Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 286 (2012) 257–272

www.elsevier.com/locate/entcs

A Simply Typed λ-Calculus of Forward Automatic Differentiation

Oleksandr Manzyuk[1,2]

Department of Computer Science
National University of Ireland Maynooth
Maynooth, Ireland

Abstract

We present an extension of the simply typed λ-calculus with pushforward operators. This extension is motivated by the desire to incorporate forward automatic differentiation, which is an important technique in numeric computing, into functional programming. Our calculus is similar to Ehrhard and Regnier's differential λ-calculus, but is based on the differential geometric idea of pushforward rather than derivative. We prove that, like the differential λ-calculus, our calculus can be soundly interpreted in differential λ-categories.

Keywords: tangent bundle, differential λ-calculus, differential category, categorical semantics.

1 Introduction

Automatic differentiation (AD) is a powerful technique for computing derivatives of functions given by programs in programming languages [6]. AD is superior to divided differences because AD-generated derivative values are free of approximation errors, and superior to symbolic differentiation because it can handle code of very high complexity and because it gives strong computational complexity guarantees. There exist many AD systems [3] (in the form of libraries and code pre-processors). A majority of these AD systems are built on top of imperative programming languages (Fortran, C/C++), whereas the idea of AD is most naturally embodied in a functional programming language. Indeed, the differentiation operator is almost a paradigmatic example of a higher-order function. However, despite a huge body of

[1] This work was supported, in part, by Science Foundation Ireland Principal Investigator grant 09/IN.1/I2637.

[2] Email: manzyuk@gmail.com

[3] http://www.autodiff.org/?module=Tools

1571-0661
doi:10.1016/j.entcs.2012.08.017

research [4] and proliferation of AD implementations, a clear semantics of AD in the presence of first-class functions is lacking, which inhibits the incorporation of AD into functional programming. Siskind and Pearlmutter [12] discuss the problems one faces trying to extend a functional programming language with AD operators. In particular, they emphasize the subtleties of AD of higher-order functions. They describe [13] a novel AD system, Stalin∇, and claim that it correctly handles higher-order functions. Although Stalin∇ does produce the correct answers for examples where the other systems are known to fail, no correctness results are proven, which is hardly satisfactory.

In order to address these issues and to lay down a theoretical foundation for a functional programming language with support for AD, we suggest extending the λ-calculus with AD operators. In this paper, we make some first steps towards this goal. There are several variations of AD: forward, reverse, and mixtures thereof. We present a simply typed λ-calculus of forward AD, leaving the more complex reverse AD for future work.

The idea of extending the λ-calculus with differential operators is not novel. Drawing motivation from linear logic, Ehrhard and Regnier introduced the differential λ-calculus [4]. Despite its origin in the denotational semantics of linear logic, the differential λ-calculus is also an attractive foundation on which to build a functional programming language with built-in support for differentiation. Unlike, for example, symbolic differentiation, the differential λ-calculus can handle not only mathematical expressions, but arbitrary λ-terms. Most notably, it can take derivatives through and of higher-order functions. However, like symbolic differentiation, the differential λ-calculus, implemented naively, yields a grossly inefficient way to compute derivatives, suffering from the loss of sharing.

We propose a variation of the differential λ-calculus, the *perturbative λ-calculus*, which, we conjecture, does not necessitate this loss of efficiency. Like forward AD, the perturbative λ-calculus is based on the differential geometric idea of pushforward rather than that of derivative. Like the differential λ-calculus, the perturbative λ-calculus can be interpreted in an arbitrary differential λ-category of Bucciarelli et al. [3]. We prove that the interpretation is sound. We believe that the proofs of confluence and strong normalization for the differential λ-calculus given in [4] can be adapted to the perturbative λ-calculus. However, a formal treatment of these questions is left for future research.

2 Forward AD

To motivate the definition of the perturbative λ-calculus, we briefly outline the ideas behind forward AD. They are most naturally explained in differential geometric terms. Let TX denote the tangent bundle of a smooth manifold X. For example, the tangent bundle $T\mathbb{R}^n$ of the Euclidean space $\mathbb{R}^n$ can be identified with the cartesian product $\mathbb{R}^n \times \mathbb{R}^n$. An element (x', x) of the tangent bundle $T\mathbb{R}^n$ is viewed

[4] The publication database of the website `http://www.autodiff.org` contains 1277 entries at the moment of writing.

as a *primal* $x \in \mathbb{R}^n$ paired with a *tangent* (or *perturbation*) $x' \in \mathbb{R}^n$. A smooth map between smooth manifolds $f : X \to Y$ gives rise to a smooth map $Tf : TX \to TY$, called the *pushforward* of f. For example, the pushforward $Tf : T\mathbb{R}^m = \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^n = T\mathbb{R}^n$ of a smooth map $f : \mathbb{R}^m \to \mathbb{R}^n$ is given by $Tf(x', x) = (J_f(x) \cdot x', f(x))$, where $J_f(x)$ is the Jacobian of f at the point x. The correspondences $X \mapsto TX$, $f \mapsto Tf$ constitute a functor from the category of smooth manifolds to itself. Preservation of composition is a consequence of the chain rule. Furthermore, T preserves products. Informally, this means that in order to compute the pushforward of a compound function it suffices to know the pushforwards of its constituents. The implementations of forward AD take advantage of this, typically in one of the following ways:

- By overloading the primitives, so that they can accept both numbers and tangent bundle pairs as inputs. Each overloaded primitive denotes two functions: the function computed originally by that primitive and its pushforward. Then any user-defined procedure built out of the overloaded primitives also denotes (and can be used to compute by supplying arguments of appropriate types) two functions: a function $f : \mathbb{R}^m \to \mathbb{R}^n$ and its pushforward $Tf : T\mathbb{R}^m \to T\mathbb{R}^n$.
- By generating (either internally by the compiler or externally by a pre-processor) the source code of a procedure computing the pushforward $Tf : T\mathbb{R}^m \to T\mathbb{R}^n$ of a function $f : \mathbb{R}^m \to \mathbb{R}^n$ from the source code of a procedure computing the function f. This transformational approach can be seen as an enhancement of symbolic differentiation that recovers sharing by simultaneously manipulating the primal and tangent values.

In either way, defining in a program a procedure computing a function f automatically gives access to a procedure computing the pushforward of f.

Let us illustrate the method with an example covering ordinary arithmetic. Because the tangent bundle functor T preserves products, it follows that the pushforwards $T(+), T(\cdot) : T\mathbb{R} \times T\mathbb{R} \simeq T(\mathbb{R} \times \mathbb{R}) \to T\mathbb{R}$ of the operations $+, \cdot : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ equip $T\mathbb{R}$ with the structure of a ring. The tangent bundle $T\mathbb{R}$ is isomorphic as a ring to the ring $\mathbb{R}[\varepsilon]/(\varepsilon^2)$ of dual numbers via $(x', x) \mapsto x + x'\varepsilon$, and below we identify $T\mathbb{R}$ with $\mathbb{R}[\varepsilon]/(\varepsilon^2)$. The operations $+$ and $\cdot$ in $T\mathbb{R}$ correspond exactly to the pushforwards of the operations $+$ and $\cdot$ in $\mathbb{R}$. Hence, we can overload $+$ and $\cdot$ to mean both. Computing the pushforward of a function built out of $+$ and $\cdot$ amounts to interpreting the body of the function over the dual numbers.

For example, the derivative of $\lambda x.\, x^2 + 1$ at the point 3 is obtained by computing the pushforward of $\lambda x.\, x^2 + 1$ at the point $3 + 1\varepsilon = 3 + \varepsilon$ and taking the perturbation part of the result. The former amounts to evaluating the expression $x^2 + 1$ at the point $3 + \varepsilon$ interpreting $+$ and $\cdot$ as addition and multiplication of dual numbers, respectively:

$$T(\lambda x.\, x^2 + 1)(3 + \varepsilon) = (\lambda x.\, x^2 + 1)(3 + \varepsilon) = (3 + \varepsilon) \cdot (3 + \varepsilon) + 1 = 10 + 6\varepsilon.$$

In other words, the derivative of a function f at a point x can be computed as $D\, f\, x = E\, (f(x + \varepsilon))$, where E is given by $E\, (x + x'\varepsilon) = x'$.

The story becomes more complicated in the presence of first-class functions because the spaces of procedure inputs and outputs need no longer be Euclidean and may be function spaces instead. Furthermore, as pointed out by Siskind and Pearlmutter [12], implementations must be careful not to confuse the perturbations that arise when pushing the same function forward multiple times.

We illustrate this problem of perturbation confusion with an example involving nested invocations of the derivative operator D, for example $D\,(\lambda x.\, x \cdot D\,(\lambda y.\, x)\,2)\,1$. Because the inner derivative is equal to 0, the value of the whole expression should also be 0. However, applying the formula for D naively, we obtain:

$$\begin{aligned} D\,(\lambda x.\, x \cdot D\,(\lambda y.\, x)\,2)\,1 &= E\,((\lambda x.\, x \cdot D\,(\lambda y.\, x)\,2)(1+\varepsilon)) \\ &= E\,((1+\varepsilon) \cdot D\,(\lambda y.\, (1+\varepsilon))\,2) \\ &= E\,((1+\varepsilon) \cdot E\,((\lambda y.\, (1+\varepsilon))(2+\varepsilon))) \\ &= E\,((1+\varepsilon) \cdot E\,(1+\varepsilon)) \\ &= E\,((1+\varepsilon) \cdot 1) = E\,(1+\varepsilon) = 1 \neq 0. \end{aligned}$$

As explained in [12], the root of this error is our failure to distinguish between the perturbations introduced by the inner and outer invocations of D. There are ways to solve this problem, for instance by tagging perturbations with a fresh ε every time D is invoked and incurring the bookkeeping overhead of keeping track of which ε is associated with which invocation of D. Nonetheless, we hope the example serves to illustrate the value and nontriviality of a clear semantics for a λ-calculus with AD.

What do we mean when we say that symbolic differentiation suffers from the loss of sharing? And how does forward AD fix this problem? Let us illustrate with an example. Consider the problem of computing the derivative of a product of n functions: $f(x) = f_1(x) \cdot f_2(x) \cdot \ldots \cdot f_n(x)$. Applying the product rule, we arrive at the expression for the derivative, which has size quadratic in n:

$$f'(x) = f_1'(x) \cdot f_2(x) \cdot \ldots \cdot f_n(x) + f_1(x) \cdot f_2'(x) \cdot \ldots \cdot f_n(x) + \cdots + f_1(x) \cdot f_2(x) \cdot \ldots \cdot f_n'(x).$$

Evaluating it naively would result in evaluating each $f_i(x)$ $n-1$ times. If our cost model is that evaluating $f_i(x)$ or $f_i'(x)$ each cost 1, and the arithmetic operations are free, then $f(x)$ has a cost of n, whereas $f'(x)$ has a cost of n^2. Contrast this with forward AD: the pushforward $Tf(x + x'\varepsilon)$ is the product (in the sense of dual numbers) of the pushforwards $Tf_1(x + x'\varepsilon)$, $Tf_2(x + x'\varepsilon)$, $\ldots$, $Tf_n(x + x'\varepsilon)$. Evaluating each $Tf_i(x+x'\varepsilon)$ amounts to evaluating $f_i(x)$ and $f_i'(x)$, and hence has a cost of 2. Therefore, the cost of $Tf(x+\varepsilon)$ is $2n$. In general, forward AD guarantees that evaluating f' takes no more than a constant factor times as many operations as evaluating f.

3 Tangent Bundles in Differential λ-Categories

The definition of the interpretation of the perturbative λ-calculus as well as the proof of its soundness rely on the theory of tangent bundles in differential λ-categories

developed in [9]. To make this paper self-contained, we summarize the results of that theory here. The reader is referred to [9] for more details.

Let $\mathbf{C}$ be a cartesian category and X, Y, Z objects of $\mathbf{C}$. We denote by $X \times Y$ the product of X and Y and by $\pi_1 : X \times Y \to X$, $\pi_2 : X \times Y \to Y$ the projections. The terminal object is denoted by $\mathbb{1}$, and for any object X, we denote by $!_X$ the unique morphism from X to $\mathbb{1}$. For a pair of morphisms $f : Z \to X$ and $g : Z \to Y$, denote by $\langle f, g\rangle : Z \to X \times Y$ the pairing of f and g, i.e., the unique morphism such that $\pi_1 \circ \langle f, g\rangle = f$ and $\pi_2 \circ \langle f, g\rangle = g$.

A cartesian category $\mathbf{C}$ is called *closed* if for any pair of objects X and Y of $\mathbf{C}$ there exists an object $X \Rightarrow Y$, called the *exponential object*, and a morphism $\mathrm{ev}_{X,Y} : (X \Rightarrow Y) \times X \to Y$, called the *evaluation morphism*, satisfying the following universal property: the map $\Lambda^{\text{-}} : \mathbf{C}(Z, X \Rightarrow Y) \to \mathbf{C}(Z \times X, Y)$ given by $\Lambda^{\text{-}}(g) = \mathrm{ev}_{X,Y} \circ (g \times \mathrm{id}_X)$ is bijective. Let $\Lambda : \mathbf{C}(Z \times X, Y) \to \mathbf{C}(Z, X \Rightarrow Y)$ denote the inverse of $\Lambda^{\text{-}}$. In other words, for a morphism $f : Z \times X \to Y$, $\Lambda(f) : Z \to X \Rightarrow Y$ is the unique morphism such that $\mathrm{ev}_{X,Y} \circ (\Lambda(f) \times \mathrm{id}_X) = f$. The morphism $\Lambda(f)$ is called the *currying* of f. We shall make use of the equations

$$\Lambda(f) \circ g = \Lambda(f \circ (g \times \mathrm{id})), \tag{1}$$

$$\mathrm{ev} \circ \langle \Lambda(f), g\rangle = f \circ \langle \mathrm{id}, g\rangle, \tag{2}$$

which follow immediately from the definition of Λ.

The notion of cartesian differential category was introduced by Blute et al. [2] as an axiomatization of differentiable maps as well as a unifying framework in which to study different notions reminiscent of the differential calculus.

Definition 3.1 ([2, Definition 1.1.1]) A category $\mathbf{C}$ is *left-additive* if each hom-set is equipped with the structure of a commutative monoid $(\mathbf{C}(X, Y), +, 0)$ such that $(g + h) \circ f = (g \circ f) + (h \circ f)$ and $0 \circ f = 0$. A morphism f in $\mathbf{C}$ is *additive* if it satisfies $f \circ (g + h) = (f \circ g) + (f \circ h)$ and $f \circ 0 = 0$.

Definition 3.2 ([2, Definition 1.2.1]) A category is *cartesian left-additive* if it is a left-additive category with products such that all projections are additive, and all pairings of additive morphisms are additive.

Remark 3.3 Let $\mathbf{C}$ be a cartesian left-additive category. Then the pairing map $\langle -, -\rangle : \mathbf{C}(Z, X) \times \mathbf{C}(Z, Y) \to \mathbf{C}(Z, X \times Y)$ is additive: $\langle f+g, h+k\rangle = \langle f, h\rangle + \langle g, k\rangle$ and $\langle 0, 0\rangle = 0$. Furthermore, for each pair of objects X and Y there are morphisms $\iota_1 \stackrel{\text{def}}{=} \langle \mathrm{id}_X, 0\rangle : X \to X \times Y$ and $\iota_2 \stackrel{\text{def}}{=} \langle 0, \mathrm{id}_Y\rangle : Y \to X \times Y$, which satisfy the equations $\pi_1 \circ \iota_1 = \mathrm{id}_X$, $\pi_2 \circ \iota_2 = \mathrm{id}_Y$, $\pi_k \circ \iota_l = 0$ if $k \neq l$, and $\iota_1 \circ \pi_1 + \iota_2 \circ \pi_2 = \mathrm{id}_{X \times Y}$. Note, however, that in contrast with additive categories, in a left-additive category these equations do not imply that the morphisms ι_1 and ι_2 equip $X \times Y$ with the structure of a coproduct of X and Y. This is so on the subcategory of additive morphisms, but not necessarily on the full category.

Definition 3.4 ([2, Section 1.4, Definition 2.1.1], [3, Definition 4.2]) A cartesian closed category is *cartesian closed left-additive* if it is a cartesian left-additive category such that each currying map $\Lambda : \mathbf{C}(Z \times X, Y) \to \mathbf{C}(Z, X \Rightarrow Y)$

is additive: $\Lambda(f+g) = \Lambda(f) + \Lambda(g)$ and $\Lambda(0) = 0$. A *cartesian (closed) differential category* is a cartesian (closed) left-additive category equipped with an operator $D : \mathbf{C}(X, Y) \to \mathbf{C}(X \times X, Y)$ satisfying the following axioms:

D1. $D(f+g) = D(f) + D(g)$ and $D(0) = 0$.

D2. $D(f) \circ \langle h + k, v\rangle = D(f) \circ \langle h, v\rangle + D(f) \circ \langle k, v\rangle$ and $D(f) \circ \langle 0, v\rangle = 0$.

D3. $D(\mathrm{id}) = \pi_1$, $D(\pi_1) = \pi_1 \circ \pi_1$, $D(\pi_2) = \pi_2 \circ \pi_1$.

D4. $D(\langle f, g\rangle) = \langle D(f), D(g)\rangle$.

D5. $D(f \circ g) = D(f) \circ \langle D(g), g \circ \pi_2\rangle$.

D6. $D(D(f)) \circ \langle\langle g, 0\rangle, \langle h, k\rangle\rangle = D(f) \circ \langle g, k\rangle$.

D7. $D(D(f)) \circ \langle\langle 0, h\rangle, \langle g, k\rangle\rangle = D(D(f)) \circ \langle\langle 0, g\rangle, \langle h, k\rangle\rangle$.

The paradigmatic example of a cartesian differential category is the category of smooth maps, whose objects are natural numbers and morphisms $m \to n$ are smooth maps $\mathbb{R}^m \to \mathbb{R}^n$. The operator D takes an $f : \mathbb{R}^m \to \mathbb{R}^n$ and produces a $D(f) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^n$ given by $D(f)(x', x) = J_f(x) \cdot x'$, where $J_f(x)$ is the Jacobian of f at the point x.

Some intuition for the axioms: D1 says D is additive; D2 that $D(f)$ is additive in its first coordinate; D3 and D4 assert that D is compatible with the product structure, and D5 is the chain rule. We refer the reader to [2, Lemma 2.2.2] for the proof that D6 is essentially requiring that $D(f)$ be linear (in the sense defined below) in its first variable. D7 is essentially independence of order of partial differentiation.

Following [2], we say that a morphism f is *linear* if $D(f) = f \circ \pi_1$. By [2, Lemma 2.2.2], the class of linear morphisms is closed under sum, composition, pairing, and product, and contains all identities, projections, and zero morphisms. Also, axiom D2 implies that any linear morphism is additive.

The notion of cartesian differential category was partly motivated by a desire to model the differential λ-calculus of Erhard and Regnier [4] categorically. Blute et al. [2] proved that cartesian differential categories are sound and complete to model suitable term calculi. However, the properties of cartesian differential categories are too weak for modeling the full differential λ-calculus because the differential operator is not necessarily compatible with the cartesian closed structure. For this reason, Bucciarelli et al. [3] introduced the notion of differential λ-category.

Definition 3.5 ([3, Definition 4.4]) A *differential λ-category* is a cartesian closed differential category such that $D(\Lambda(f)) = \Lambda(D(f) \circ \langle \pi_1 \times 0_X, \pi_2 \times \mathrm{id}_X\rangle)$ holds, for each $f : Z \times X \to Y$.

This differential λ-category axiom is essentially requiring that the evaluation morphism ev be linear in its first argument.

Example 3.6 Blute et al. [1] proved that the category of convenient vector spaces and smooth maps is a cartesian closed differential category. We have shown in [9] that it is in fact a differential λ-category. We refer the reader to [3] for two other examples of differential λ-categories.

The differential operator D allows us to replicate the construction of the tangent bundle of a smooth manifold from differential geometry in any cartesian differential category. Let $\mathbf{C}$ be a cartesian differential category. The *tangent bundle functor* $T : \mathbf{C} \to \mathbf{C}$ is defined by $TX = X \times X$ and $T(f) = \langle D(f), f \circ \pi_2 \rangle$.

Lemma 3.7 ([9, Lemma 3.1.3]) *If f is linear, then $T(f) = f \times f$.*

The tangent bundle functor T is part of a monad. For each object X of $\mathbf{C}$, the unit η_X of the monad is the morphism $\langle 0, \mathrm{id}_X \rangle : X \to X \times X = TX$, and the multiplication μ_X of the monad is the morphism $\langle \pi_2 \circ \pi_1 + \pi_1 \circ \pi_2, \pi_2 \circ \pi_2 \rangle : TTX = (X \times X) \times (X \times X) \to X \times X = TX$. The monad (T, η, μ) is strong [10, Definition 3.2]. The tensorial strength $t_{X,Y} : X \times TY \to T(X \times Y)$, also called the right tensorial strength, is given by $t_{X,Y} = \langle \langle 0, \pi_1 \circ \pi_2 \rangle, \langle \pi_1, \pi_2 \circ \pi_2 \rangle \rangle = \langle 0 \times \pi_1, \mathrm{id}_X \times \pi_2 \rangle$. Using the symmetry $c_{A,B} = \langle \pi_2, \pi_1 \rangle : A \times B \to B \times A$ of $\mathbf{C}$, we may also define the left tensorial strength by $t'_{X,Y} = T(c_{Y,X}) \circ t_{Y,X} \circ c_{TX,Y} : TX \times Y \to T(X \times Y)$. More explicitly, $t'_{X,Y} = \langle \langle \pi_1 \circ \pi_1, 0 \rangle, \langle \pi_2 \circ \pi_1, \pi_2 \rangle \rangle = \langle \pi_1 \times 0, \pi_2 \times \mathrm{id}_Y \rangle$.

Because the functor T is part of a strong monad, by [7, Theorem 2.1] T becomes a monoidal functor $(T, \psi, \psi^0) : \mathbf{C} \to \mathbf{C}$ if we put $\psi_{X,Y}$ equal to the composite $\psi_{X,Y} = \mu_{X \times Y} \circ T(t_{X,Y}) \circ t'_{X,TY} : TX \times TY \to T(X \times Y)$ and by putting $\psi^0 = \eta_{\mathbb{1}} : \mathbb{1} \to T\mathbb{1}$. The definition of $\psi_{X,Y}$ is asymmetric, and indeed there is also a morphism $\tilde{\psi}_{X,Y} = \mu_{X \times Y} \circ T(t'_{X,Y}) \circ t_{TX,Y} : TX \times TY \to T(X \times Y)$ that also makes T into a monoidal functor. The morphisms ψ and $\tilde{\psi}$ can be computed explicitly. It is shown in [9, Lemmas 3.4.1, 3.4.2] that ψ and $\tilde{\psi}$ are equal and coincide with the distributivity isomorphism σ. In particular, (T, η, μ) is a commutative monad [7, Definition 3.1].

Lemma 3.8 ([9, Lemma 3.4.5]) *Let $f : Z \to X$, $g : Z \to Y$ be morphisms in $\mathbf{C}$. Then $T\langle f, g \rangle = \psi \circ \langle T(f), T(g) \rangle$.*

From now on we assume that $\mathbf{C}$ is a differential λ-category.

Proposition 3.9 ([9, Proposition 3.6.2]) *Let $g : A \times B \to C$ be a morphism in $\mathbf{C}$. Let $h = T(g) \circ t' : TA \times B \to TC$. Then $T(\Lambda(g)) = \langle \Lambda(\pi_1 \circ h), \Lambda(\pi_2 \circ h) \rangle : TA \to T(B \Rightarrow C)$.*

The cartesian closed category $\mathbf{C}$ is a symmetric monoidal closed category, with the monoidal structure given by product, and hence also a closed category of Eilenberg and Kelly [5]. By [5, Proposition 4.3], the monoidal functor $(T, \psi, \psi^0) : \mathbf{C} \to \mathbf{C}$ gives rise to a closed functor $(T, \hat{\psi}, \psi^0) : \mathbf{C} \to \mathbf{C}$, where $\hat{\psi} = \hat{\psi}_{X,Y} : T(X \Rightarrow Y) \to (TX \Rightarrow TY)$ is given by $\hat{\psi} = \Lambda(T(\mathrm{ev}) \circ \psi)$. We have shown in [9, Section 3.7] that

$$T(\mathrm{ev}) \circ \psi = \langle \mathrm{ev} \circ (\pi_1 \times \pi_2) + D(\mathrm{ev}) \circ t \circ (\pi_2 \times \mathrm{id}), \mathrm{ev} \circ (\pi_2 \times \pi_2) \rangle : T(X \Rightarrow Y) \times TX \to TY. \tag{3}$$

The cartesian closed category $\mathbf{C}$ gives rise to a category $\underline{\mathbf{C}}$ enriched in $\mathbf{C}$: the objects of $\underline{\mathbf{C}}$ are the objects of $\mathbf{C}$, and $\underline{\mathbf{C}}(X, Y) = X \Rightarrow Y$, for each pair of object X and Y of $\mathbf{C}$. By [8, Theorem 1.3], the functor $T : \mathbf{C} \to \mathbf{C}$ equipped with the

tensorial strength t gives rise to a $\mathbf{C}$-functor $\underline{T} : \underline{\mathbf{C}} \to \underline{\mathbf{C}}$ such that $\underline{T}X = TX$ and $\underline{T} = \underline{T}_{X,Y} : (X \Rightarrow Y) \to (TX \Rightarrow TY)$ is given by $\underline{T} = \Lambda(T(\mathrm{ev}) \circ t)$.

Theorem 3.10 ([9, Theorem 3.8.2]) *$\underline{T}$ is a linear morphism.*

Proposition 3.11 ([9, Proposition 3.8.3]) *Let $f : Z \times X \to Y$ be a morphism in $\mathbf{C}$. Then $\underline{T} \circ \Lambda(f) = \Lambda(T(f) \circ t)$.*

We shall also need the following definition and easy lemma.

Definition 3.12 ([3, Definition 4.6]) Let $\mathrm{sw} = \mathrm{sw}_{X,Y,Z}$ denote the morphism $\mathrm{sw} = \langle\langle \pi_1 \circ \pi_1, \pi_2 \rangle, \pi_2 \circ \pi_1 \rangle : (X \times Y) \times Z \to (X \times Z) \times Y$. Clearly, sw is a linear morphism. Furthermore, $\mathrm{sw} \circ \langle\langle f, g \rangle, h \rangle = \langle\langle f, h \rangle, g \rangle$.

Lemma 3.13 $T(\mathrm{sw}) \circ t' \circ (t \times \mathrm{id}) = t \circ \mathrm{sw} : (X \times TZ) \times Y \to T((X \times Y) \times Z)$.

4 Perturbative λ-Calculus

In this section we describe the perturbative λ-calculus. Its syntax is very similar to the syntax of the differential λ-calculus of Ehrhard and Regnier [4] with two notable differences:

- Instead of introducing a syntactic form $\mathsf{D}\, s \cdot t$ denoting the derivative of s in direction t, we extend the λ-calculus with a syntactic form $\mathsf{T}\, s$ denoting the pushforward of s.
- To syntactically enforce the fact that pairs are linear, instead of introducing a syntax for pairs, we introduce two syntactic forms: $\iota_k s$ (injection of s into the kth factor of a product) and $\pi_k s$ (projection of s onto the kth coordinate), $k = 1, 2$. With the syntax for injections, pairs can be introduced as syntactic sugar and linearity is automatic.

More formally, the set Λ^p of *perturbative λ-terms* and the set Λ^s of *simple λ-terms* are defined by mutual induction as follows:

$$\begin{aligned} \Lambda^p &: \quad M, N ::= 0 \mid s \mid s + N, \\ \Lambda^s &: \quad\quad s, t ::= x \mid \lambda x.\, s \mid sN \mid \mathsf{T}\, s \mid \iota_k s \mid \pi_k s, \quad k = 1, 2, \end{aligned}$$

Thus, every perturbative λ-term is a formal sum of a finite (possibly empty) bag of simple λ-terms. The operation $+$ is extended to arbitrary perturbative λ-terms in the obvious way as the union of bags. The term 0 denoting the empty bag is the neutral element of the sum. We consider perturbative λ-terms up to α-conversion, associativity and commutativity of the sum. We write $M \equiv N$ if M and N are syntactically equal up to the aforementioned equivalences. The term $\mathsf{T}\, s$ is the *pushforward* of s. The set $\mathrm{FV}(M)$ of free variables of M is defined as usual. We

$$\frac{\Gamma(x) = \sigma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma; x : \sigma \vdash s : \tau}{\Gamma \vdash \lambda x.\, s : \sigma \to \tau} \qquad \frac{\Gamma \vdash s : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash sN : \tau}$$

$$\frac{}{\Gamma \vdash 0 : \sigma} \qquad \frac{\Gamma \vdash s : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash s + N : \sigma} \qquad \frac{\Gamma \vdash s : \sigma_k}{\Gamma \vdash \iota_k s : \sigma_1 \times \sigma_2} \qquad \frac{\Gamma \vdash s : \sigma_1 \times \sigma_2}{\Gamma \vdash \pi_k s : \sigma_k}$$

$$\frac{\Gamma \vdash s : \sigma \to \tau}{\Gamma \vdash \mathsf{T}\, s : \mathsf{T}\,\sigma \to \mathsf{T}\,\tau}$$

Fig. 1. Typing rules of the perturbative λ-calculus

introduce the following syntactic sugar:

$$\lambda x.\, \Big(\sum\nolimits_{i=1}^{n} s_i\Big) \stackrel{\text{def}}{=} \sum\nolimits_{i=1}^{n} \lambda x.\, s_i, \qquad \iota_k\Big(\sum\nolimits_{i=1}^{n} s_i\Big) \stackrel{\text{def}}{=} \sum\nolimits_{i=1}^{n} \iota_k s_i,$$
$$\mathsf{T}\Big(\sum\nolimits_{i=1}^{n} s_i\Big) \stackrel{\text{def}}{=} \sum\nolimits_{i=1}^{n} \mathsf{T}\, s_i, \qquad \pi_k\Big(\sum\nolimits_{i=1}^{n} s_i\Big) \stackrel{\text{def}}{=} \sum\nolimits_{i=1}^{n} \pi_k s_i,$$
$$\Big(\sum\nolimits_{i=1}^{n} s_i\Big) N \stackrel{\text{def}}{=} \sum\nolimits_{i=1}^{n} s_i N, \qquad \langle M_1, M_2 \rangle \stackrel{\text{def}}{=} \iota_1 M_1 + \iota_2 M_2,$$

where the sums reduce to the term 0 if $n = 0$. Note that the terms in the left hand sides of the above equations are not valid terms in the language. They are *abbreviations* for the respective terms in the right hand sides of the equations. Note also that this way we syntactically capture the linearity of abstractions, pushforwards, injections, projections, pairs, and applications in the operator position.

Let us define the type system that characterizes the simply typed perturbative λ-calculus. We assume that we are given some atomic types α, β, ..., and if σ and τ are types, then so are $\sigma \times \tau$ and $\sigma \to \tau$. We define[5] a type function T by $\mathsf{T}\sigma = \sigma \times \sigma$. The typing rules are shown in Figure 1. They are the standard typing rules of the simply typed λ-calculus with products, extended by the typing rules for T and the sum.

Let M, P be perturbative λ-terms and x a variable. The *substitution* of P for x in M, denoted by $M[P/x]$, is defined by induction on M as shown in Figure 2. The usual caveat applies to the definition of $(\lambda y.\, s)[P/x]$, namely that by α-conversion we may assume that $x \neq y$ and $y \notin \mathrm{FV}(P)$.

We impose the reduction rules listed in Figure 3: the usual β-reduction $(\lambda x.\, s)N \to_\beta s[N/x]$ and projection-injection rules $\pi_k(\iota_k s) \to_\times s$ and $\pi_k(\iota_l s) \to_\times 0$, where $k, l \in \{1, 2\}$, $k \neq l$. There is also a reduction rule for T, which is similar in shape to the reduction rule for D in the differential λ-calculus, but is suggested by the general categorical properties of tangent bundles in differential λ-categories [9]. More specifically, the reduction rule for T is $\mathsf{T}(\lambda x.\, s) \to_{\mathsf{T}} \lambda x.\, \mathsf{T}_x\, s$, where $\mathsf{T}_x\, M$ is defined by induction on M according to the equations shown in Figure 4. The definition of $\mathsf{T}_x(\lambda y.\, s)$ is subject to the standard side condition, namely that by

[5] The definition of T is motivated by our desire to interpret the perturbative λ-calculus in differential λ-categories, for example, the category of convenient vector spaces and smooth maps. The tangent bundle of a convenient vector space E is isomorphic to $E \times E$, which we reflect in the calculus by defining $\mathsf{T}\sigma$ to be $\sigma \times \sigma$. It would be interesting to generalize T to allow types σ to be general smooth spaces and $\mathsf{T}\sigma$ the tangent bundle, which would not in general be definable as $\sigma \times \sigma$. We leave this for future work.

$$y[P/x] = \begin{cases} P & \text{if } x = y, \\ y & \text{otherwise,} \end{cases} \qquad 0[P/x] = 0,$$

$$(\lambda y.\, s)[P/x] = \lambda y.\, (s[P/x]), \qquad (s + N)[P/x] = s[P/x] + N[P/x],$$
$$(sN)[P/x] = (s[P/x])(N[P/x]), \qquad (\iota_k s)[P/x] = \iota_k(s[P/x]),$$
$$(\mathsf{T}\, s)[P/x] = \mathsf{T}(s[P/x]), \qquad (\pi_k s)[P/x] = \pi_k(s[P/x]).$$

Fig. 2. Definition of substitution

$$(\lambda x.\, s)N \to_\beta s[N/x] \qquad \pi_k(\iota_k s) \to_\times s \qquad \pi_k(\iota_l s) \to_\times 0 \qquad \mathsf{T}(\lambda x.\, s) \to_{\mathsf{T}} \lambda x.\, \mathsf{T}_x\, s$$

Fig. 3. Reduction rules of the perturbative λ-calculus

$$\mathsf{T}_x\, y = \begin{cases} x & \text{if } x = y \\ \iota_2 y & \text{otherwise} \end{cases} \qquad \mathsf{T}_x\, 0 = 0$$

$$\mathsf{T}_x(\lambda y.\, s) = \langle \lambda y.\, \pi_1(\mathsf{T}_x\, s), \lambda y.\, \pi_2(\mathsf{T}_x\, s) \rangle \qquad \mathsf{T}_x(s + N) = \mathsf{T}_x\, s + \mathsf{T}_x\, N$$
$$\mathsf{T}_x(sN) = (\mathsf{T}_x\, s) \diamond (\mathsf{T}_x\, N) \qquad \mathsf{T}_x(\iota_k s) = \langle \iota_k(\pi_1(\mathsf{T}_x\, s)), \iota_k(\pi_2(\mathsf{T}_x\, s)) \rangle$$
$$\mathsf{T}_x(\mathsf{T}\, s) = \langle \mathsf{T}\, (\pi_1(\mathsf{T}_x\, s)), \mathsf{T}\, (\pi_2(\mathsf{T}_x\, s)) \rangle \qquad \mathsf{T}_x(\pi_k s) = \langle \pi_k(\pi_1(\mathsf{T}_x\, s)), \pi_k(\pi_2(\mathsf{T}_x\, s)) \rangle$$

$$M \diamond N \stackrel{\text{def}}{=} \langle (\pi_1 M)(\pi_2 N) + \pi_1((\mathsf{T}\, (\pi_2 M))\, N), (\pi_2 M)(\pi_2 N) \rangle$$

Fig. 4. Definition of T_x

α-conversion we may assume that x is different from y. To simplify the notation, we have introduced some syntactic sugar: the operation $\diamond$, which is also defined in Figure 4. The typing rule for $\diamond$ can be derived from the typing rules for other syntactic forms and reads as follows:

$$\frac{\Gamma \vdash M : \mathsf{T}(\sigma \to \tau) \quad \Gamma \vdash N : \mathsf{T}\, \sigma}{\Gamma \vdash M \diamond N : \mathsf{T}\, \tau}$$

Let us provide some intuition for these equations. We define $\mathsf{T}_x\, x = x$ because the pushforward of the identity function is the identity function; similarly, we set $\mathsf{T}_x\, y = \iota_2 y$ if $x \neq y$ because the pushforward of the constant function is a constant function returning the lifted value. The definitions of T_x over abstractions and applications are suggested by Proposition 3.9 and equation (3), respectively. The definitions of T_x over T, π_k, and ι_k have the given shape because $\underline{T}$, π_k, and ι_k are linear morphisms ($\underline{T}$ is linear by Theorem 3.10). Finally, the definition of T_x over sums follows from the additivity of the tangent bundle functor.

Denote by $\to$ the contextual closure of $\to_\beta \cup \to_\times \cup \to_{\mathsf{T}}$.

Lemma 4.1 *The type system of the simply typed perturbative λ-calculus satisfies*

$$\frac{\Gamma \vdash (\lambda x.\, s)N : \tau}{\Gamma \vdash (\lambda x.\, s)N = s[N/x] : \tau}\ (\beta) \qquad \frac{\Gamma \vdash \mathsf{T}(\lambda x.\, s) : \mathsf{T}\,\sigma \to \mathsf{T}\,\tau}{\Gamma \vdash \mathsf{T}(\lambda x.\, s) = \lambda x.\, \mathsf{T}_x\, s : \mathsf{T}\,\sigma \to \mathsf{T}\,\tau}\ (\mathsf{T})$$

$$\frac{\Gamma; x : \sigma \vdash s = t : \tau}{\Gamma \vdash \lambda x.\, s = \lambda x.\, t : \sigma \to \tau}\ (\xi) \qquad \frac{\Gamma \vdash s_1 = s_2 : \sigma \to \tau \quad \Gamma \vdash N_1 = N_2 : \sigma}{\Gamma \vdash s_1 N_1 = s_2 N_2 : \tau}\ (Ap)$$

$$\frac{\Gamma \vdash s = t : \sigma \quad x : \tau \notin \Gamma}{\Gamma; x : \tau \vdash s = t : \sigma}\ (W) \qquad \frac{\Gamma \vdash s = t : \sigma \to \tau}{\Gamma \vdash \mathsf{T}\,s = \mathsf{T}\,t : \mathsf{T}\,\sigma \to \mathsf{T}\,\tau}\ (\mathsf{T}Ap)$$

$$\frac{\Gamma \vdash s = t : \sigma_1 \times \sigma_2}{\Gamma \vdash \pi_k s = \pi_k t : \sigma_k}\ (\pi) \qquad \frac{\Gamma \vdash s_1 = s_2 : \sigma \quad \Gamma \vdash N_1 = N_2 : \sigma}{\Gamma \vdash s_1 + N_1 = s_2 + N_2 : \sigma}\ (Sum)$$

$$\frac{\Gamma \vdash s = t : \sigma_k}{\Gamma \vdash \iota_k s = \iota_k t : \sigma_1 \times \sigma_2}\ (\iota) \qquad \frac{\Gamma \vdash s : \sigma}{\Gamma \vdash \pi_k(\iota_k s) = s : \sigma} \quad \frac{\Gamma \vdash s : \sigma}{\Gamma \vdash \pi_k(\iota_l s) = 0 : \sigma}\ (\pi\iota)$$

Fig. 5. Perturbative λ-theory rules

the following properties.

(a) If $\Gamma; x : \sigma \vdash M : \tau$ *and* $\Gamma \vdash N : \sigma$, *then* $\Gamma \vdash M[N/x] : \tau$.

(b) If $\Gamma; x : \sigma \vdash M : \tau$, *then* $\Gamma; x : \mathsf{T}\,\sigma \vdash \mathsf{T}_x\, M : \mathsf{T}\,\tau$.

(c) If $\Gamma \vdash \pi_k(\iota_k N) : \sigma$, *then* $\Gamma \vdash N : \sigma$.

(d) If $\Gamma \vdash N : \sigma$ *and* $N \to N'$, *then* $\Gamma \vdash N' : \sigma$.

Proof. (a) and (b) follow by straightforward induction on the length of the proof of the corresponding typing judgment. (c) is obvious. (d) follows from (a), (b), and (c) because type derivations are contextual. □

To facilitate the proof that applying the reduction rules preserves the meaning, which is what soundness really means, we introduce the notion of perturbative λ-theory. Let $\mathcal{T}$ be a collection of judgments of the shape $\Gamma \vdash M = N : \sigma$ such that $\Gamma \vdash M : \sigma$ and $\Gamma \vdash N : \sigma$. $\mathcal{T}$ is called a *perturbative λ-theory* if it is closed under the rules shown in Figure 5 (where $k, l \in \{1, 2\}$ and $k \neq l$) together with the obvious rules for reflexivity, transitivity, and symmetry. Equations (β), (T), $(\pi\iota)$ are imposed by the reduction rules. The remaining rules say that the equality relation is contextual.

5 Categorical Semantics

In this section we show that, like the simply typed variant of the differential λ-calculus of Ehrhard and Regnier [4], the simply typed perturbative λ-calculus can be modeled by differential λ-categories of Bucciarelli et al. [3]. We have shown in [9] that an arbitrary differential λ-category is equipped with a canonical pushforward construction. The perturbative λ-calculus captures the pushforward construction as a syntactic operation.

Let $\mathbf{C}$ be a differential λ-category. Let us define the interpretation of the simply typed perturbative λ-calculus from the previous section in the category $\mathbf{C}$. The definition is similar to the interpretation of the simply typed differential λ-calculus

in a differential λ-category defined in [3]. Types are interpreted as follows: $|\alpha| = A$, for some object A, $|\sigma \times \tau| = |\sigma| \times |\tau|$, and $|\sigma \to \tau| = |\sigma| \Rightarrow |\tau|$. Contexts are interpreted as usual: $|\emptyset| = \mathbb{1}$ and $|\Gamma; x : \sigma| = |\Gamma| \times |\sigma|$. The interpretation of a judgment $\Gamma \vdash M : \sigma$ will be a morphism from $|\Gamma|$ to $|\sigma|$ denoted by $|M^\sigma|_\Gamma$ and defined inductively as follows:

- $|x^\sigma|_{\Gamma;x:\sigma} = \pi_2 : |\Gamma| \times |\sigma| \to |\sigma|$;
- $|y^\tau|_{\Gamma;x:\sigma} = |y^\tau|_\Gamma \circ \pi_1 : |\Gamma| \times |\sigma| \to |\tau|$ for $x \neq y$;
- $|(sN)^\tau|_\Gamma = \mathrm{ev} \circ \langle |s^{\sigma\to\tau}|_\Gamma, |N^\sigma|_\Gamma \rangle : |\Gamma| \to |\tau|$;
- $|(\lambda x.\, s)^{\sigma\to\tau}|_\Gamma = \Lambda(|s^\tau|_{\Gamma;x:\sigma}) : |\Gamma| \to |\sigma| \Rightarrow |\tau|$;
- $|(\mathsf{T}\, s)^{\mathsf{T}\,\sigma\to\mathsf{T}\,\tau}|_\Gamma = \underline{T}_{|\sigma|,|\tau|} \circ |s^{\sigma\to\tau}|_\Gamma : |\Gamma| \to T|\sigma| \Rightarrow T|\tau|$;
- $|0^\sigma|_\Gamma = 0 : |\Gamma| \to |\sigma|$;
- $|(s + N)^\sigma|_\Gamma = |s^\sigma|_\Gamma + |N^\sigma|_\Gamma : |\Gamma| \to |\sigma|$;
- $|(\iota_k s)^{\sigma_1\times\sigma_2}|_\Gamma = \iota_k \circ |s^{\sigma_k}|_\Gamma : |\Gamma| \to |\sigma_1| \times |\sigma_2|$, $k = 1, 2$;
- $|(\pi_k s)^{\sigma_k}|_\Gamma = \pi_k \circ |s^{\sigma_1\times\sigma_2}|_\Gamma : |\Gamma| \to |\sigma_k|$, $k = 1, 2$.

The only interesting case is that of $\mathsf{T}\, s$, which is what we were really after; the other cases are standard. It follows from the definitions that $|\langle M, N\rangle|_\Gamma = \langle |M|_\Gamma, |N|_\Gamma \rangle$. We shall omit the superscript σ in $|M^\sigma|_\Gamma$ when there is no risk of confusion.

Given a differential λ-category $\mathbf{C}$, we define the *theory of* $\mathbf{C}$ by

$$\mathrm{Th}(\mathbf{C}) = \{\Gamma \vdash M = T : \sigma \mid \Gamma \vdash M : \sigma, \quad \Gamma \vdash N : \sigma, \quad |M^\sigma|_\Gamma = |N^\sigma|_\Gamma\}.$$

We are going to prove that the interpretation $|\cdot|$ is *sound* for the simply typed perturbative λ-calculus, i.e., that $\mathrm{Th}(\mathbf{C})$ is a perturbative λ-theory. We begin by proving some lemmas. Lemmas 5.1 and 5.2 are standard.

Lemma 5.1 $|M|_{\Gamma;x:\sigma;y:\tau} = |M|_{\Gamma;y:\tau;x:\sigma} \circ \mathrm{sw}$.

Lemma 5.2 *If* $\Gamma \vdash M : \tau$ *and* $x \notin \mathrm{FV}(M)$, *then* $|M^\tau|_{\Gamma;x:\sigma} = |M^\tau|_\Gamma \circ \pi_1$.

Lemma 5.3 *Let* $\Gamma \vdash M : \mathsf{T}(\sigma \to \tau)$ *and* $\Gamma \vdash N : \mathsf{T}\,\sigma$. *Then*

$$|(M \diamond N)^{\mathsf{T}\,\tau}|_\Gamma = T(\mathrm{ev}) \circ \psi \circ \langle |M^{\mathsf{T}(\sigma\to\tau)}|_\Gamma, |N^{\mathsf{T}\,\sigma}|_\Gamma \rangle.$$

Proof. Expanding $M \diamond N$ and applying the definition of the interpretation, we find that $|M \diamond N|_\Gamma$ is equal to

$$\begin{aligned} &\langle\, \mathrm{ev} \circ \langle \pi_1 \circ |M|_\Gamma, \pi_2 \circ |N|_\Gamma \rangle + \pi_1 \circ \mathrm{ev} \circ \langle \underline{T} \circ \pi_2 \circ |M|_\Gamma, |N|_\Gamma \rangle, \\ &\quad \mathrm{ev} \circ \langle \pi_2 \circ |M|_\Gamma, \pi_2 \circ |N|_\Gamma \rangle \rangle \\ &= \langle\, \mathrm{ev} \circ (\pi_1 \times \pi_2) + \pi_1 \circ \mathrm{ev} \circ (\underline{T} \circ \pi_2 \times \mathrm{id}), \mathrm{ev} \circ (\pi_2 \times \pi_2) \rangle \circ \langle |M|_\Gamma, |N|_\Gamma \rangle. \end{aligned}$$

The summand $\pi_1 \circ \mathrm{ev} \circ (\underline{T} \circ \pi_2 \times \mathrm{id})$ is equal to $\pi_1 \circ T(\mathrm{ev}) \circ t \circ (\pi_2 \times \mathrm{id}) = D(\mathrm{ev}) \circ t \circ (\pi_2 \times \mathrm{id})$ by the definitions of $\underline{T}$ and T. We conclude that $|M \diamond N|_\Gamma$ is equal to $\langle \mathrm{ev} \circ (\pi_1 \times \pi_2) + D(\mathrm{ev}) \circ t \circ (\pi_2 \times \mathrm{id}), \mathrm{ev} \circ (\pi_2 \times \pi_2) \rangle \circ \langle |M|_\Gamma, |N|_\Gamma \rangle$, which coincides with $T(\mathrm{ev}) \circ \psi \circ \langle |M|_\Gamma, |N|_\Gamma \rangle$ by equation (3). □

Lemma 5.4 (Substitution) *Let* $\Gamma; x : \sigma \vdash M : \tau$ *and* $\Gamma \vdash P : \sigma$. *Then* $|(M[P/x])^\tau|_\Gamma = |M^\tau|_{\Gamma;x:\sigma} \circ \langle \mathrm{id}_{|\Gamma|}, |P^\sigma|_\Gamma \rangle$.

Proof. The proof is by induction on M. The only new cases are $M \equiv \mathsf{T}\, s$, $M \equiv \iota_k s$, and $M \equiv \pi_k s$. However, they are straightforward. For example, $|(\mathsf{T}\, s)[P/x]|_\Gamma = |\,\mathsf{T}(s[P/x])|_\Gamma = \underline{T} \circ |s[P/x]|_\Gamma$ by the definitions of substitution and $|\cdot|$, which is equal to $\underline{T} \circ |s|_{\Gamma;x:\sigma} \circ \langle \mathrm{id}_{|\Gamma|}, |P|_\Gamma \rangle = |\,\mathsf{T}\, s|_{\Gamma;x:\sigma} \circ \langle \mathrm{id}_{|\Gamma|}, |P|_\Gamma \rangle$ by the induction hypothesis and the definition of $|\cdot|$. □

Lemma 5.5 *Let* $\Gamma; x : \sigma \vdash M : \tau$. *Then* $|(\mathsf{T}_x\, M)^{\mathsf{T}\,\tau}|_{\Gamma;x:\mathsf{T}\,\sigma} = T(|M^\tau|_{\Gamma;x:\sigma}) \circ t$.

Proof. The proof is by induction on M.

- $M \equiv x$. Then, on the one hand, $|\,\mathsf{T}_x\, x|_{\Gamma;x:\mathsf{T}\,\sigma} = |x|_{\Gamma;x:\mathsf{T}\,\sigma} = \pi_2$ by the definitions of T_x and $|\cdot|$. On the other hand, expanding the definitions of $|\cdot|$ and t, and observing that $T(\pi_2) = \pi_2 \times \pi_2$ by Lemma 3.7 because π_2 is linear, we obtain $T(|x|_{\Gamma;x:\sigma}) \circ t = T(\pi_2) \circ t = (\pi_2 \times \pi_2) \circ \langle 0 \times \pi_1, \mathrm{id} \times \pi_2 \rangle$, which reduces to π_2 by the properties of left-additive cartesian categories.
- $M \equiv y \neq x$. Then, on the one hand, by the definitions of T_x and $|\cdot|$, we have $|\,\mathsf{T}_x\, y|_{\Gamma;x:\mathsf{T}\,\sigma} = |\iota_2 y|_{\Gamma;x:\mathsf{T}\,\sigma} = \langle 0, |y|_{\Gamma;x:\mathsf{T}\,\sigma} \rangle = \langle 0, |y|_\Gamma \circ \pi_1 \rangle$. On the other hand, expanding the definitions of $|\cdot|$ and t, and observing that $T(\pi_1) = \pi_1 \times \pi_1$ by Lemma 3.7 because π_1 is linear, we obtain $T(|y|_{\Gamma;x:\sigma}) \circ t = T(|y|_\Gamma \circ \pi_1) \circ t = T(|y|_\Gamma) \circ T(\pi_1) \circ t = T(|y|_\Gamma) \circ (\pi_1 \times \pi_1) \circ \langle 0 \times \pi_1, \mathrm{id} \times \pi_2 \rangle$, which reduces to $T(|y|_\Gamma) \circ \langle 0, \pi_1 \rangle$ by the properties of left-additive cartesian categories. It follows from the definition of the interpretation $|\cdot|$ that $|y|_\Gamma$ is a composition of projections, and hence is linear. Lemma 3.7 implies that $T(|y|_\Gamma) = |y|_\Gamma \times |y|_\Gamma$. Furthermore, each linear morphisms is additive, so that, in particular, $|y|_\Gamma \circ 0 = 0$. The assertion follows.
- $M \equiv sN$. By the definition of T_x and Lemma 5.3, we have $|\,\mathsf{T}_x(sN)|_{\Gamma;x:\mathsf{T}\,\sigma} = |(\mathsf{T}_x\, s) \diamond (\mathsf{T}_x\, N)|_{\Gamma;x:\mathsf{T}\,\sigma} = T(\mathrm{ev}) \circ \psi \circ \langle |\,\mathsf{T}_x\, s|_{\Gamma;x:\mathsf{T}\,\sigma}, |\,\mathsf{T}_x\, N|_{\Gamma;x:\mathsf{T}\,\sigma} \rangle$. By the induction hypothesis, $|\,\mathsf{T}_x\, s|_{\Gamma;x:\mathsf{T}\,\sigma} = T(|s|_{\Gamma;x:\sigma}) \circ t$ and $|\,\mathsf{T}_x\, N|_{\Gamma;x:\mathsf{T}\,\sigma} = T(|N|_{\Gamma;x:\sigma}) \circ t$. Therefore, $|\,\mathsf{T}_x(sN)|_{\Gamma;x:\mathsf{T}\,\sigma} = T(\mathrm{ev}) \circ \psi \circ \langle T(|s|_{\Gamma;x:\sigma}) \circ t, T(|N|_{\Gamma;x:\sigma}) \circ t \rangle = T(\mathrm{ev}) \circ \psi \circ \langle T(|s|_{\Gamma;x:\sigma}), T(|N|_{\Gamma;x:\sigma}) \rangle \circ t$, which is equal to $T(\mathrm{ev}) \circ T(\langle |s|_{\Gamma;x:\sigma}, |N|_{\Gamma;x:\sigma} \rangle) \circ t = T(\mathrm{ev} \circ \langle |s|_{\Gamma;x:\sigma}, |N|_{\Gamma;x:\sigma} \rangle) \circ t = T(|sN|_{\Gamma;x:\sigma}) \circ t$ by Lemma 3.8, the functoriality of T, and the definition of $|\cdot|$.
- $M \equiv (\lambda y.\, s)^{\tau_1 \to \tau_2}$, where by α-conversion we may assume that x is different from y. By the definitions of T_x and $|\cdot|$, we have $|\,\mathsf{T}_x(\lambda y.\, s)|_{\Gamma;x:\mathsf{T}\,\sigma} = |\langle \lambda y.\, \pi_1(\mathsf{T}_x\, s), \lambda y.\, \pi_2(\mathsf{T}_x\, s) \rangle|_{\Gamma;x:\mathsf{T}\,\sigma} = \langle |\lambda y.\, \pi_1(\mathsf{T}_x\, s)|_{\Gamma;x:\mathsf{T}\,\sigma}, |\lambda y.\, \pi_2(\mathsf{T}_x\, s)|_{\Gamma;x:\mathsf{T}\,\sigma} \rangle = \langle \Lambda(\pi_1 \circ |\,\mathsf{T}_x\, s|_{\Gamma;x:\mathsf{T}\,\sigma;y:\tau_1}), \Lambda(\pi_2 \circ |\,\mathsf{T}_x\, s|_{\Gamma;x:\mathsf{T}\,\sigma;y:\tau_1}) \rangle$. By Lemma 5.1, this can be transformed as $\langle \Lambda(\pi_1 \circ |\,\mathsf{T}_x\, s|_{\Gamma;y:\tau_1;x:\mathsf{T}\,\sigma} \circ \mathrm{sw}), \Lambda(\pi_2 \circ |\,\mathsf{T}_x\, s|_{\Gamma;y:\tau_1;x:\mathsf{T}\,\sigma} \circ \mathrm{sw}) \rangle$ which coincides with $\langle \Lambda(\pi_1 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma}) \circ t \circ \mathrm{sw}), \Lambda(\pi_2 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma}) \circ t \circ \mathrm{sw}) \rangle$ by the induction hypothesis. Proposition 3.13, equation (1), and the functoriality of

T allows us to write this expression as follows:

$$\begin{aligned}
&\langle\, \Lambda(\pi_1 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma}) \circ T(\mathrm{sw}) \circ t' \circ (t \times \mathrm{id})),\\
&\quad \Lambda(\pi_2 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma}) \circ T(\mathrm{sw}) \circ t' \circ (t \times \mathrm{id}))\rangle\\
&= \langle\, \Lambda(\pi_1 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma} \circ \mathrm{sw}) \circ t'), \Lambda(\pi_2 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma} \circ \mathrm{sw}) \circ t')\rangle \circ t\\
&= \langle\, \Lambda(\pi_1 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma} \circ \mathrm{sw}) \circ t'), \Lambda(\pi_2 \circ T(|s|_{\Gamma;y:\tau_1;x:\sigma} \circ \mathrm{sw}) \circ t')\rangle \circ t.
\end{aligned}$$

By Proposition 3.9, the last expression is equal to $T(\Lambda(|s|_{\Gamma;y:\tau_1;x:\sigma} \circ \mathrm{sw})) \circ t$, which is in turn equal to $T(\Lambda(|s|_{\Gamma;x:\sigma;y:\tau_1})) \circ t = T(|\lambda y.\, s|_{\Gamma;x:\sigma}) \circ t$ by Lemma 5.1.

- $M \equiv \mathsf{T}\, s$. By the definitions of T_x and $|\cdot|$, and by the induction hypothesis we have $|\,\mathsf{T}_x(\mathsf{T}\, s)|_{\Gamma;x:\mathsf{T}\,\sigma} = |\langle \mathsf{T}(\pi_1(\mathsf{T}_x\, s)), \mathsf{T}(\pi_2(\mathsf{T}_x\, s))\rangle|_{\Gamma;x:\mathsf{T}\,\sigma} = \langle \underline{T} \circ \pi_1 \circ |\,\mathsf{T}_x\, s|_{\Gamma;x:\mathsf{T}\,\sigma}, \underline{T} \circ \pi_2 \circ |\,\mathsf{T}_x\, s|_{\Gamma;x:\mathsf{T}\,\sigma}\rangle = (\underline{T} \times \underline{T}) \circ |\,\mathsf{T}_x\, s|_{\Gamma;x:\mathsf{T}\,\sigma} = (\underline{T} \times \underline{T}) \circ T(|s|_{\Gamma;x:\sigma}) \circ t$. The morphism $\underline{T}$ is linear by Theorem 3.10. Therefore $\underline{T} \times \underline{T} = T(\underline{T})$ by Lemma 3.7. We conclude that $|\,\mathsf{T}_x(\mathsf{T}\, s)|_{\Gamma;x:\mathsf{T}\,\sigma} = T(\underline{T}) \circ T(|s|_{\Gamma;x:\sigma}) \circ t = T(\underline{T} \circ |s|_{\Gamma;x:\sigma}) \circ t = T(|\,\mathsf{T}\, s|_{\Gamma;x:\sigma}) \circ t$ by the functoriality of T.
- The remaining cases ($M \equiv 0$, $M \equiv s + N$, $M \equiv \iota_k s$, and $M \equiv \pi_k s$) are straightforward.

The lemma is proven. □

Theorem 5.6 *Let* $\mathbf{C}$ *be a differential* λ*-category. Then* $\mathrm{Th}(\mathbf{C})$ *is a perturbative* λ*-theory.*

Proof. We have to check that $\mathrm{Th}(\mathbf{C})$ is closed under the rules from Figure 5.

(β) By the definition of $|\cdot|$, we have $|(\lambda x.\, s)N|_\Gamma = \mathrm{ev} \circ \langle \Lambda(|s|_{\Gamma;x:\sigma}), |N|_\Gamma\rangle$. On the other hand, by Lemma 5.4, we have $|s[N/x]|_\Gamma = |s|_{\Gamma;x:\sigma} \circ \langle \mathrm{id}_{|\Gamma|}, |N|_\Gamma\rangle$, which is equal to $\mathrm{ev} \circ \langle \Lambda(|s|_{\Gamma;x:\sigma}), |N|_\Gamma\rangle$ by (2).

(T) By the definition of $|\cdot|$, we have $|(\mathsf{T}(\lambda x.\, s))^{\mathsf{T}\,\sigma \to \mathsf{T}\,\tau}|_\Gamma = \underline{T} \circ \Lambda(|s^\tau|_{\Gamma;x:\sigma})$. On the other hand, by Lemma 5.5, $|(\lambda x.\, \mathsf{T}_x\, s)^{\mathsf{T}\,\sigma \to \mathsf{T}\,\tau}|_\Gamma = \Lambda(|(\mathsf{T}_x\, s)^{\mathsf{T}\,\tau}|_{\Gamma;x:\mathsf{T}\,\sigma}) = \Lambda(T(|s^\tau|_{\Gamma;x:\sigma}) \circ t)$, which is equal to $\underline{T} \circ \Lambda(|s^\tau|_{\Gamma;x:\sigma})$ by Proposition 3.11.

($\pi\iota$) By the definition of $|\cdot|$, we have $|\pi_k(\iota_l s)|_\Gamma = \pi_k \circ \iota_l \circ |s|_\Gamma$, which is equal to $|s|_\Gamma$ if $k = l$ and to 0 otherwise.

The weakening rule (W) follows from Lemma 5.2. Symmetry, reflexivity, and transitivity are obvious from the definition of $\mathrm{Th}(\mathbf{C})$. The remaining rules follow from the definition of the interpretation. □

6 Conclusions and Future Work

We have presented a novel calculus, the perturbative λ-calculus, which shares some similarity with the differential λ-calculus of Ehrhard and Regnier [4], but is based on the differential geometric idea of pushforward rather than that of derivative. We believe that this feature makes the perturbative λ-calculus well suited for reasoning about forward AD. We have shown that the perturbative λ-calculus can be modeled by differential λ-categories. The following issues have been left open in this paper:

- Confluence and strong normalization of the perturbative λ-calculus. We believe

that the proofs from [4] for the differential λ-calculus can be adapted for the perturbative λ-calculus.

- We conjecture that the perturbative λ-calculus does not suffer from the inefficiencies of the differential λ-calculus. Furthermore, we believe that the perturbative λ-calculus can provide the computational complexity guarantees of forward AD. Formal proofs of these statements require giving an operational semantics of the perturbative λ-calculus.
- The practical value of the perturbative λ-calculus is rather limited. To become a theoretical foundation of a practical programming language, the perturbative λ-calculus has to be extended with other programming language features (recursion, sum types, polymorphism etc.).
- We have considered only forward AD. The other modes of AD require other novel extensions to the λ-calculus.

We plan to address these issues in forthcoming papers.

Acknowledgement

I would like to thank Alexey Radul for fruitful discussions and for carefully reading the preliminary versions of this paper. I am grateful to anonymous referees for valuable suggestions that have improved the exposition.

References

[1] Blute, R., T. Ehrhard and C. Tasson, *A convenient differential category*, Cahier de Topologie et Géométrie Différentielle Catégoriques (2011), to appear.
URL http://arxiv.org/abs/1006.3140

[2] Blute, R. F., J. R. B. Cockett and R. A. G. Seely, *Cartesian differential categories*, Theory Appl. Categ. **22** (2009), pp. 622–672.

[3] Bucciarelli, A., T. Ehrhard and G. Manzonetto, *Categorical models for simply typed resource calculi*, Electron. Notes Theor. Comput. Sci. **265** (2010), pp. 213–230.

[4] Ehrhard, T. and L. Regnier, *The differential lambda-calculus*, Theoret. Comput. Sci. **309** (2003), pp. 1–41.

[5] Eilenberg, S. and G. M. Kelly, *Closed categories*, in: *Proc. Conf. Categorical Algebra (La Jolla, Calif., 1965)*, Springer, New York, 1966 pp. 421–562.

[6] Griewank, A., "Evaluating derivatives," Frontiers in Applied Mathematics **19**, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000, xxiv+369 pp., principles and techniques of algorithmic differentiation.

[7] Kock, A., *Monads on symmetric monoidal closed categories*, Arch. Math. (Basel) **21** (1970), pp. 1–10.

[8] Kock, A., *Strong functors and monoidal monads*, Arch. Math. (Basel) **23** (1972), pp. 113–120.

[9] Manzyuk, O., *Tangent bundles in differential λ-categories* (2012), preprint.
URL http://arxiv.org/abs/1202.0411

[10] Moggi, E., *Notions of computation and monads*, Inform. and Comput. **93** (1991), pp. 55–92, selections from the 1989 IEEE Symposium on Logic in Computer Science.

[11] Pearlmutter, B. A. and J. M. Siskind, *Reverse-mode AD in a functional framework: Lambda the ultimate backpropagator*, ACM Trans. Program. Lang. Syst. **30** (2008), pp. 1–36.

[12] Siskind, J. M. and B. A. Pearlmutter, *Nesting forward-mode AD in a functional framework*, Higher Order Symbol. Comput. **21** (2008), pp. 361–376.

[13] Siskind, J. M. and B. A. Pearlmutter, *Using polyvariant union-free flow analysis to compile a higher-order functional-programming language with a first-class derivative operator to efficient Fortran-like code*, Technical Report TR-ECE-08-01, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA (2008).
URL http://docs.lib.purdue.edu/ecetr/367