (from outer space!) the only process calculus genuinely requiring a ternary interaction rule.

The catalyst is necessary because names are now not merely quoted processes, but pairs of contexts and processes, $@\langle K, Q\rangle$. Additionally, while the rho-calculus leaves unbound

terms of the form $*@P$ inert, the space calculus (don't worry i'm not going to keep saying "from outer space!" over and over again) allows these terms to evolve to produce catalysts (thus allowing for autocatalysis). Foreshadowing what's to come, we'll see a new reduction rule of the form

$$\mathsf{U}(x)|*@\langle K, Q\rangle \rightarrow \mathsf{COMM}(K)|x!(Q)$$

## 3 Contextual interlude

There is a natural half-way point on the way to these ideas. There is an extension of the rho-calculus where the COMM rule is parameterized by a context. In the original rho-calculus, we have terms of the form

PROCESS
$$P, Q ::= 0 \mid \mathsf{for}(y \leftarrow x)P \mid x!(Q) \mid P|Q \mid *x$$

NAME
$$x, y ::= @P$$

And a core reduction rule of the form

COMM
$$\frac{x_t \equiv_{\mathsf{N}} x_s}{\mathsf{for}(y \leftarrow x_t)P \mid x_s!(Q) \rightarrow P\{@Q/y\}}$$

We can contemplate a parameterization of this rule in terms of a context parameter $K$.

COMM(K)
$$\frac{x_t \equiv_{\mathsf{N}} x_s}{\mathsf{for}(y \leftarrow x_t)P \mid x_s!(Q) \rightarrow P\{@K[Q]/y\}}$$

This choice allows for multiple instances of the rule using different contexts, such as $K_1$ and $K_2$. Thus, affording us a notion of composition of calculi, say $\langle \mathsf{Proc}, \mathsf{COMM}(K_1)\rangle$, and $\langle \mathsf{Proc}, \mathsf{COMM}(K_2)\rangle$ through the composition of contexts, yielding $\langle \mathsf{Proc}, \mathsf{COMM}(K_2 \circ K_1)\rangle$. Where we use Proc for the set of terms freely generated by the grammar.

This idea is useful for modeling "impedance matching" of behaviors at different layers or different scales. For example, the TCP protocol is at a different layer than the HTTP protocol. Yet, it is common for applications to utilize both protocols. We can use the contexts to factor all the boilerplate protocol translation from one layer to the other. A similar sort of application can be found in modeling chemical versus biochemical versus biological level phenomena.

As useful as the technique is, it is fixed. It doesn't allow for programmable contexts. The space calculus can be thought of as the next natural step in this line of reasoning, affording a dynamic, programmable factorization of a contextual interface.

# 4 Delimited continuations

There is a third source of inspiration for the space calculus: delimited continuations [6]. One way to interpret delimited continuations is as the reification of an evaluation context [1]. The space calculus takes this idea quite literally, making evaluation contexts not only first class, but necessary catalysts in a reduction.

# 5 The calculus from outer space

## 5.1 Syntax and semantics

We now give a technical presentation of the calculus. The typical presentation of such a calculus follows the style of giving generators and relations on them. The grammar, below, describing term constructors, freely generates the set of processes, Proc. This set is then quotiented by a relation known as structural congruence and it is over this set that the notion of dynamics is expressed.

### Process grammar

PROCESS

$$P, Q ::= 0 \mid \mathsf{U}(x) \mid \mathsf{for}(y \leftarrow x)P \mid x!(Q) \mid P|Q \mid *x \mid \mathsf{COMM}(K)$$

NAME

$$x, y ::= @\langle K, P\rangle$$

CONTEXT

$$K ::= \square \mid \mathsf{for}(y \leftarrow x)K \mid x!(K) \mid P|K$$

**Definition 1.** Free and bound names *The calculation of the free names of a process, $P$, denoted* $\mathsf{FN}(P)$ *is given recursively by*

$$\mathsf{FN}(0) = \emptyset \qquad \mathsf{FN}(\mathsf{U}(x)) = \{x\} \qquad \mathsf{FN}(\mathsf{for}(y \leftarrow x)P) = \{x\} \cup \mathsf{FN}(P) \setminus \{y\}$$

$$\mathsf{FN}(x!(P)) = \{x\} \cup \mathsf{FN}(P) \qquad \mathsf{FN}(P|Q) = \mathsf{FN}(P) \cup \mathsf{FN}(Q) \qquad \mathsf{FN}(*x) = \{x\}$$

$$\mathsf{FN}(\mathsf{COMM}(K)) = \mathsf{FN}(K) \qquad \mathsf{FN}(\square) = \emptyset \qquad \mathsf{FN}(\mathsf{for}(y \leftarrow x)K) = \{x\} \cup \mathsf{FN}(K) \setminus \{y\}$$

$$\mathsf{FN}(x!(K)) = \{x\} \cup \mathsf{FN}(K) \qquad \mathsf{FN}(P|K) = \mathsf{FN}(P) \cup \mathsf{FN}(K)$$

*An occurrence of $x$ in a process $P$ is bound if it is not free. The set of names occurring in a process (bound or free) is denoted by* $\mathsf{N}(P)$.

## 5.2 Substitution

We use Proc for the set of processes, @Proc for the set of names, and $\{\overrightarrow{y}/\overrightarrow{x}\}$ to denote partial maps, $s : @\mathsf{Proc} \to @\mathsf{Proc}$. A map, $s$ lifts, uniquely, to a map on process terms, $\widehat{s} : \mathsf{Proc} \to \mathsf{Proc}$. Historically, it is convention to use $\sigma$ to range over lifted subsitutions, $\widehat{s}$, to write the application of a substitution, $\sigma$ to a process, $P$, with the substitution on the

right, $P\sigma$, and the application of a substitution, $s$, to a name, $x$, using standard function application notation, $s(x)$. In this instance we choose not to swim against the tides of history. Thus,

**Definition 2.** *given $x = @P'$, $u = @\langle K, Q'\rangle$, $s = \{u/x\}$ we define the lifting of $s$ to $\widehat{s}$ (written below as $\sigma$) recursively by the following equations.*

$$0\sigma := 0$$

$$(P|Q)\sigma := P\sigma|Q\sigma$$

$$(\mathsf{for}(y \leftarrow v)P)\sigma := \mathsf{for}(z \leftarrow \sigma(v))((P\widehat{\{z/y\}})\sigma)$$

$$(x!(Q))\sigma := \sigma(x)!(Q\sigma)$$

$$(*y)\sigma := \begin{cases} K[Q'] & y \equiv_{\mathsf{N}} x \\ *y & otherwise \end{cases}$$

*where*

$$\widehat{\{w/v\}}(x) = \{w/v\}(x) = \begin{cases} w & x \equiv_{\mathsf{N}} v \\ x & otherwise \end{cases}$$

and $z$ is fresh for $P$. Our $\alpha$-equivalence will be built in the standard way from this substitution.

**Definition 3.** *Then two processes, $P, Q$, are alpha-equivalent if $P = Q\{\overrightarrow{y}/\overrightarrow{x}\}$ for some $\overrightarrow{x} \subseteq \mathsf{BN}(Q)$, $\overrightarrow{y} \subseteq \mathsf{BN}(P)$, where $Q\{\overrightarrow{y}/\overrightarrow{x}\}$ denotes the capture-avoiding substitution of $\overrightarrow{y}$ for $\overrightarrow{x}$ in $Q$.*

**Definition 4.** *The* structural congruence $\equiv$ *between processes [13] is the least congruence containing alpha-equivalence and satisfying the commutative monoid laws (associativity, commutativity and $\mathsf{0}$ as identity) for parallel composition $|$.*

**Definition 5.** *The* name equivalence $\equiv_{\mathsf{N}}$ *is the least congruence satisfying these equations*

$$\textsc{Quote-drop}\quad @*x \equiv_{\mathsf{N}} x \qquad\qquad \textsc{Struct-equiv}\quad \frac{P \equiv Q}{@P \equiv_{\mathsf{N}} @Q}$$

The astute reader will have noticed that the mutual recursion of names and processes imposes a mutual recursion on alpha-equivalence and structural equivalence via name-equivalence. Fortunately, all of this works out pleasantly and we may calculate in the natural way, free of concern. The reader interested in the details is referred to [9].

*Remark 1.* One particularly useful consequence of these definitions is that $\forall K, P.@\langle K, P\rangle \notin \mathsf{FN}(P)$. It gives us a succinct way to construct a name that is distinct from all the names in $P$ and hence fresh in the context of $P$. For those readers familiar with the work of Pitts and Gabbay, this consequence allows the system to completely obviate the need for a fresh operator, and likewise provides a canonical approach to the semantics of freshness.

### 5.3 Operational semantics

Finally, we introduce the computational dynamics. What marks these algebras as distinct from other more traditionally studied algebraic structures, e.g. vector spaces or polynomial rings, is the manner in which dynamics is captured. In traditional structures, dynamics is typically expressed through morphisms between such structures, as in linear maps between vector spaces or morphisms between rings. In algebras associated with the semantics of computation, the dynamics is expressed as part of the algebraic structure itself, through a reduction reduction relation typically denoted by $\rightarrow$. Below, we give a recursive presentation of this relation for the calculus used in the encoding.

$$\textsc{Catalyze}\quad \mathsf{U}(x) \mid *@\langle K, Q\rangle \rightarrow \mathsf{COMM}(K) \mid x!(Q)$$

$$\textsc{Comm}\quad \frac{x_t \equiv_{\mathsf{N}} x_s}{\mathsf{COMM}(K) \mid \mathsf{for}(y \leftarrow x_t)P \mid x_s!(Q) \rightarrow P\{@\langle K, Q\rangle/y\}}$$

$$\textsc{Par}\quad \frac{P \rightarrow P'}{P|Q \rightarrow P'|Q}$$

$$\textsc{Equiv}\quad \frac{P \equiv P' \qquad P' \rightarrow Q' \qquad Q' \equiv Q}{P \rightarrow Q}$$

We write $P \rightarrow$ if $\exists Q$ such that $P \rightarrow Q$ and $P \not\rightarrow$, otherwise.

### 5.4 Movement in space: an example

In the following example let $x = @\langle \square, 0\rangle$ and $y = @\langle K, Q\rangle$ for some $K$ and $Q$.

$$\begin{aligned} & \mathsf{for}(y \leftarrow x) *y | x!(P) | \mathsf{COMM}(K_1) \\ \rightarrow \quad & *y\{@\langle K_1, P\rangle/y\} \\ \rightarrow \quad & K_1[P] \end{aligned}$$

Now, if $K_1$ is also of the form $\mathsf{for}(y' \leftarrow x') *y' | x'!(\square) | \mathsf{COMM}(K_2) | R$, then $P$ will move to the location $K_2[P]|R$. That is, $K_1[P] \rightarrow K_2[P]|R$. And if $K_2$ is likewise of the form $\mathsf{for}(y'' \leftarrow x'') *y'' | x''!(\square) | \mathsf{COMM}(K_3) | R'$, then $P$ will move to the location $K_3[P]|R|R'$.

Thus, we have a means to describe movement of a process from location to location.

## 6 Replication

TBD

**Bisimulation** The computational dynamics gives rise to another kind of equivalence, the equivalence of computational behavior. As previously mentioned this is typically captured *via* some form of bisimulation.

The notion we use in this paper is derived from weak barbed bisimulation [11].

**Definition 6.** *An* observation relation, $\downarrow_{\mathcal{N}}$, *over a set of names,* $\mathcal{N}$, *is the smallest relation satisfying the rules below.*

$$\frac{y \in \mathcal{N},\ x \equiv_{\mathsf{N}} y}{x!(v) \downarrow_{\mathcal{N}} x} \qquad (\textsc{Out-barb})$$

$$\frac{P \downarrow_{\mathcal{N}} x \ \text{or}\ Q \downarrow_{\mathcal{N}} x}{P|Q \downarrow_{\mathcal{N}} x} \qquad (\textsc{Par-barb})$$

*We write* $P \Downarrow_{\mathcal{N}} x$ *if there is* $Q$ *such that* $P \Rightarrow Q$ *and* $Q \downarrow_{\mathcal{N}} x$.

**Definition 7.** *An* $\mathcal{N}$-barbed bisimulation *over a set of names,* $\mathcal{N}$, *is a symmetric binary relation* $\mathcal{S}_{\mathcal{N}}$ *between agents such that* $P\ \mathcal{S}_{\mathcal{N}} Q$ *implies:*

1. *If* $P \rightarrow P'$ *then* $Q \Rightarrow Q'$ *and* $P'\ \mathcal{S}_{\mathcal{N}} Q'$.
2. *If* $P \downarrow_{\mathcal{N}} x$, *then* $Q \Downarrow_{\mathcal{N}} x$.

$P$ *is* $\mathcal{N}$*-barbed bisimilar to* $Q$, *written* $P \dot{\approx}_{\mathcal{N}} Q$, *if* $P\ \mathcal{S}_{\mathcal{N}} Q$ *for some* $\mathcal{N}$*-barbed bisimulation* $\mathcal{S}_{\mathcal{N}}$.

# 7 Main theorem

There are calculi that emphasize space. In particular, Cardelli and Gordon proposed the ambient calculus [5]. It is strikingly different from the $\pi$ and rho-calculus. Specifically, the engine of computation in Milner's and Meredith's calculi is substitution. This is a local transformation of a single term. The pure ambient calculus evolution does not use substitution, but whole term tree rewrite. This is astonishingly expressive. In fact, Cardelli has often pointed out that there is fully abstract encoding of the ambient calculus into the *pi*-calculus. Beyond the ambient calculus, Cardelli has proposed his brane calculi [3].

We don't address this more recent development. Instead, we demonstrate a fully abstract encoding of the ambient calculus into the space calculus showing a substitution-based calculus can enjoy much of the same characteristics of a whole tree term rewrite calculus.

## 7.1 Review of the ambient calculus

PROCESS

$$P, Q ::= 0 \mid n[P] \mid M.(P) \mid (\mathsf{new}\ n)P \mid P|Q$$

ACTION

$$m ::= \mathsf{in}\ n \mid \mathsf{out}\ n \mid \mathsf{open}\ n$$

## 7.2 Encoding

TBD

### 7.3 Full abstraction

**Theorem 1.** $P \dot{\approx} Q \iff [\![P]\!] \dot{\approx} [\![Q]\!]$

## 8 Conclusions and future work

We have presented a calculus enjoying a notion of space derived from a reconciliation of Huet's zipper and Milner's channels. We have shown this suffices to encode calculi like Cardelli and Gordon's ambient calculus.

One of the interesting directions of investigation is to integrate these ideas with the quantum interpretation of the rho-calculus, using a 2-norm probability distribution as the source of non-determinism. This brings together in a single, compact calculus both space-like phenomena and quantum non-determinism.

*Acknowledgments.* The author wishes to thank Iceland Air. Their excellent service from Seattle to Keflavik allowed the author to work undisturbed when the flash of insight that resulted in the ideas in this paper suddenly dawned.

# References


1. Zena M. Ariola, Hugo Herbelin, and Amr Sabry. A type-theoretic foundation of continuations and prompts. In *ICFP*, pages 40–53. ACM, 2004.
2. Gérard Berry and Gérard Boudol. The chemical abstract machine. *Theor. Comput. Sci.*, 96(1):217–248, 1992.
3. Luca Cardelli. Brane calculi. In *CMSB*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–278. Springer, 2004.
4. Luca Cardelli and Philippa Gardner. Processes in space. In *CiE*, volume 6158 of *Lecture Notes in Computer Science*, pages 78–87. Springer, 2010.
5. Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *HOOTS*, volume 10 of *Electronic Notes in Theoretical Computer Science*, pages 198–201. Elsevier, 1997.
6. R. Kent Dybvig, Simon L. Peyton Jones, and Amr Sabry. A monadic framework for delimited continuations. *J. Funct. Program.*, 17(6):687–730, 2007.
7. Gérard P. Huet. The zipper. *J. Funct. Program.*, 7(5):549–554, 1997.
8. Conor McBride. Clowns to the left of me, jokers to the right (pearl): dissecting data structures. In *POPL*, pages 287–295. ACM, 2008.
9. L. Gregory Meredith and Matthias Radestock. A reflective higher-order calculus. *Electr. Notes Theor. Comput. Sci.*, 141(5):49–67, 2005.
10. Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992.
11. Robin Milner. The polyadic $\pi$-calculus: A tutorial. *Logic and Algebra of Specification*, Springer-Verlag, 1993.
12. C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation*. San Francisco: W.H. Freeman and Co., 1973, 1973.
13. Davide Sangiorgi and David Walker. *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press, 2001.