Linear and Monoidal Differential Turing Categories

Isaiah Hilsenrath and Jean-Simon Pacaud Lemay

May 19, 2026

1 Introduction

Differential λ-calculus [7] is an extension of λ-calculus which enables one to treat differentiation of analytic functions purely syntactically. Just as simply typed λ-calculus is sound and complete with respect to Cartesian closed categories by the Curry-Howard-Lambek correspondence, simply typed differential λ-calculus is sound and complete with respect to Cartesian (closed) differential categories [4]. In [5], Cockett and Gallagher refined these ideas to provide a categorical semantics of the *untyped* differential λ-calculus, introducing the Cartesian differential Turing category with differential canonical codes. This coherently combined a Cartesian differential category with a type of category that provides a sound and complete semantics for the ordinary untyped λ-calculus: a Turing category with canonical codes.

Now, an important source of examples of Cartesian differential categories comes from the categorical semantics of differential linear logic: monoidal differential categories [3]. Briefly, a monoidal differential category is a symmetric monoidal category with a comonad ! and a natural isomorphism $d_A\colon !A \otimes A \to !A$, called the deriving transformation, that satisfies certain coherences that capture the fundamental properties of differentiation, like the product rule and chain rule. The coKleisli category of a monoidal differential category is a Cartesian differential category [2]. It is then natural to ask what is the analogue of a monoidal differential category whose coKleisli is a Cartesian differential Turing category.

We solve this problem in two stages. First, we define a type of linear category called the linear Turing category (with canonical codes): a linear analogue of the ordinary Turing category (with canonical codes). We show that the coKleisli category of a linear Turing category is an ordinary Turing category. Further, we provide a sufficient condition for its coEilenberg-Moore category to be an ordinary Turing category. Second, we define a monoidal differential Turing category: a linear Turing category with a differential category structure. We show that that this category is the solution to our question. That is, we show that the coKleisli category of a monoidal differential Turing category with *additive* canonical codes is indeed a Cartesian differential Turing category with *differential* canonical codes. We also show that this type of category is the optimal solution and provide an example of a monoidal differential Turing category: the category of computably enumerable relations.

2 Cartesian Differential Turing Categories

Let's recall the motivation for Turing categories and Cartesian differential Turing categories. Recall that the two key term-formation rules of simply-typed λ-calculus are application and abstraction, as shown below:

$$\frac{\Gamma \vdash m\colon A \to B \qquad \Gamma \vdash n\colon A}{\Gamma \vdash mn\colon B}\ (\text{App.}) \qquad\qquad \frac{\Gamma, x\colon A, \Gamma' \vdash m\colon B}{\Gamma, \Gamma' \vdash \lambda x.m\colon A \to B}\ (\text{Abs.})$$

One important observation is that we can capture untyped λ-calculus by using the same term-formation rules as in simply-typed λ-calculus, but with a universal type U, giving us the following rules instead:

$$\frac{\Gamma \vdash m\colon U \qquad \Gamma \vdash n\colon U}{\Gamma \vdash mn\colon U}\ (\text{App.}) \qquad\qquad \frac{\Gamma, x\colon U, \Gamma' \vdash m\colon U}{\Gamma, \Gamma' \vdash \lambda x.m\colon U}\ (\text{Abs.})$$

So, if Cartesian closed categories provide a categorical semantics of simply-typed λ-calculus and untyped λ-calculus is obtained from simply-typed λ-calculus by replacing implicational types with a universal type, we would imagine that a categorical semantics for untyped λ-calculus is a category similar to a Cartesian closed category, but with exponential objects replaced by a fixed "universal" object. This leads us to the definition of a Turing category.

Definition 2.1. A **Turing category**[1] $\mathbb{X}$ is a Cartesian category with an object T and a morphism $T \times B \xrightarrow{\bullet_{B,C}} C$ (called **application**) for each pair of objects B and C with the following property: for each $A \times B \xrightarrow{f} C$, $\exists\ A \xrightarrow{c_f} T$ such that the diagram

$$\begin{array}{ccc} T \times B & \xrightarrow{\bullet} & C \\ {\scriptstyle c_f \times \mathrm{id}_B}\uparrow & \nearrow_{f} & \\ A \times B & & \end{array}$$

commutes. c_f is referred to as a **code** of f.

Note that the code of a morphism is not unique as it would be in a Cartesian closed category, but we would still like a consistent choice of codes that is compatible with composition.

Definition 2.2. A Turing category $\mathbb{X}$ has **canonical codes** when there exists a function $\lambda_{A,B,C} \colon \mathbb{X}(A \times B, C) \to \mathbb{X}(A, T)$ for each triple of objects A, B, C such that the diagrams

$$\begin{array}{ccc} T \times B & \xrightarrow{\bullet_{B,C}} & C \\ {\scriptstyle \lambda_{A,B,C}[f] \times \mathrm{id}_B}\uparrow & \nearrow_{f} & \\ A \times B & & \end{array} \qquad \begin{array}{ccc} & & T \\ & \nearrow_{\lambda_{D,B,C}[(g \times \mathrm{id}_B);f]} & \uparrow{\scriptstyle \lambda(f)} \\ D & \xrightarrow[g]{} & A \end{array}$$

commute for all $A \times B \xrightarrow{f} C$ and $D \xrightarrow{g} A$.

As planned, untyped λ-calculus is sound and complete with respect to Turing categories with canonical codes [5, Theorem 2.1-2.2].

Theorem 2.3. *Let $\mathcal{C}(\Lambda^u_\beta)$ be the category induced by the untyped λ-calculus.*

1. *$\mathcal{C}(\Lambda^u_\beta)$ is a Turing category with canonical codes, where $U \times U \xrightarrow{\bullet_{U,U}} U = (m \colon U, n \colon U) \vdash mn \colon U$.*
2. *Given a Turing category $\mathbb{X}$ with canonical codes, there exists a Cartesian functor $(\!|-|\!) \colon \mathcal{C}(\Lambda^u_\beta) \to \mathbb{X}$ preserving Turing structure.*

Now, we would like to improve this to a categorical semantics for the differential λ-calculus, which is a type of untyped λ-calculus with the new term-formation rules corresponding to differentiation, addition, and zero shown below:

$$\frac{\Gamma, x \colon X, \Gamma' \vdash m \colon Y \qquad \Gamma, \Gamma' \vdash p \colon X \qquad \Gamma, \Gamma' \vdash v \colon X}{\Gamma, \Gamma' \vdash \frac{dm}{dx}(p) \cdot v}\ (\text{Diff.}) \qquad \frac{\Gamma \vdash m \colon A \qquad \Gamma \vdash n \colon A}{\Gamma \vdash m + n \colon A}\ (\text{Add.})$$

$$\frac{}{\Gamma \vdash 0 \colon A}\ (\text{Zero})$$

This should be accomplished by adding a left-additive structure, to take care of (Add.) and (Zero), and an operator $D \colon \mathbb{X}(A, B) \to \mathbb{X}(A \times A, B)$ on morphisms satisfying the Leibniz rule, chain rule, and suitable coherence conditions, to take care of (Diff.). Such a category is called a Cartesian differential category. So to achieve a categorical semantics for differential λ-calculus, we want a Turing category structure and canonical codes compatible with a Cartesian differential category structure.

[1] Note that there is a more general notion of a Turing category defined on a **restriction category**. Here, we are focusing only on the so-called **total** case.

Definition 2.4. A Turing category $\mathbb{X}$ is a **Cartesian differential Turing category** (CDTC) when it is also a Cartesian differential category and its application map $\bullet_{B,C} : T \times B \to C$ is linear in its first argument for all pairs of objects B, C.

Definition 2.5. A CDTC $\mathbb{X}$ has **additive canonical codes** when $\mathbb{X}$ has canonical codes s.t.

1. for all morphisms $f, g : A \times B \to C$ in $\mathbb{X}$, $\lambda_{A,B,C}[f + g] = \lambda_{A,B,C}[f] + \lambda_{A,B,C}[g]$.
2. for all objects A, B, C in $\mathbb{X}$, $\lambda_{A,B,C}[0_{A\times B,C}] = 0_{A,T}$.

Definition 2.6. A CDTC $\mathbb{X}$ has **differential canonical codes** when $\mathbb{X}$ has additive codes s.t. for all morphisms $f : A \times B \to C$ in $\mathbb{X}$,

$$A \times A \xrightarrow{D\left[A \xrightarrow{\lambda[f]} T\right]} T = \lambda\left[(A \times A) \times B \xrightarrow{\langle \pi_0 \times \mathrm{id}, \pi_1 \times 0\rangle} (A \times B) \times (A \times B) \xrightarrow{D[f]} C\right].$$

As planned, an analogue of 2.3 holds, so differential λ-calculus is sound and complete with respect to CDTCs with differential canonical codes [5, Theorems 4.1, 4.6]. Our goal in this work is to construct monoidal differential Turing categories (MDTCs) with additive canonical codes and prove that their coKleisli categories are CDTCs with differential canonical codes. This extends the classical relationship between monoidal differential categories and Cartesian differential categories [2] to the setting of Turing categories.

3 Linear Turing Categories

Our first step is to answer our coKleisli question without the differential structures: we would like to define a category whose coKleisli category is an ordinary Turing category with canonical codes. But how should we do this? Recall that a Turing category is like a Cartesian closed category with a fixed "universal" object, rather than exponential objects; the coKleisli category of a *monoidal closed* Seely category is a Cartesian closed category [8]; and a monoidal closed category is similar to a Cartesian closed category where the Cartesian product $\times$ is replaced with the monoidal product $\otimes$. Therefore, we would expect that adding a Seely-style structure to the definition of a Turing category with canonical codes and replacing the Cartesian product $\times$ with the monoidal product $\otimes$ would give us a category whose coKleisli category is an ordinary Turing category with canonical codes. This leads us to the definition of a linear Turing category with canonical codes.

Definition 3.1. A **linear Turing category** $\mathbb{X}$ is a monoidal storage category (a.k.a. new-Seely category [1]) together with an object T and a morphism $T \otimes B \xrightarrow{\bullet_{B,C}} C$ (called **application**) for each pair of objects B and C with the following property: for each $A \otimes B \xrightarrow{f} C$, $\exists\ A \xrightarrow{c_f} T$ such that the diagram

$$\begin{array}{ccc} T \otimes B & \xrightarrow{\bullet} & C \\ {\scriptstyle c_f \otimes \mathrm{id}_B}\uparrow & \nearrow_{f} & \\ A \otimes B & & \end{array}$$

commutes. The morphism c_f is referred to as a **code** of f.

Definition 3.2. A linear Turing category $\mathbb{X}$ has **canonical codes** when there exists a function $\lambda_{A,B,C} : \mathbb{X}(A \otimes B, C) \to \mathbb{X}(A, T)$ for each triple of objects A, B, C such that the diagrams

$$\begin{array}{ccc} T \otimes B & \xrightarrow{\bullet_{B,C}} & C \\ {\scriptstyle \lambda_{A,B,C}[f] \otimes \mathrm{id}_B}\uparrow & \nearrow_{f} & \\ A \otimes B & & \end{array} \qquad \begin{array}{ccc} & & T \\ & {\scriptstyle \lambda_{D,B,C}[(g \otimes \mathrm{id}_!);f]} \nearrow & \uparrow{\scriptstyle \lambda(f)} \\ D & \xrightarrow[g]{} & A \end{array}$$

commute for all $A \otimes B \xrightarrow{f} C$ and $D \xrightarrow{g} A$.

Using the Seely isomorphism $\chi_{A,B} : !(A \times B) \to !A \otimes !B$, we can translate the application map $\bullet$ in our base linear Turing category to an application map $\bullet'$ in our coKleisli category.[2]

Theorem 3.3. *Let $\mathbb{X}$ be a LTC. Then, the coKleisli category $\mathbb{X}_!$ is a Turing category with Turing object T if for objects B, C, the application $\bullet'_{B,C} \colon T \times B \to C$ in $\mathbb{X}_!$ is defined by*

$$[\![\bullet'_{B,C}]\!] \coloneqq !(T \times B) \xrightarrow{\chi_{T,B}} !T \otimes !B \xrightarrow{\epsilon_T \otimes \mathrm{id}_{!B}} T \otimes !B \xrightarrow{\bullet_{!B,C}} C.$$

Also, for $f \colon A \times B \to C$ in $\mathbb{X}_!$, the map $c'_f \colon A \to T$ in $\mathbb{X}_!$ defined by $[\![c'_f]\!] \coloneqq !A \xrightarrow{c_{\chi^{-1}_{A,B}; [\![f]\!]}} T$ is a code of f in $\mathbb{X}_!$ if $c_{\chi^{-1}_{A,B}; [\![f]\!]}$ is a code of $!A \otimes !B \xrightarrow{\chi^{-1}_{A,B}} !(A \times B) \xrightarrow{[\![f]\!]} C$ in $\mathbb{X}$.

Theorem 3.4. *Let $\mathbb{X}$ be linear Turing category. If $\mathbb{X}$ has canonical codes given by the family of functions $\{\lambda_{A,B,C} : A, B, C \in \mathrm{Ob}\,\mathbb{X}\}$, then the Turing category $\mathbb{X}_!$ also has canonical codes when for each triple of objects A, B, C, the function $\lambda'_{A,B,C} \colon \mathbb{X}_!(A \times B, C) \to \mathbb{X}_!(A, T)$ is defined by*

$$[\![\lambda'_{A,B,C}[f]]\!] \coloneqq !A \xrightarrow{\lambda_{!A,!B,C}\left[!A \otimes !B \xrightarrow{\chi^{-1}_{A,B}} !(A \times B) \xrightarrow{[\![f]\!]} C\right]} T.$$

for all morphisms $A \times B \xrightarrow{f} C$ in $\mathbb{X}_!$.

While we won't need it for our treatment of monoidal differential Turing categories, we find an interesting analogue for the other typical linear-nonlinear adjunction, as long as the application maps are "neat."

Definition 3.5. A linear Turing category $\mathbb{X}$ is called **neat** if for each pair of objects B, C, the diagram

$$\begin{array}{ccccc} !T \otimes !B & \xrightarrow{\epsilon_T \otimes !B} & T \otimes !B & \xrightarrow{\bullet_{!B,!C}} & !C \\ \downarrow{\scriptstyle \mathrm{id}_{!T} \otimes \delta_B} & & & & \downarrow{\scriptstyle \delta_C} \\ !T \otimes !!B & \xrightarrow[m_{T,!B}]{} & !(T \otimes !B) & \xrightarrow[!\bullet_{!B,!C}]{} & !!C \end{array}$$

commutes. That is, $!T \otimes !B \xrightarrow{\epsilon_T \otimes \mathrm{id}_{!B}} T \otimes !B \xrightarrow{\bullet_{!B,!C}} !C$ is a coalgebra morphism.

Just as the coEilenberg-Moore category of a linear category is a Cartesian category when contraction and weakening are !-coalgebra morphisms, we find that the coEilenberg-Moore category of a linear Turing category (with canonical codes) is a Turing category (with canonical codes) when application $\bullet$ acts **like** a !-coalgebra morphisms on the free !-coalgebras.

Theorem 3.6. *Given a neat linear Turing category $\mathbb{X}$, its coEilenberg-Moore category $\mathbb{X}^!$ is a Turing category with Turing object $!T$ if for objects B and C, the application $\bullet'_{B,C} \colon T \otimes B \to C$ in $\mathbb{X}^!$ is defined by*

$$\bullet'_{B,C} \coloneqq !T \otimes B \xrightarrow{\epsilon_T \otimes h_B} T \otimes !B \xrightarrow{\bullet_{!B,!C}} !C \xrightarrow{h_C} C.$$

Also, for $f \colon A \otimes B \to C$ in $\mathbb{X}^!$, the map $c'_f \colon A \to T$ in $\mathbb{X}^!$ defined by $c'_f \coloneqq A \xrightarrow{h_A} !A \xrightarrow{\delta_A} !!A \xrightarrow{!c_{m_{A,B}; f}} T$ is a code of f in $\mathbb{X}^!$ if $c_{m_{A,B}; f}$ is a code of $!A \otimes !B \xrightarrow{m_{A,B}} !(A \otimes B) \xrightarrow{!f} !C$ in $\mathbb{X}$.

Theorem 3.7. *Let $\mathbb{X}$ be a neat linear Turing category. If $\mathbb{X}$ has canonical codes given by the family of function $\{\lambda_{A,B,C} : A, B, C \in \mathrm{Ob}\,\mathbb{X}\}$, then the total Turing category $\mathbb{X}^!$ also has canonical codes when for each triple of objects A, B, C, the function $\lambda'_{A,B,C} \colon \mathbb{X}^!(A \otimes B, C) \to \mathbb{X}^!(A, T)$ is defined by*

$$\lambda'_{A,B,C}[f] \coloneqq A \xrightarrow{h_A} !A \xrightarrow{\delta_A} !!A \xrightarrow{!\lambda_{!A,!B,!C}\left[!A \otimes !B \xrightarrow{m_{A,B}} !(A \otimes B) \xrightarrow{!f} !C\right]} T.$$

for all morphisms $(A, h_A) \otimes (B, h_B) \xrightarrow{f} (C, h_C)$ in $\mathbb{X}^!$.

[2] The appearance of the dereliction ϵ_T in the definition of $\bullet'$ corresponds to the fact that $A \Rightarrow B$ in intuitionistic logic corresponds to $!A \multimap B$ in linear logic (see [8]).

4 Monoidal Differential Turing Categories

Now, we can throw in our differential structures!

Definition 4.1. A linear Turing category $\mathbb{X}$ is called a **monoidal differential Turing category** (MDTC) when it is also a monoidal differential category.

It is important to note that unlike for CDTCs, in the definition of MDTC, we do not need any compatibility conditions between the differential and Turing structures. The compatibility in the coKleisli category turns out to follow automatically:

Lemma 4.2. *Let $\mathbb{X}$ be an MDTC. Then, for each pair of objects B, C, the application $\bullet'_{B,C}$ in the Cartesian differential category $\mathbb{X}_!$ is linear in its first argument.*

This lemma follows by a short computation together with [6, Corollary 23]. We also need additive canonical codes, whose definition in MDTCs mirrors definition 2.5.

Definition 4.3. An MDTC $\mathbb{X}$ has **additive canonical codes** when it has canonical codes such that

1. for all $f, g\colon A \otimes !B \to C$ in $\mathbb{X}$, $\lambda_{A,B,C}[f+g] = \lambda_{A,B,C}[f] + \lambda_{A,B,C}[g]$;
2. for all objects A, B, C in $\mathbb{X}$, $\lambda_{A,B,C}[0_{A\otimes !B,C}] = 0_{A,T}$.

Now, we can state our main theorem:

Theorem 4.4. *Let $\mathbb{X}$ be an MDTC (with additive canonical codes). Then, $\mathbb{X}_!$ is a CDTC (with differential canonical codes).*

Notice that we didn't need to develop differential canonical codes for MDTCs! The idea here is that $\langle \pi_0 \times \mathrm{id}, \pi_1 \times 0\rangle; D[f]$ in Definition 2.6 is replaced by $d_A \times \mathrm{id}; f$ in the monoidal differential setting, where d is the deriving transform of our monoidal differential category, and then d_A can be pulled out of the code by canonicity.

It turns out that theorem 4.4 is actually the optimal answer to our coKleisli question:

Theorem 4.5. *Let $\mathbb{X}$ be a monoidal differential category. Then, $\mathbb{X}_!$ is a CDTC if and only if $\mathbb{X}$ is a linear Turing category.*

That is, if we want to extend the coKleisli construction of Cartesian differential categories from monoidal differential categories to a coKleisli construction of CDTCs from a subclass of monoidal differential categories, the most general subclass comprises the monoidal differential categories that are also linear Turing categories, i.e., the class of MDTCs.

5 Example of an MDTC

We conclude with an example of an MDTC. Let **ceRel** be the subcategory of **Rel** such that the objects are computably enumerable subsets $A \subset \mathbb{N}^k \subset \mathbb{N}$, and a morphism $R\colon A \to B$ is a computably enumerable $R \subset A \times B$. We define an endofunctor ! on **ceRel** to be the usual **bag functor** on **Rel** but encoded in the natural numbers. That is, given a set A, $!A \coloneqq \{p_{a_1} \cdots p_{a_n} : n \in \mathbb{N}, a_1, \ldots, a_n \in A\} \subset \mathbb{N}$, where p_i is the ith prime number; and given a computably enumerable relation $R\colon A \to B$, $!R \coloneqq \{(p_{a_1} \cdots p_{a_n}, p_{b_1} \cdots p_{b_n}) : (a_i, b_i) \in R\ \forall i \in [n]\}$. For example, $!\mathbb{N} = \mathbb{N}^+$ by the prime decomposition theorem. Intuitively, $p_{a_1} \cdots p_{a_n} \in !A$ encodes the finite multiset $[\![a_1, \ldots, a_n]\!]$, so the multiplicity of a given $a \in A$ is reflected by the power of p_a in the prime factorization of the encoding.

Theorem 5.1. ceRel *is a monoidal differential Turing category, where bang is the bag functor !, the Turing object is $\mathbb{N}$, and we have the following structure morphisms:*

1. *application is $\bullet_{B,C} \coloneqq \{((t,b),c) : t \in \mathbb{N}, b \in B, c \in C, \Phi(t,b,z)\downarrow\}$,*
2. *the deriving transform is $d_A \coloneqq \{((\bar{a}, a), a \cdot p_a) : \bar{a} \in !A, a \in A\} \subset (!A \times A) \times !A$,*
3. *dereliction is $\epsilon_A \coloneqq \{(p_a, a) : a \in A\} \subset !A \times A$,*
4. *digging is $\delta_A \coloneqq \{(\bar{a}_1 \cdots \bar{a}_n, p_{\bar{a}_1}, \cdots, p_{\bar{a}_n}) : \bar{a}_1, \ldots, \bar{a}_n \in !A\} \subset !A \times !!A$,*

where $\Phi(t,b,z)\downarrow$ means that the universal Turing machine halts when given input (t,b,z).

References

[1] G.M. Bierman, *What is a categorical model of Intuitionistic Linear Logic?*, Lecture Notes in Computer Science **902** (1995), 78-93, Typed Lambda Calculi and Applications 1995.

[2] R.F. Blute, J.R.B. Cockett, and R.A.G. Seely, *Cartesian differential categories*, Theory and Applications of Categories **22** (2009), no. 23, 622–672.

[3] R.F. Blute, J.R.B. Cockett, and R.A.G. Seely, *Differential categories*, Mathematical Structures in Computer Science **16** (2006), no. 6, 1049–1083.

[4] A. Bucciarelli, T. Ehrhard, and G. Manzonetto, *Categorical models for simply typed resource calculi*, Electronic Notes in Theoretical Computer Science **265** (2010), 213–230, Proceedings of MFPS 2010.

[5] J.R.B. Cockett and J. Gallagher, *Categorical models of the differential-calculus*, Mathematical Structures in Computer Science **29** (2019), no. 10, 1513–1555.

[6] G. Cruttwell, J. Gallagher, J.S.P. Lemay, and D. Pronk, *Monoidal reverse differential categories*, Mathematical Structures in Computer Science **32** (2022), no. 10, 1313–1363.

[7] T. Ehrhard and L. Regnier, *The differential lambda-calculus*, Theoretical Computer Science **309** (2003), no. 1, 1–41.

[8] R.A.G. Seely, *Linear logic, $*$-autonomous categories and cofree coalgebras*, AMS Contemporary Mathematics **92** (1989), 371-382, Conference on Categories in Computer Science and Logic.