A Language for Differentiable Functions

Pietro Di Gianantonio[1] Abbas Edalat[2]

[1] Dip. di Matematica e Informatica
Università di Udine, 33100 Udine, Italy
pietro.digianatonio@uniud.it
[2] Department of Computing, Imperial College London
ae@ic.ac.uk

Abstract. We introduce a typed lambda calculus in which real numbers, real functions, and in particular continuously differentiable and more generally Lipschitz functions can be defined. Given an expression representing a real-valued function of a real variable in this calculus, we are able to evaluate the expression on an argument but also evaluate the L-derivative of the expression on an argument. The language is an extension of PCF with a real number data-type but is equipped with primitives for min and weighted average to capture computable continuously differentiable or Lipschitz functions on real numbers. We present an operational semantics and a denotational semantics based on continuous Scott domains and several logical relations on these domains. We then prove an adequacy result for the two semantics. The denotational semantics also provides denotational semantics for Automatic Differentiation. We derive a definability result showing that for any computable Lipschitz function there is a closed term in the language whose evaluation on any real number coincides with the value of the function and whose derivative expression also evaluates on the argument to the value of the L-derivative of the function.

Introduction

Real-valued locally Lipschitz maps on finite dimensional Euclidean spaces enjoy a number of fundamental properties which make them the appropriate choice of functions in many different areas of applied mathematics and computation. They contain the class of continuously differentiable functions, are closed under composition and the absolute value, min and max operations, and contain the important class of piecewise polynomial functions, which are widely used in geometric modelling, approximation and interpolation and are supported in MatLab [4]. Lipschitz maps with uniformly bounded Lipschitz constants are closed under convergence with respect to the sup norm. Another fundamental property of these maps is that a Lipschitz vector field in $\mathbb{R}^n$ has a unique solution in the initial value problem [3].

In the past thirty years, motivated by applications in control theory and optimisation and using an infinitary double limit superior operation, the notion

of Clarke gradient has been developed as a convex and compact set-valued generalized derivative for real-valued locally Lipschitz maps [2]. For example, the absolute value function, which is not classically differentiable at zero, is a Lipschitz map which has Clarke gradient $[-1, 1]$ at zero. The Clarke gradient extends the classical derivative for continuously differentiable functions.

Independently, a domain-theoretic Scott continuous Lipschitz derivative, later called the L-derivative, was introduced in [9] for interval-valued functions of an interval variable and was used to construct a domain for locally Lipschitz maps; these results were then extended to higher dimensions [10]. It was later shown that on finite dimensional Euclidean spaces the L-derivative actually coincides with the Clarke gradient [6]. In finite dimensions, therefore, the L-derivative provides a simple and finitary representation for the Clarke gradient.

Since the mid 1990's, a number of typed lambda calculi, namely extensions of PCF with a real number data type, have been constructed, including Real PCF, RL and LPR [12, 5, 19], which are essentially equivalent and in which computable continuous functions can be defined. Moreover, IC-Reals, a variant of LPR with seven digits, has been implemented with reasonable efficiency in C and Haskell [14].

It was relatively straightforward in [8] to equip Real PCF with the integral operator, which is in fact a continuous functional. However, adding a derivative operator to the language has proved to be non-trivial since classic differentiation may not be defined on continuous functions and even when defined it may not result in a continuous function. The development of the Scott continuous L-derivative, defined in a finitary manner, has therefore been essential for construction of a language with a derivative operator.

The aim of this work is to take the current extensions of PCF with a real number data type into a new category and define a typed lambda calculus, in which real numbers, real functions and in particular continuously differentiable and Lipschitz functions are definable objects. Given an expression e representing a function from real numbers to real numbers in this language, we would be able to evaluate both e and its L-derivative on an argument. In this paper we will only be concerned with the theoretical feasibility of such a language and not with questions of efficiency.

To develop such a language, we need to find a suitable replacement for the test for positiveness ($(\mathsf{0}<)$), which is used in the current extensions of PCF with real numbers to define functions by cases. In fact, a function defined using the conditional with this constructor will not be differentiable at zero even if the two outputs of the conditional are both differentiable: Suppose we have two real computable functions f and g whose derivatives $D\,f$ and $D\,g$ are also computable, and consider $l = \lambda x.\ \mathsf{if}\ (\mathsf{0}<)\ x\ \mathsf{then}\ f\,x\ \mathsf{else}\ g\,x$. The function l is computable and there is an effective way to obtain approximations of the value of $l(x)$ including at 0. However, there is no effective way to generate any approximation for the derivative of l, i.e., $D\,l$, at the point 0. In fact, it is correct to generate an approximation of Dl on 0 only if $f(0) = g(0)$, but this equality

is undecidable, i.e., it cannot be established by observing the computation of f and g at 0 for any finite time.

In this paper, instead of the test $(\mathsf{0}<)$, we will use the functions minimum, negation and weighted average when defining continuously differentiable or Lipschitz maps. These primitives are of course definable in Real PCF, RL and LPR, but the definitions are based on the test $(\mathsf{0}<)$, which means that the information about the derivative is lost.

By a simple transfer of the origin and a rescaling of coordinates we can take the interval $[-1, 1]$ as the domain of definition of Lipschitz maps. Furthermore, by a rescaling of the values of Lipschitz maps (i.e., multiplying them with the reciprocal of their Lipschitz constant) we can convert them to non-expansive maps, i.e., we can take their Lipschitz constant to be one. Concretely, we take digits similar to those in Real PCF as constructors and develop an operational semantics and a denotational semantics based on three logical relations, and prove an adequacy result. The denotational semantics for first order types is closely related but different from the domain constructed in [9] in that we capture approximations to the function part and to the derivative part regarded as a sublinear map on the tangent space. Finally, we prove a definability result and show that every computable Lipschitz map is definable in the language as the limit of a sequence of piecewise linear maps with the convergence of their L-derivatives.

We note that all our proofs can be found in the Appendix.

0.1 Related work

Given a programme to evaluate the values of a function defined in terms of a number of basic primitives, Automatic Differentiation (also called Algorithmic Differentiation) seeks to use the chain rule to compute the derivative of the function [13]. AD is distinct from symbolic differentiation and from numerical differentiation. Our work can be regarded as providing denotational semantics for *forward* Automatic Differentiation and can be used to extend AD to computation of the generalised derivative of Lipschitz functions.

In [11], the *differential* λ*-calculus*, and in [17], the perturbative λ-calculus that integrates the latter with AD, have been introduced which syntactically model the derivative operation on power series in a typed λ-calculus or a full linear logic. Although apparently similar, our calculus and these two λ-calculi differ in almost every aspect: motivation, syntax, semantics, and the class of definable real functions. (i) These λ-calculi have been presented to analyse linear substitution and formal differentiation, (ii) the syntax is quite structured and contains constructors that have no correspondence in our setting, (iii) the semantics is based on differential categories and not on domain theory, and (iv) the definable real functions are limited to analytical maps which have power series expansion.

On the other hand, Computable Analysis [20, 21] and Constructive Analysis [1] are not directly concerned with computation of the derivative and both

only deal with continuously differentiable functions. In fact, a computable real-valued function with a continuous derivative has a computable derivative if and only if the derivative has a recursive modulus of uniform continuity [15, p. 191], [20, p. 53], which is precisely the definition of a differentiable function in constructive mathematics [1, p. 44].

1 Syntax

We denote the new language with PCDF (Programming language for Computable and Differentiable Functions).

The types of PCDF are the types of a slightly modified version of PCF where natural numbers are replaced by integers, together with a new type ι, an expression e of type ι denotes a real number in the interval $[-1, 1]$ or a partial approximation of a real number, represented by a closed intervals contained in $[-1, 1]$. The set T of type expressions is defined by the grammar:

$$\sigma ::= o \mid \nu \mid \iota \mid \sigma \to \sigma$$

where o is the type of booleans and ν is the type of integer numbers.

The expressions of PCDF are the expressions of PCF together with a new set of constants for dealing with real numbers. This set of constants is composed by the following elements:

(i) A constructor for real numbers given by: $\mathsf{dig} : \nu \to \nu \to \nu \to \iota \to \iota$. It is used to build affine transformations, and real numbers are obtained by a limiting process. The expression $\mathsf{dig}\, l\, m\, n$ represents the affine transformation $\lambda x.(l + m \cdot x)/n$, if $0 \leq m < n$ and $|l| \leq n - m$, or the constant function $\lambda x.0$ otherwise. The above condition on l, m, n implies that $\mathsf{dig}\, l\, m\, n$ represents a rational affine transformation mapping the interval $[-1, 1]$ strictly into itself with a non-negative slope or derivative $0 \leq m/n < 1$.
In this way we use three integers to encode a rational affine transformation; of course it is possible to devise other encodings where just natural numbers or a single natural number is used, however these alternative encodings will be more complex.
The affine transformations definable by dig are also called generalised digits. Since there is no constant having type ι, an expression e having type ι can never normalise and its evaluation proceeds by producing expressions in the form $\mathsf{dig}\, l\, m\, n\, e'$. These expressions give partial information about the value represented by e, namely they state that e represents a real number contained in the interval $[(l + m)/n,\ (l - m)/n]$, which is the range of the function $\lambda x.(l + m \cdot x)/n$. During the reduction process, this interval is repeatedly refined and the exact result, a completely defined real number, can be obtained as the limit of this sequence.

(ii) The opposite sign function (negation) $\mathsf{opp} : \iota \to \iota$.

(iii) $\mathsf{add} : \nu \to \nu \to \iota \to \iota$, representing the function $\lambda\, p\, q\, x\, .\, \min((p/q + x), 1)$, if $0 < p < 2 \cdot q$, and the constant function $\lambda x.\, 0$ otherwise.

We define $\mathsf{sub}\, p\, q\, x$ as syntactic sugar for the expression $\mathsf{opp}\,(\mathsf{add}\ p\, q\,(\mathsf{opp}\, x))$, which returns the value $\max((x - (p/q), -1))$.

(iv) A weighted average function $\mathsf{av} : \nu \to \nu \to \iota \to \iota \to \iota$. The expression $\mathsf{av}\, p\, q$ represents the function $\lambda x\, y\, .\, (p/q) \cdot x + (1 - p/q) \cdot y$, if $0 < p < q$, and the constant function $\lambda x\, y\, .\, 0$ otherwise.

(v) The minimum function

$$\mathsf{min} : \iota \to \iota \to \iota$$

with the obvious action on pairs of real numbers. We define $\mathsf{max}\ x\, y$ as syntactic sugar for the expression $\mathsf{opp}\,(\mathsf{min}\,(\mathsf{opp}\, x)(\mathsf{opp}\, y))$.

(vi) A test function $(\mathsf{0}<) : \iota \to o$, which returns true if the argument is strictly greater than zero, and false if the argument is strictly smaller that zero. The test function can be used for constructing functions that are not differentiable, an example being the function $\lambda x.\, \mathsf{if}\ (\mathsf{0}<)\ (x)\ \mathsf{then}\ 1\ \mathsf{else}\ 0$; as a consequence we impose some restriction in its use.

(vii) The if-then-else constructor on reals, $\mathsf{if}_\iota : o \to \iota \to \iota \to \iota$, and the parallel if-then-else constructor $\mathsf{pif}_\iota : o \to \iota \to \iota \to \iota$.

The main use for the parallel if operator is to evaluate, without loss of information, derivative of expressions containing the min operator. However, the parallel if operator can be completely avoided in defining non-expansive functions on real numbers. In fact in the constructive proof of our definability result, the parallel if operator is never used.

(viii) A new binding operator D. The operator D can bind only variables of type ι and can be applied only to expressions of type ι. In our language, $\mathsf{D}x.e$ represents the derivative of the real function $\lambda x.e$.

The differential operator D can be applied only to expressions that contain neither the constant $(\mathsf{0}<)$ nor the differential operator D itself.

We note that, with the exception of the test functions $(\mathsf{0}<)$, all the new constants represent functions on reals that are non-expansive; the if-then-else constructors are also non-expansive if the distance between true (tt) and false (ff) is defined to be equal to two, while the test function $(\mathsf{0}<)$ cannot be non-expansive, whatever metric is defined on the Boolean values. The expressions containing neither the constant $(\mathsf{0}<)$ nor the differential operator D are called *non-expansive* since they denote functions on real numbers that are non-expansive. This fact, intuitively true, is formally proved by Proposition 2. The possibility to syntactically characterise a sufficiently rich set of expressions representing non-expansive functions is a key ingredient in our approach that allows us to obtain information about the derivative of a function expression without completely evaluating it. For example, from the fact that $e : \iota$ is a non-expansive expression, one can establish that the derivative of $\lambda x.e$, at any point, is contained in the interval $[-1, 1]$ and that the derivative of $\lambda x.\mathsf{dig}\, l\, m\, n\, e$ is contained in the smaller interval $[-m/n, m/n]$.

2 Operational semantics

The operational semantics is given by a *small step reduction relation*, $\rightarrow$, which is obtained by adding to the PCF reduction rules the following set of extra rules for the new constants.

The operational semantics of add and min operators uses an extra constant $\mathsf{aff} : \nu \rightarrow \nu \rightarrow \nu \rightarrow \iota \rightarrow \iota$. The expression $\mathsf{aff}\, l\, m\, n$ is intended to represent general affine transformations (including expansive ones) with a non-negative derivative, i.e., the affine transformation $\lambda x.(l + mx)/n$ with $m \geq 0, n > 0$. A property preserved (i.e., invariant) by the reduction rules is that the constant aff appears only as the head of one of the arguments of min or as the head of the fourth argument of aff. It follows that in any expression e' in the reduction chain of a standard expression e (without the extra constants aff), the constant aff can appear only in the above positions.

The generalised digit $\mathsf{dig}\, l\, m\, n$ is a special case of an affine transformation. Therefore, in applying the reduction rules, we use the convention that any reduction rule containing, on the left hand side, a general affine transformation aff can be applied also to terms where the affine transformation aff is substituted by the constructor dig.

On affine transformations we will use the following notations:

- $(\mathsf{aff}\, l_1\, m_1\, n_1) \circ (\mathsf{aff}\, l_2\, m_2\, n_2)$ stands for $\mathsf{aff}\, (l_1 \cdot n_2 + m_1 \cdot l_2)\, (m_1 \cdot m_2)\, (n_1 \cdot n_2)$, i.e., the composition of affine transformations.
- If $m \neq 0$, $(\mathsf{aff}\, l\, m\, n)^{-1}$ stands for $(\mathsf{aff}\, (-l)\, n\, m)$, i.e., the inverse affine transformation; if $m = 0$, the expression $(\mathsf{aff}\, l\, m\, n)^{-1}$ is undefined.
- The symbols l, m, n, p, q stand for values of integer type.

The reduction rules are the PCF reduction rules together the following set of extra rules. First we have three simple reductions:

- $\mathsf{dig}\, l\, m\, n\, e \rightarrow \mathsf{dig}\, 0\, 0\, 1\, e$ if $m < 0$ or $n \leq 0$ or $|l| > n - m$.
- $\mathsf{add}\, p\, q\, e \rightarrow \mathsf{dig}\, 0\, 0\, 1\, e$ if $p \leq 0$ or $q \leq p$.
- $\mathsf{av}\, p\, q\, e_1\, e_2 \rightarrow \mathsf{dig}\, 0\, 0\, 1\, e_1$ if $p \leq 0$ or $q \leq p$.

The above rules deal with those instances of dig, add, av with integer arguments that reduce to the constant zero digit. An implicit condition on the following set of rules is that they apply only if none of the above three rules can be applied.

1. $\mathsf{dig}\, l_1\, m_1\, n_1 (\mathsf{dig}\, l_2\, m_2\, n_2\, e) \rightarrow ((\mathsf{dig}\, l_1\, m_1\, n_1) \circ (\mathsf{dig}\, l_2\, m_2\, n_2))\, e$
2. $\mathsf{opp}\, (\mathsf{dig}\, l\, m\, n\, e) \rightarrow \mathsf{dig}\, (-l)\, m\, n\, (\mathsf{opp}\, e)$
3. $\mathsf{add}\, p\, q\, e \rightarrow \mathsf{min}\, (\mathsf{aff}\, p\, q\, q\, e)(\mathsf{dig}\, 1\, 0\, 1\, e)$
   note that $(\mathsf{aff}\, p\, q\, q)$ and $(\mathsf{dig}\, 1\, 0\, 1)$ represent the functions $\lambda x.p/q + x$ and $\lambda x.1$ respectively.
4. $\mathsf{av}\, p\, q\, (\mathsf{dig}\, l\, m\, n\, e_1)\, e_2 \rightarrow \mathsf{dig}\, l'\, m'\, n'(\mathsf{av}\, p'\, q'\, e_1\, e_2)$
   where $l' = l \cdot p, \quad m' = q' = m \cdot p + n \cdot q - n \cdot p, \quad n' = n \cdot q$ and $p' = m \cdot p$.
   By a straightforward calculation, one can check that the left and the right parts of the reduction rules represent the same affine transformation on the arguments e_1, e_2.

5. $\mathsf{av}\, p\, q\, e_1\, (\mathsf{dig}\, l\, m\, n\, e_2) \;\to\; \mathsf{dig}\, l'\, m'\, n'(\mathsf{av}\, p'\, q'\, e_1\, e_2)$
   where $l' = l(q-p), \quad m' = q' = np + mq - mp, \quad n' = nq$ and $p' = np$.
6. $\mathsf{min}\, (\mathsf{dig}\, l_1\, m_1\, n_1\, e_1)(\mathsf{aff}\, l_2\, m_2\, n_2\, e_2) \;\to\; \mathsf{dig}\, l_1\, m_1\, n_1\, e_1$
   if $(l_1 + m_1)/n_1 \leq (l_2 - m_2)/n_2$.
   The above condition states that every point in the image of $(\mathsf{dig}\, l_1\, m_1\, n_1)$ is smaller, in the usual Euclidean order, than every point in the image of $(\mathsf{aff}\, l_2\, m_2\, n_2)$, i.e., the first argument of min is certainly smaller that the second.
7. $\mathsf{min}\, (\mathsf{aff}\, l_1\, m_1\, n_1\, e_1)(\mathsf{dig}\, l_2\, m_2\, n_2\, e_2) \;\to\; \mathsf{dig}\, l_2\, m_2\, n_2\, e_2$
   if $(l_2 + m_2)/n_2 \leq (l_1 - m_1)/n_1$
   The symmetric version of the previous rule.
8. $\mathsf{min}\, (\mathsf{dig}\, l\, m\, n\, e_1)\, e_2 \;\to$
   $\mathsf{dig}\, l'\, m'\, n'(\mathsf{min}\, ((\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{dig}\, l\, m\, n)\, e_1)\, ((\mathsf{dig}\, l'\, m'\, n')^{-1}\, e_2))$
   if $l + m < n$ and $l' = l + m - n$, $m' = l + m + n \neq 0$, $n' = 2 \cdot n$.
   The above equations imply that if $(\mathsf{dig}\, l\, m\, n)$ has image $[a, b]$ then $(\mathsf{dig}\, l'\, m'\, n')$ has image $[-1, b]$. The rule is justified by the fact if the first argument of min are smaller than b then the value of min is also smaller than b.
9. $\mathsf{min}\, e_1\, (\mathsf{dig}\, l\, m\, n\, e_2) \;\to$
   $\mathsf{dig}\, l'\, m'\, n'(\mathsf{min}\, ((\mathsf{dig}\, l'\, m'\, n')^{-1}\, e_1)\, ((\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{dig}\, l\, m\, n)\, e_2))$
   if $l + m < n$ and $l' = l + m - n$, $m' = l + m + n \neq 0$, $n' = 2 \circ n$.
   The symmetric version of the previous rule.
10. $\mathsf{min}\, (\mathsf{aff}\, l_1\, m_1\, n_1\, e_1)(\mathsf{aff}\, l_2\, m_2\, n_2\, e_2) \;\to$
    $\mathsf{dig}\, l'\, m'\, n'\, (\mathsf{min}\, ((\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{aff}\, l_1\, m_1\, n_1)\, e_1)((\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{aff}\, l_2\, m_2\, n_2)\, e_2))$
    if $-1 < (l_1 + m_1)/n_1 \leq (l_2 - m_2)/n_2$ and $l' = l_1 - m_1 + n_1$, $m' = m_1 - l_1 + m_1$, $n' = 2 \cdot n_1$.
    The above equation implies that if $(\mathsf{dig}\, l_1\, m_1\, n_1)$ has image $[a, b]$ then $(\mathsf{dig}\, l'\, m'\, n')$ has image $[a, 1]$. The rule is justified by the fact if both arguments of min are greater that a then the value of min is also greater than a.
11. $\mathsf{min}\, (\mathsf{aff}\, l_1\, m_1\, n_1\, e_1)(\mathsf{aff}\, l_2\, m_2\, n_2\, e_2) \;\to$
    $\mathsf{dig}\, l'\, m'\, n'\, (\mathsf{min}\, ((\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{aff}\, l_1\, m_1\, n_1)\, e_1)((\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{aff}\, l_2\, m_2\, n_2)\, e_2))$
    if $-1 < (l_2 - m_2)/n_2 < (l_1 + m_1)/n_1$, and $l' = l_2 - m_2 + n_2$, $m' = m_2 - l_2 + m_2$, $n' = 2 \cdot n_2$.
    The symmetric version of the previous rule.
12. $\mathsf{aff}\, l_1\, m_1\, n_1(\mathsf{aff}\, l_2\, m_2\, n_2\, e) \;\to\; ((\mathsf{aff}\, l_1\, m_1\, n_1) \circ (\mathsf{aff}\, l_2\, m_2\, n_2))\, e$
13. $\mathsf{aff}\, l\, m\, n\, e \;\to\; \mathsf{dig}\, l\, m\, n\, e$
    if $-1 \leq (l - m)/n$, $(l + m)/n \leq 1$ and e is not in the form $\mathsf{aff}\, a' b' e$.
14. $(\mathsf{0}{<})\, (\mathsf{dig}\, l\, m\, n\, e) \;\to\; \mathsf{tt} \qquad$ if $(l - m)/n > 0$
15. $(\mathsf{0}{<})\, (\mathsf{dig}\, l\, m\, n\, e) \;\to\; \mathsf{ff} \qquad$ if $(l - m)/n < 0$
16. $\mathsf{pif}_\iota\ e\ \mathsf{then}\ \mathsf{dig}\, l_1\, m_1\, n_1\, e_1\ \mathsf{else}\ \mathsf{dig}\, l_2\, m_2\, n_2\, e_2 \;\to$
    $\mathsf{dig}\, l'\, m'\, n'(\mathsf{pif}\ e\ \mathsf{then}\ (\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{dig}\, l_1\, m_1\, n_1)\, e_1$
    $\mathsf{else}\ \ (\mathsf{dig}\, l'\, m'\, n')^{-1} \circ (\mathsf{dig}\, l_2\, m_2\, n_2)\, e_2)$
    where $n' = 2 \cdot n_1 \cdot n_2$, $m' = \max((l_1 + m_1) \cdot n_2,\, (l_2 + m_2) \cdot n_1) - \min((l_1 - m_1) \cdot n_2,\, (l_2 - m_2) \cdot n_1)$, $l' = 2 \cdot \min((l_1 - m_1) \cdot n_2,\, (l_2 - m_2) + m'$.
    Here the values l', m', n' are defined in such a way that if $(\mathsf{dig}\, l_1\, m_1\, n_1)$ has image $[a_1, b_1]$ and $(\mathsf{dig}\, l_2\, m_2\, n_2)$ has image $[a_2, b_2]$, then $(\mathsf{dig}\, l'\, m'\, n')$ has image the convex closure of the set $[a_1, b_1] \cup [a_2, b_2]$.

The remaining rules for $\mathsf{pif}, \mathsf{if}$ are included in the reduction rules for PCF and therefore are omitted from the present list.

17. $\dfrac{N \to N'}{MN \to MN'}$ if M is a constant different from the combinator Y or is an expression in the form $\mathsf{min}\, n_1$, $\mathsf{min}\, n_1\, n_2$, $\mathsf{min}\, n_1\, n_2\, n_3$, $\mathsf{min}\, n_1\, n_2\, n_3\, M'$, $\mathsf{av}\, n_1$, $\mathsf{av}\, n_1\, n_2$, $\mathsf{av}\, n_1\, n_2\, M'$, $\mathsf{add}\, n_1$, $\mathsf{add}\, n_1\, n_2$, $\mathsf{pif}_\iota\ M'$ then, $\mathsf{pif}_\iota\ M'$ $\mathsf{then}\ M''$ else, where n_1, n_2, n_3 are values.

The reduction rules for the derivative operator are:

1. $\mathsf{D}x.\, x \to \lambda y.\, \mathsf{dig}\, 0\, 0\, 1\, y$
2. $\mathsf{D}x.\, \mathsf{dig}\, l\, m\, n\, e \to \lambda y.\, \mathsf{dig}\, 0\, m\, n\, (\mathsf{D}x.\, e)y$
3. $\mathsf{D}x.\, \mathsf{opp}\, e \to \lambda y.\, \mathsf{opp}\, (\mathsf{D}x.\, e)y$
4. $\mathsf{D}x.\, \mathsf{add}\, p\, e\, q \to \lambda y.\, \mathsf{pif}_\iota\ (0<)\ (\mathsf{add}\, (q-p)\, q\, (\mathsf{opp}\, e))\ \mathsf{then}\ (\mathsf{D}x.\, e)y\ \mathsf{else}\ \mathsf{dig}\, 0\, 0\, 1\, y$
5. $\mathsf{D}x.\, \mathsf{av}\, p\, q\, e_1\, e_2 \to \lambda y.\, \mathsf{av}\, p\, q\, ((\mathsf{D}x.\, e_1)y)\, ((\mathsf{D}x.\, e_2)y)$
6. $\mathsf{D}x.\, \mathsf{min}\ e_1 e_2 \to$
   $\lambda y.\, \mathsf{pif}\ (\lambda x.\, (0<)\, (\mathsf{av}\, 1\, 2\, (\mathsf{opp}\, e_1)e_2))y\ \mathsf{then}\ (\mathsf{D}x.\, e_1)y\ \mathsf{else}\ (\mathsf{D}x.\, e_2)y$
7. $\mathsf{D}x.\, \mathsf{pif}_\iota\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \to \lambda y.\, \mathsf{pif}_\iota\ (\lambda x.\, e_1)y\ \mathsf{then}\ (\mathsf{D}x.\, e_1)y\ \mathsf{else}\ (\mathsf{D}x.\, e_2)y$
8. $\mathsf{D}x.\, \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \to \lambda y.\, \mathsf{if}\ (\lambda x.\, e_1)y\ \mathsf{then}\ (\mathsf{D}x.\, e_1)y\ \mathsf{else}\ (\mathsf{D}x.\, e_2)y$
9. $\mathsf{D}x.\, Ye \to \mathsf{D}x.\, e(Ye)$
10. $\mathsf{D}x.\, (\lambda y.\, e)e_1 \ldots e_n \to \mathsf{D}x.\, e[e_1/y]e_2 \ldots e_n$

Note that the rules for the derivative operator are a direct derivation of the usual rules for the symbolic computation of the derivative of a function.

2.1 Examples

We will give some examples for defining non-analytic functions in later sections; in particular we will show in the proof of definability how easily piecewise linear maps with rational coefficients are defined in the language. A useful technique to define analytic functions and real constants is to consider their Taylor series expansions and reduce the Taylor series to a sequence of applications of affine transformations. For example the value $e-2$, where e is the Euler constant, is given by the Taylor series $1/2!+1/3!+1/4!\ldots$. Denoting the affine transformation $\lambda x.(1+x)/n$ as f, the above series can be expressed as $f(2)(f(3)(f(4)(\ldots)\ldots)$. It follows that in PCDF $e-2$ can be expressed as

$$(Y\, \lambda f : \nu \to \iota.\, \lambda n : \nu.\, f\, n.\, \mathsf{dig}\, 1\, 1\, n\, (f(n+1)))\, 2.$$

Given an expression to represent product in PCDF, it is possible to use the above technique to express analytic functions. For example, suppose hp defines the half-product function $\lambda xy.\, x \cdot y/2$. Then, one can express the function $\lambda x.\, e^{x/2} - 1 - x/2$ by the PCDF.

$$\lambda x : \iota.\, \mathsf{hp}\, x\, ((Y \lambda f : \nu \to \iota \to \iota.\, \lambda n : \nu.\, \lambda x : \iota.\, \mathsf{hp}\, x\, (\mathsf{dig}\, 1\, 1\, n\, (f(n+1))))\, 2\, x)$$

The half product hp is definable in PCDF by reducing product to a series of applications of the average and minimum function. The actual definition of the

function hp is lengthy and we will not present it here. As a simpler example of the technique involved, we present the definition of the function $\lambda x.x^2/2$.

Consider the following mutual recursive definition of the terms $g, h : \nu \to \iota \to \iota$

$g\,0\,x = \mathsf{max}\,x\,(\mathsf{opp}\,x)$

$h\,n\,x = \mathsf{add}\,1\,(2^{n+1})\,(g\,n\,(\mathsf{add}\,1\,(2^{n+1})\,x))$

$g\,(n+1)\,x = \mathsf{min}\,(h\,n\,x)(h\,n\,(\mathsf{opp}\,x))$

By standard techniques, one can derive a PCDF expression g satisfying the above recursive definition. The careful reader can check that the term:
$\lambda x.\,(\mathsf{sub}\,1\,2\,(\mathsf{av}\,1\,2\,(g\,0\,x)(\mathsf{av}\,1\,2\,(g\,1\,x)(\mathsf{av}\,1\,2\,(g\,2\,x)\ldots(\mathsf{av}\;1\,2\,(g\,n\,x)\,(\mathsf{dig}\,101x)\ldots)$
represents the step-wise linear interpolation of the function $\lambda x.x^2/2$ on the points of the set $\{i/2^n | i \in Z, -2^n \leq i \leq 2^n\}$. It follows that the function $\lambda x.x^2/2$ is defined by the term:
$\lambda x.\,(\mathsf{sub}\,1\,2((Y\lambda f : \nu \to \iota \to \iota.\,\lambda n : \nu.\,\lambda x : \iota.\,(\mathsf{av}\,1\,2\,(g\,n\,x)\,(f\,(n+1)\,x)))\,0\,x)$.

3 Denotational Semantics

The denotational semantics for PCDF is given in the standard way as a family of continuous Scott domains, $UD := \{\mathcal{D}_\sigma \mid \sigma \in T\}$. The basic types are interpreted using the standard flat domains of integers and booleans. The domain associated to real numbers is the product domain $\mathcal{D}_\iota = \mathcal{I} \times \mathcal{I}$, where $\mathcal{I}$ is the continuous Scott domain consisting of the non-empty compact subintervals of the interval $I = [-1, 1]$ partially ordered with reverse inclusion. Elements of $\mathcal{I}$ can represent either a real number x, i.e., the degenerated interval $[x, x]$, or some partial information about a real number x, i.e., an interval $[a, b]$, with $x \in [a, b]$. On the elements of $\mathcal{I}$, we consider both the set-theoretic operation of intersection ($\cap$), the pointwise extensions of the arithmetic operations, and the lattice operations on the domain information order $(\sqcap, \sqcup)$, [12]. Function types have the usual interpretation of call-by-name programming languages: $\mathcal{D}_{\sigma\to\tau} = \mathcal{D}_\sigma \to \mathcal{D}_\tau$.

A hand waiving explanation for the definition of the domain $\mathcal{D}_\iota = \mathcal{I} \times \mathcal{I}$, is that the first component is used to define the value part of the function while the second component is used to define the derivative part. More precisely, a (non-expansive) function f from I to I, is described, in the domain, by the product of two functions $\langle f_1, f_2\rangle : (\mathcal{I} \times \mathcal{I}) \to (\mathcal{I} \times \mathcal{I})$: the function $f_1 : (\mathcal{I} \times \mathcal{I}) \to \mathcal{I}$ represents the value part of f, in particular $f_1(i, j)$ is the image of the interval i under f for all intervals j, i.e., f_1 depends only on the first argument. The second function $f_2 : (\mathcal{I} \times \mathcal{I}) \to \mathcal{I}$ represents the derivative part. If $D\,f$ denotes the derivative of f, then $f_2(i, j)$ is the image of the intervals i and j under the function $\lambda x, y.D\,f\,(x) \cdot y$. Thus, f_2 is linear in its second component and $f_2(\{x\}, \{1\})$ is the derivative of f at the point x.

Note that with respect to the above interpretation, composition behaves correctly, that is if the pair $\langle f_1, f_2\rangle : (\mathcal{I} \times \mathcal{I}) \to (\mathcal{I} \times \mathcal{I})$ describes the value part and the derivative part of a function $f : I \to I$ and $\langle g_1, g_2\rangle : (\mathcal{I} \times \mathcal{I}) \to (\mathcal{I} \times \mathcal{I})$ describes a function $g : I \to I$ then $\langle h_1, h_2\rangle$ describes, by the chain rule, the function $f \circ g$ with $h_1(i, j) = f_1(g_1(i, j), g_2(i, j))$ and $h_2(i, j) = f_2(g_1(i, j), g_2(i, j))$.

The L-derivative of the non-expansive map $f : I \to I$ is the Scott continuous function $\mathcal{L}(f) : I \to \mathcal{I}$ defined by [6]:

$$\mathcal{L}(f)(x) = \bigcap\{b \in \mathcal{I} : \exists \text{ open interval } O \subset I,\ x \in O \text{ with } \tfrac{f(u)-f(v)}{u-v} \in b \text{ for all } u, v \in O, u \neq v\}.$$

Consider now the case of functions in two arguments. Given a function $g : I \to I \to I$, its domain description will be an element in $(\mathcal{I}\times\mathcal{I}) \to (\mathcal{I}\times\mathcal{I}) \to (\mathcal{I}\times\mathcal{I})$, which is isomorphic to $((\mathcal{I} \times \mathcal{I}) \times (\mathcal{I} \times \mathcal{I})) \to (\mathcal{I} \times \mathcal{I})$. Thus again, the domain description of g consists of a pair of functions $\langle g_1, g_2\rangle$, with g_1 describing the value part. If $D\,g\,(x_1, x_2)$ is the linear transformation representing the derivative of g at (x_1, x_2), then the function g_2 is a domain extension of the real function $\lambda x_1, y_1, x_2, y_2.D\,g\,(x_1, x_2) \cdot (y_1, y_2)$.

This approach for describing functions on reals is also used in (forward mode) Automatic Differentiation [13]. While Automatic Differentiation is different from our method in that it does not consider the domain of real numbers and the notion of partial reals, it is similar to our approach in that it uses two real numbers as input and a pair of functions to describe the derivative of functions on reals. Authomatic differentiation is also used in [17], while the idea of using two separated components to describe the value part and the derivative part in the domain-theoretic setting can be found also in [9].

The semantic interpretation function $\mathcal{E}$ is defined, by structural induction, in the standard way:

$$\begin{array}{ll}
\mathcal{E}[\![c]\!]_\rho & = \mathcal{B}[\![c]\!] \\
\mathcal{E}[\![x]\!]_\rho & = \rho(x) \\
\mathcal{E}[\![e_1 e_2]\!]_\rho & = \mathcal{E}[\![e_1]\!]_\rho(\mathcal{E}[\![e_2]\!]_\rho) \\
\mathcal{E}[\![\lambda x^\sigma.e]\!]_\rho & = \lambda d \in \mathcal{D}_\sigma.\mathcal{E}[\![e]\!]_{(\rho[d/x])}
\end{array}$$

The semantic interpretation of any PCF constant is the usual one, while the semantic interpretation of the new constants on reals is given by:

$$\mathcal{B}[\![\mathsf{dig}\,]\!](l, m, n, \langle i, j\rangle) = \begin{cases} \bot & \text{if } \ l = \bot \ \vee \ m = \bot \ \vee \ n = \bot \\ \langle [0,0], [0,0]\rangle & \text{if } \ \neg(0 \leq m < n \ \wedge \ |l| \leq n - m) \\ \langle l/n + m/n \cdot i,\ m/n \cdot j\rangle & \text{otherwise} \end{cases}$$

$$\mathcal{B}[\![\mathsf{opp}\,]\!](\langle i, j\rangle) = \langle -i,\ -j\rangle$$

$$\mathcal{B}[\![\mathsf{add}\,]\!](p, q, \langle i, j\rangle) = \begin{cases} \bot & \text{if } p = \bot \ \vee \ q = \bot \\ \langle [0,0], [0,0]\rangle & \text{if } \ \neg(0 < 2 \cdot p < q) \\ \langle i + p/q, j\rangle & \text{if } i + p/q < 1 \\ \langle [1,1], [0,0]\rangle & \text{if } i + p/q > 1 \\ \langle i + p/q \cap [-1,1],\ j \sqcap [0,0]\rangle & \text{otherwise} \end{cases}$$

$$\mathcal{B}[\![\mathsf{av}\,]\!](p, q, \langle i_1, j_1\rangle, \langle i_2, j_2\rangle)$$

$$= \begin{cases} \bot & \text{if } p = \bot \ \vee \ q = \bot \\ \langle [0,0], [0,0]\rangle & \text{if } \ \neg(0 < p < q) \\ \langle p/q \cdot i_1 + (1 - p/q) \cdot i_2,\ p/q \cdot j_1 + (1 - p/q) \cdot j_2\rangle & \text{otherwise} \end{cases}$$

$$\mathcal{B}[\![\mathsf{min}\,]\!](\langle i_1, j_1\rangle, \langle i_2, j_2\rangle) = \begin{cases} \langle i_1, j_1\rangle & \text{if } i_1 < i_2 \\ \langle i_2, j_2\rangle & \text{if } i_1 > i_2 \\ \langle i_1 \mathsf{min}\, i_2, j_1 \sqcap j_2\rangle & \text{otherwise} \end{cases}$$

$$\mathcal{B}[\![(\mathsf{0}<)\,]\!](\langle i, j\rangle) = \begin{cases} tt & \text{if } i > 0 \\ ff & \text{if } i < 0 \\ \bot & \text{otherwise} \end{cases}$$

The interpretation of the derivative operator is given by:

$$\mathcal{E}[\![\mathsf{D}x.e]\!]_\rho = \lambda d \in \mathcal{I} \times \mathcal{I}\,.\,\langle \pi_2(\mathcal{E}[\![e]\!]_\rho[\langle \pi_1 d, 1\rangle/x]), \bot\rangle$$

Note that the function $\mathcal{B}[\![(\mathsf{0}<)\,]\!]$ loses the information given by the derivative part, while the function $\mathcal{E}[\![\mathsf{D}x.e]\!]_\rho$, is a sort of translation of the function $\mathcal{E}[\![\lambda x.e]\!]_\rho$: The value of $\mathcal{E}[\![\mathsf{D}x.e]\!]_\rho$ is obtained from the derivative part of $\mathcal{E}[\![\lambda x.e]\!]_\rho$, while the derivative part of $\mathcal{E}[\![\mathsf{D}x.e]\!]_\rho$ is set to $\bot$.

Consider some examples. The absolute value function can be implemented through the term $\mathrm{Ab} = \lambda x.\mathsf{max}\,(\mathsf{opp}\,x)x$ with the following semantic interpretation:

$$\mathcal{E}[\![\mathrm{Ab}]\!]_\rho(\langle i, j\rangle) = \begin{cases} \langle i, j\rangle & \text{if } i > 0 \\ \langle -i, -j\rangle & \text{if } i < 0 \\ \langle [k^-, k^+], [-1, 1]j\rangle & \text{otherwise,} \end{cases}$$

where $k^- = \max(i^-, -i^+)$, $k^+ = \max(i^+, -i^-)$ with $i = [i^-, i^+]$.

When the absolute value function is evaluated at 0, where it is not differentiable, the derivative part of the semantic interpretation returns a partial value: $\pi_2(\mathcal{E}[\![\mathrm{Ab}]\!]_\rho(\{0\}, \{1\}) = [-1, 1]$. This partial value coincides with the Clarke gradient, equivalently the L-derivative, of the absolute value function.

The function $\frac{|x-y|}{2}$, is represented by the expression

$$\text{Ab-dif} = \lambda x.y.\mathsf{max}\,(x \mathsf{av}\, 1/2(\mathsf{opp}\,y))((\mathsf{opp}\,x)\mathsf{av}\, 1/2y)$$

whose semantics is the function:

$$\mathcal{E}[\![\text{Ab-dif}]\!]_\rho(\langle i_1, j_1\rangle, \langle i_2, j_2\rangle) =$$
$$\begin{cases} \langle \frac{i_1 - i_2}{2}, \frac{j_1 - j_2}{2}\rangle & \text{if } i_1 > i_2 \\ \langle \frac{i_2 - i_1}{2}, \frac{j_2 - j_1}{2}\rangle & \text{if } i_1 < j_1 \\ \langle [k^-, k^+], [-1/2, 1/2](j_1 - j_2)\rangle & \text{otherwise,} \end{cases},$$

where $k^- = \max(i_1^- - i_2^+, i_2^- - i_1^+)$ and $k^+ = \max(i_1^+ - i_2^-, i_2^+ - i_1^-)$.

From $[\![\text{Ab-dif}]\!]$ it is possible to evaluate the partial derivative of the function $\frac{|x-y|}{2}$, not only along the axes x and y, but along any direction. Considering the Euclidean distance, the derivative of the function at $(0, 0)$ in the direction of the unit vector $(u/\sqrt{u^2+v^2}, v/\sqrt{u^2+v^2})$ is given by
$\mathcal{E}[\![\text{Ab-dif}]\!]_\rho(\langle\{0\}, \{u/\sqrt{u^2+v^2}\}\rangle, \langle\{0\}, \{v/\sqrt{u^2+v^2}\}\rangle)$, that is the the interval $[-1/2, 1/2]\frac{u-v}{\sqrt{u^2+v^2}}$. Again this value coincides with the value of the Clarke gradient of the function $\frac{|x-y|}{2}$ at $(0, 0)$ in the direction $(u/\sqrt{u^2+v^2}, v/\sqrt{u^2+v^2})$.

3.1 Logical relations characterization

In the present approach we choose to define the semantic domains in the simplest possible way. As a consequence, our domains contain also points that are not consistent with the intended meaning, for example, the domain $\mathcal{D}_{\iota\to\iota} = (\mathcal{I}\times\mathcal{I}) \to (\mathcal{I}\times\mathcal{I})$ contains also the product of two functions $\langle f_1, f_2\rangle$ where the derivative part f_2 is not necessarily linear in its second argument and is not necessarily consistent with the value part, i.e., the function f_1; moreover the value part f_1 can be a function depending also on its second argument.

However the semantic interpretation of (non-expansive) PCDF expressions will not have this pathological behaviour. A proof of this fact and a more precise characterisation of the semantic interpretation of expressions can be obtained through the technique of logical relations [18]. In particular we define a set of logical relations on the semantic domains and prove that, for any non-expansive PCDF expression e, the semantic interpretation of e satisfies these relations. Using this method, we can establish a list of properties for the semantic interpretation of PCDF expressions.

Definition 1. *The following list of relations are defined on the domain $\mathcal{D}_\iota$.*

- **Independence***: A binary relation R^i_ι consisting of the pairs of the form $(\langle i, j_1\rangle, \langle i, j_2\rangle)$. The relation R^i_ι is used to establish that, for a given function, the value part of the result is independent from the derivative part of the argument: $f_1(i, j_1) = f_1(i, j_2)$.*
- **Sub-linearity***: A family of relations $R^{l,r}_\iota$ indexed by a rational number $r \in [-1, 1]$. The family $R^{l,r}_\iota$ consists of pairs of the form $(\langle i, j_1\rangle, \langle i, j_2\rangle)$ where $j_1 \sqsubseteq r \cdot j_2$. These relations are used to establish the sublinearity of the derivative part: $f_2(i, r \cdot j) \sqsubseteq r \cdot f_2(i, j)$.*
- **Consistency***: A family of ternary relation $R^{d,r}_\iota$ indexed by a rational number $r \in (0, 2]$, consisting of triples of the form $(\langle i_1, j_1\rangle, \langle i_2, j_2\rangle, \langle i_3, j_3\rangle)$ with $i_3 \sqsubseteq i_1 \sqcap i_2$ and $(r \cdot j_3)$ consistent with $(i_1 - i_2)$, that is the intervals $(r \cdot j_3)$ and $(i_1 - i_2)$ have a non-empty intersection. This relation is used to establish the consistency of the derivative part of a function with respect to the value part.*

The above relations are defined on the other ground domains $\mathcal{D}_o$ and $\mathcal{D}_\nu$ as the diagonal relations in two or three arguments, e.g., $R^{d,r}_\nu(l, m, n)$ iff $l = m = n$. The relations are extended inductively to higher order domains by the usual definition on logical relations: $R^i_{\sigma\to\tau}(f, g)$ iff for every $d_1, d_2 \in \mathcal{D}_\sigma$, $R^i_\sigma(d_1, d_2)$ implies $R^i_\tau(f(d_1), g(d_2))$, and similarly for the other relations.

Proposition 1. *For any closed expression $e : \sigma$, for any rational number $r \in [-1, 1]$, the semantic interpretation $\mathcal{E}[\![e]\!]_\rho$ of e, is self-related by R^i_σ, $R^{l,r}_\sigma$, i.e. $R^i_\sigma(\mathcal{E}[\![e]\!]_\rho, \mathcal{E}[\![e]\!]_\rho)$, and similarly for $R^{l,r}_\sigma$. Moreover, if the expression $e : \sigma$ is non-expansive, the semantic interpretation $\mathcal{E}[\![e]\!]_\rho$, is self-related by $R^{d,r}_\sigma$.*

We now show how the three relations ensure the three properties of independence, sublinearity and consistency. To any element $f = \langle f_1, f_2\rangle$ in the domain $\mathcal{D}_{\iota\to\iota} = (\mathcal{I}\times\mathcal{I}) \to (\mathcal{I}\times\mathcal{I})$ we associate a partial function $f_v : I \to I$ with

$$f_v(x) = \begin{cases} y & \text{if } f_1(\langle\{x\}, \bot\rangle) = \{y\} \\ \text{undefined} & \text{if } f_1(\langle\{x\}, \bot\rangle) \text{ is a proper interval} \end{cases}$$

and a total function

$$f_d : I \to \mathcal{I} = \lambda x.f_2(\langle\{x\}, \{1\}\rangle))$$

The preservation of the relations R^i_ι, $R^{l,r}_\iota$ has the following straightforward consequences:

Proposition 2. *(i) For any function $f = \langle f_1, f_2\rangle$ in $\mathcal{D}_{\iota\to\iota}$ self-related by $R^i_{\iota\to\iota}$, for every i, j_1, j_2, $f_1(\langle i, j_1\rangle) = f_1(\langle i, j_2\rangle)$, the return value part is independent from the derivative argument.*

(ii) For any function $f = \langle f_1, f_2\rangle$ in $\mathcal{D}_{\iota\to\iota}$ self-related by $R^{l,r}_{\iota\to\iota}$ for every i, j, and for every rational $r \in [-1, 1]$, $f_2(\langle i, r \cdot j\rangle) \sqsubseteq r \cdot f_2(\langle i, j\rangle)$. It follows that:

- *$(f_2(\langle i, \{r\}\rangle))/r \sqsubseteq f_2(\langle i, \{1\}\rangle)$, i.e., the most precise approximation of the L-derivative is obtained by evaluating the function with 1 as its second argument,*
- *for every i, j, $f_2(\langle i, -j\rangle) = -f_2(\langle i, j\rangle)$, i.e., the derivative part is an odd function.*

The preservation of the relation $R^{d,r}_\iota$ induces the following properties (see the Appendix for the proof):

Proposition 3. *For any function f in $\mathcal{D}_{\iota\to\iota}$ self-related by $R^{d,r}_{\iota\to\iota}$:*

(i) the function f_v is non-expansive;

(ii) on the open sets where the functions f_v is defined, the function f_d is an approximation to the L-derivative of the function f_v;

(iii) if f is a maximal element of $\mathcal{D}_{\iota\to\iota}$ then f_v is a total function and f_d is the associated L-derivative.

3.2 Subdomains

By definition, the logical relations are closed under directed lubs, and as a consequence the sets of elements self-related by them are also closed under directed lubs.

For any ground type σ the relations R^i_σ, $R^{l,r}_\sigma$, $R^{d,r}_\sigma$ are closed under arbitrary meets, meaning that if $\forall j \in J\ .\ R^i_\sigma(d_j, e_j)$ then $R^i_\sigma(\bigsqcap_{j\in J} d_j, \bigsqcap_{j\in J} e_j)$ and similarly for the other relations $R^{l,r}_\sigma$, $R^{d,r}_\sigma$. The proof is immediate for $\sigma = o, \nu$, and is a simple check for $\sigma = \iota$. The following result shows that this closure property holds also for $\sigma = \iota \to \iota$.

Proposition 4. *The set of elements in $\mathcal{D}_{\iota\to\iota}$ self-related by any of the three relations $R^i_{\iota\to\iota}$, $R^{l,r}_{\iota\to\iota}$, and $R^{d,r}_{\iota\to\iota}$ is closed under arbitrary meets.*

Proof. For the independence relation $R^{i}_{\iota\to\iota}$, the closure property is trivial to check. For the consistency relation $R^{d,r}_{\iota\to\iota}$, the closure under non-empty meets follows immediately from the fact that this relation is downward closed. The closure property for the sublinearity relation $R^{l,r}_{\iota\to\iota}$ is given in the Appendix.

We now employ the following result whose proof can be found in the Appendix.

Proposition 5. *In a continuous Scott domain, a non-empty subset closed under lubs of directed subsets and closed under non-empty meets is a continuous Scott subdomain.*

Corollary 1. *If σ is a ground type or first order type, then the set of elements in $\mathcal{D}_\sigma$ self-related by the three logical relations is a continuous Scott subdomain of $\mathcal{D}_\sigma$.*

As we do not deal with second or higher order real types in this extended abstract, we will not discuss the corresponding subdomains here.

3.3 Adequacy

As usual once an operational and denotational semantics are defined, it is necessary to present an adequacy theorem stating that the two semantics agree.

Let us denote by $[a,b] \ll Eval(e)$ the fact that there exits three integers l,m,n such that $e \to^\star \mathsf{dig}\, l\, m\, n\, e'$ and $[(l-m)/n,(l+m)/n] \subset (a,b)$. The proof of the following theorem is presented in the Appendix.

Theorem 1 (Adequacy). *For every closed term e with type ι, interval $[a,b]$ and environment ρ, we have:*

$$[a,b] \ll Eval(e) \text{ iff } [a,b] \ll \pi_1(\mathcal{E}[\![e]\!]_\rho)$$

In the operational semantics that we have proposed, the calculus of the derivative is performed through a sort of symbolic computation: the rewriting rules specify how to evaluate the derivative of the primitive functions and the application of the derivative rules essentially transforms a function expression into the function expression representing the derivative. The denotational semantics provides an alternative approach to the computation of the derivative, which almost exactly coincides with the computation performed by Automatic Differentiation. We can interpret our adequacy result as a proof that symbolic computation of the derivative and the computation of the derivative through Automatic Differentiation coincide. We remark in passing that, inspired by the denotational semantics, it is possible to define an alternative operational semantics that will perform the computation of the derivative in the same way that is performed by Automatic Differentiation.

3.4 Function definability

We will show in the following theorem that any computable Lipschitz function can be obtained in our framework as the limit, in the sup norm, of a sequence of piecewise linear maps definable in PCDF such that every piecewise linear map in the sequence gives lower and upper bounds for the function and the L-derivative of the function is contained in the L-derivatives of the piecewise linear maps, which converge to the classical derivative of the function wherever it is continuously differentiable.

Theorem 2. *For any maximal computable function f in $\mathcal{D}_{\iota\to\iota}$ preserving the logical relations $R^{i}_{\iota\to\iota}$, $R^{l,r}_{\iota\to\iota}$, $R^{d,r}_{\iota\to\iota}$, there exists a closed PCDF expression* f *such that:*

$$\forall x \in I.\ f_v(x) = (\mathcal{E}[\![\mathsf{f}]\!]_\rho)_v(x) \ \wedge\ f_d(x) = (\mathcal{E}[\![\mathsf{f}]\!]_\rho)_d(x)$$

The above definability result states that if we consider only the behaviour of the domain functions on the total elements of $\mathcal{D}_\iota$ (i.e. the elements representing completely defined real numbers) then PCDF is sufficiently rich to represent the computable elements of $\mathcal{D}_{\iota\to\iota}$.

We do not consider the problem of defining PCDF expressions whose semantics coincides with domain functions also on partial elements. The reason for this choice is that this later problem is technically more difficult and less interesting from a practical point of view.

The proof of the above result is quite lengthy: we define a general methodology to transform the information that can be extracted from a domain function into a PCDF expression. The Appendix contains a description of the construction.

4 Conclusion

We have integrated, in a single language, exact real number computation with the evaluation of the derivatives of function expressions.

The language has been designed using a minimal set of primitives sufficient to define any computable (and differentiable) function. It can be seen as a theoretical basis for the implementation of exact real number computation in a programming language. In a practical implementation, however, one needs both to extend the set of primitive functions and to carefully redesign the reduction strategy to increase both the usability of the language and the efficiency of the computation.

The main result presented here is an adequate denotational semantics for differentiable functions, which has required original ideas in developing the semantics domains, and a definability result showing the expressivity of the language.

The present research can be extended in several directions. Some possible future works are the following.

- An obvious problem to consider is whether the definability result presented in the paper can be extended to a larger class of function domains. We claim

that the techniques presented here can be easily adapted to functions with several arguments. This is not however the case when considering higher order functions, whose definability is an open problem.
– A second direction for possible further research is the treatment of the second derivative and more generally derivative of arbitrary order.

References

1. E. Bishop and D. Bridges. *Constructive Analysis*. Springer-Verlag, 1985.
2. F. H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley, 1983.
3. E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, 1955.
4. T. A. Davis and K. Sigmon. *MATLAB Primer*. CRC Press, seventh edition, 2005.
5. P. Di Gianantonio. An abstract data type for real numbers. *Theoretical Computer Science*, 221:295-326, 1999.
6. A. Edalat. A continuous derivative for real-valued functions. In *New Computational Paradigms, Changing Conceptions of What is Computable*, pages 493–519. Springer, 2008.
7. A. Edalat. A differential operator and weak topology for Lipschitz maps. *Topology and its Applications*, 157,(9):1629-1650, June 2010.
8. A. Edalat, M. Escardó Integration in Real PCF. *Information and Computation*, 160:128–166, 2000.
9. A. Edalat, A. Lieutier. Domain theory and differential calculus (Functions of one variable). *Mathematical Structures in Computer Science*, 14(6):771–802, December 2004.
10. A. Edalat, A. Lieutier, and D. Pattinson. A computational model for multi-variable differential calculus. In *Proc. FoSSaCS 2005*, volume 3441 of *LNCS*, 3441: 505–519, 2005.
11. Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3), 2003.
12. M. H. Escardó. PCF extended with real numbers. *Theoretical Computer Science*, 162(1):79–115, August 1996.
13. A. Griewank and A. Walther. *Evaluating Derivatives*. Siam, second edition, 2008.
14. `www.doc.ic.ac.uk/exact-computation/`.
15. K. Ko. *Complexity Theory of Real Numbers*. Birkhäuser, 1991.
16. G. Lebourg. Generic differentiability of lipschitzian functions. *Transaction of AMS*, 256:125–144, 1979.
17. O. Manzyiuk. A simply typed lambda calculus for forward automatic differentiation. In *Proc MFPS 12*, *ENTCS* 259–273, 2012.
18. J. C. Mitchell. *Foundations of Programming Languages*. MIT Press, 1996.
19. P. J. Potts, A. Edalat, and M. Escardó. Semantics of exact real arithmetic. In *Twelfth Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1997.
20. M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics*. Springer-Verlag, 1988.
21. K. Weihrauch. *Computable Analysis (An Introduction)*. Springer, 2000.

Appendix

We give the details of several proofs.

4.1 Proof of Proposition 1

Proof. The proof is quite standard, and is based on the fact that the relations R^i_σ, $R^{l,r}_\sigma$, $R^{d,r}_\sigma$ are logical. First one proves that the semantic interpretation of (non-expansive) constants are self-related by $R^{d,r}_\sigma$, R^i_σ, and $R^{l,r}_\sigma$. Then, to show that the fixed-point operator preserves the relations, one shows that the bottom elements are self-related by R^i_σ, $R^{l,r}_\sigma$, and $R^{d,r}_\sigma$, and that the relations are closed under the lub of chains. Finally, by the basic lemma of logical relations, one obtains the result. □

4.2 Proof of Proposition 3

Proof. (i) Let x and y be two real numbers for which the function f_v is defined. For any rational $r \geq |x - y|$ we have that
$R^{d,r}_\iota(\langle\{x\},[-1,1]\rangle, \langle\{y\},[-1,1]\rangle, \langle[x,y],[-1,1]\rangle)$. Therefore
$R^{d,r}_\iota(f(\langle\{x\},[-1,1]\rangle), f(\langle\{y\},[-1,1]\rangle), f(\langle[x,y],[-1,1]\rangle))$, which implies that
$f_v(x) - f_v(y) \in r \cdot f_2(\langle[x,y],[-1,1]\rangle)$, and thus $-1 \leq \frac{f_v(x)-f_v(y)}{x-y} \leq 1$.
(ii) Given any $x \in I$ and any open interval O containing $f_d(x) = f_2(\langle\{x,\},\{1\}\rangle)$, let $[a,b] \ll \{x\}$ be a rational interval such that f_v is define on $[a,b]$ and $f_2(\langle[a,b],\{1\}\rangle) \subseteq O$, and let $r = b - a$, we have
$R^{d,r}_\iota(\langle\{b\},[-1,1]\rangle, \langle\{a\},[-1,1]\rangle, \langle[a,b],\{1\}\rangle)$, by repeating the arguments of the previous point, it follows $\frac{f_v(b)-f_v(a)}{b-a} \in f_2(\langle[a,b],\{1\}\rangle)$. By monotonicity of f it follows that for any pair of rationals $a', b' \in (a,b)$, we have: $\frac{f_v(b')-f_v(a')}{b'-a'} \in f_2(\langle[a,b],\{1\}\rangle)$, and by continuity of f_v for any pair of real numbers $x, y \in (a,b)$ $\frac{f_v(x)-f_v(y)}{x-y} \in O$.
(iii) If f is a maximal element $\mathcal{D}_{\iota\to\iota}$, by point (i) the function f_v is non-expansive on the points where it is defined. It follows that if the function f_v is not defined at a given point x, it is always possible to construct a function f° such that $f \sqsubseteq f^\circ$ and f°_v defined on x, leading to a contradiction. Similar arguments can be used to prove that f_d is the L-derivative of f_v and not only an approximation of the L-derivative.

4.3 Proof of Proposition 4

The sublinearity relation $R^{l,r}_{\iota\to\iota}$ is closed under non-empty meets.

Proof. To show that sublinearity is closed under meets, assume that $f_k : \mathcal{I} \to \mathcal{I}$ with $k \in K$ is a family of Scott continuous functions satisfying the sublinearity $f_k(r[x,y]) \geq r f_k([x,y])$ for all $[x,y] \in \mathcal{I}$ and some (rational) $r \in [-1,1]$. We show that the meet $\bigsqcap_k f_k$ will also be sublinear.

We use the lower and upper parts of any $f : \mathcal{I} \to \mathcal{I}$ as $f^-, f^+ : T \to [-1,1]$ where $T = \{(x,y) \in [-1,1] \times [-1,1] : x \leq y\}$. Note that f^- and f^+ are lower and upper semi-continuous respectively. Sublinearity of f is equivalent to $f^+(r(x,y)) \geq rf^+((x,y))$ and $f^-(r(x,y)) \leq rf^-((x,y))$ for $r \in [-1,1]$ and $(x,y) \in T$.

We have: $(\bigsqcap_k f_i)^+ = g$ with $g = \limsup g_0$ where $g_0 = \sup_{k \in K} f_k^+$ and similarly $(\bigsqcap_k f_i)^- = h$ with $h = \liminf h_0$ where $h_0 = \inf_{k \in K} f_k^-$.

The sublinearity condition for f_k is equivalent to $f_k^+(r(x,y)) \geq rf_k^+((x,y))$ and $f_k^-(r(x,y)) \leq rf_k^-((x,y))$ for $r \in [-1,1]$ and $(x,y) \in T$.

By taking pointwise sup and inf respectively we get: $g_0(r(x,y)) \geq rg_0((x,y))$ and $h_0(r(x,y)) \leq rh_0((x,y))$. By taking limsup and liminf respectively we obtain: $g(r(x,y)) \geq rg((x,y))$ and $h(r(x,y)) \leq rh((x,y))$ as required.

4.4 Proof of Proposition 5

In a continuous Scott domain, a non-empty subset closed under lubs of direct subsets and closed under non-empty meets is a continuous Scott subdomain.

Proof. Let D be a continuous Scott domain and $C \subset D$ a non-empty subset with the above closure properties. Given an element of $d \in D$, denote by $i(d)$ the greatest lower bound (meet) in C of the set $\{c \mid d \sqsubseteq c, c \in C\}$, if this set is not empty, otherwise let $i(d)$ be undefined.

Then i, regarded as a partial function from D to C, preserves the well below relation $\ll$. In fact given two elements $x, y \in D$ with $y \ll_D x$ and $i(x)$ defined, we check that $i(y) \ll_C i(x)$. Let A be a directed subset of elements in C, with $i(x) \sqsubseteq \bigsqcup_C A$. Since C is closed under lub of directed sets, $\bigsqcup_C A = \bigsqcup_D A$, and by construction of i, we have $x \sqsubseteq i(x)$. Thus, $x \sqsubseteq \bigsqcup_D C$, and, by hypothesis, there exists $a \in A$ such that $y \sqsubseteq a$. Since i is monotone and coincides with the identity on the elements of C, we have $i(y) \sqsubseteq i(a) = a$ and therefore $i(y) \ll_C i(x)$. It follows that if a set B is a basis of D then $i(B)$ forms is a basis for C. In fact given an element $x \in C$, the set $A = \{a \in B \mid a \ll x\}$, is a directed set with lub x, then $i(A)$ is a directed set of elements well below $i(x) = x$, having x as lub. Therefore C is a continuous dcpo, and since it has non-empty meets, it is also consistently complete.

4.5 Proof of Adequacy Theorem 1

For every closed term e with type ι and environment ρ, we have:

$$[a,b] \ll Eval(e) \text{ iff } [a,b] \ll \pi_1(\mathcal{E}[\![e]\!]_\rho)$$

Proof. We use the technique of computability predicates to prove both the soundness and the completeness of the operational semantics. Note that the soundness of the operational semantics cannot be proved by simply showing that the reduction rules preserve the denotational semantics, since this is simply not true. A simple example being the expression $(\mathsf{D}x.\,\mathsf{dig}\,0\,1\,2\,e_1)\,e_2$ that reduces

to $\mathsf{dig}\,0\,1\,2\,((\mathsf{D}x.\,e_1)\,e_2)$. The elements $\mathcal{E}[\![(\mathsf{D}x.\,\mathsf{dig}\,0\,1\,2\,e_1)\,e_2]\!]_\rho$ and $\mathcal{E}[\![\mathsf{dig}\,0\,1\,2\,((\mathsf{D}x.\,e_1)\,e_2)]\!]_\rho$ do not coincide on their second component (the first is $\bot$, the other is above $[-1/2, 1/2]$. More generally, all the reduction rules for the derivative operator do not preserve the semantics on the second element.

We define a computability predicate Comp for closed terms of type o, ν by requiring that the denotational and operational semantics coincide, in the usual way. A closed term e having type ι satisfies the predicate Comp_ι if for every closed rational interval $[a, b]$ and environment ρ we have: $[a, b] \ll Eval(e)$ iff $[a, b] \ll \pi_1(\mathcal{E}[\![e]\!]_\rho)$

The computability predicate is then extended, by induction on types, to closed elements of any type, and, by closure, to arbitrary elements.

Using the standard techniques for computability predicate, it is possible to prove that all constants are computable, and that λ-abstraction preserves the computability of the expressions. Therefore all expressions not containing the derivative operator are computable.

To prove that the computability predicate is satisfied by expressions containing the derivative operator, we show, by structural induction on the non-expansive expression e, that the expression $\mathsf{D}x.e$ is also computable.

The proof considers many cases. As an example, we take the case where $e = \mathsf{min}\,e_1e_2$. By the induction hypothesis we can assume the computability of $\mathsf{D}x.e_1$ and $\mathsf{D}x.e_2$. Since the expressions e_1, e_2, pif and av do not contain the derivative operator, we can assume that they are computable.

We need to prove, for any computable expression e', that
$[a, b] \ll Eval((\mathsf{D}x.\mathsf{min}\,e_1e_2)e')$ iff $[a, b] \ll \pi_1(\mathcal{E}[\![(\mathsf{D}x.\mathsf{min}\,e_1e_2)e']\!]_\rho)$.

On the one hand, we have the following chain of implications:
$[a, b] \ll Eval((\mathsf{D}x.\mathsf{min}\,e_1e_2)e')$ iff, by the denotational semantics rules,
$[a, b] \ll Eval(\mathsf{pif}\ (0<)\ (\mathsf{av}\,1\,2\,(\mathsf{opp}\,e_1[e'/x])(e_2[e'/x]))$
$\qquad\qquad \mathsf{then}\ (\mathsf{D}x.\,e_1)e'\ \mathsf{else}\ (\mathsf{D}x.\,e_2)e')$
iff, by computability of the expression right hand side
$[a, b] \ll \pi_1(\mathcal{E}[\![\mathsf{pif}\ \ (0<)\ (\mathsf{av}\,0\,1\,2\,(\mathsf{opp}\,e_1[e'/x])\,(e_2[e'/x]))$
$\qquad\qquad \mathsf{then}\ (\mathsf{D}x.\,e_1)e'\ \mathsf{else}\ (\mathsf{D}x.\,e_2)e']\!]_\rho)$

If we pose: $\mathcal{E}[\![e']\!]_\rho = \langle i, j\rangle$, $\mathcal{E}[\![e_1]\!]_{\rho[\langle i,1\rangle/x]} = \langle i_1, j_1\rangle$, $\mathcal{E}[\![e_1]\!]_{\rho[\langle i,j\rangle/x]} = \langle i'_1, j'_1\rangle$, $\mathcal{E}[\![e_2]\!]_{\rho[\langle i,1\rangle/x]} = \langle i_2, j_2\rangle$, $\mathcal{E}[\![e_2]\!]_{\rho[\langle i,j\rangle/x]} = \langle i'_2, j'_2\rangle$.

by applying the denotational semantics rule we can derive that right hand side in the last relation is equal to j_1 if $i'_1 < i'_2$, to j_2 if $i'_2 < i'_1$, and to $j_1 \sqcup j_2$ otherwise.

On the other hand, by the rules of denotational semantics:
$\pi_1(\mathcal{E}[\![(\mathsf{D}x.\mathsf{min}\,e_1e_2)e]\!]_\rho) = \pi_2(\mathcal{E}[\![\mathsf{min}\,e_1e_2]\!]_{\rho[i,1\rangle/x]})$ which is equal to j_1 if $i_1 < i_2$, to j_2 if $i_2 < i_1$, and to $j_1 \sqcup j_2$ otherwise.

Since $\mathcal{E}[\![e_1]\!]$ and $\mathcal{E}[\![e_2]\!]$ are self-related by R^i_σ, their value parts are independent from the derivative part so $i_1 = i'_1$ and $i_2 = i'_2$, from which the result follows.

The other cases can be proved in a similar way. □

4.6 Function definability

Here we will not give a detailed proof but present a general construction that can be used to define, inside PCDF, any computable non-expansive function. The presentation is quite lengthy and proceeds incrementally showing, in several steps, how to define larger and larger classes of computable maximal elements in $\mathcal{D}_{\iota\to\iota}$. Each step will introduce a new ingredient in the construction. More precisely, we first present a construction that can deal with any piecewise continuously differentiable function (i.e., a function that is continuously differentiable except for a finite number of points at which the left and right derivatives exist), then we extend it to treat functions that are piecewise continuously differentiable except for a finite number of points (of essential discontinuities of the derivative at which the left and right derivatives do not exist), and finally we give a definability result for general Lipschitz maps.

Notation. Given two real numbers x, r we denote with $x \pm r$ the interval having center in x and diameter $2r$. Given a total function f, we denote by $f \pm r$ the partial function $\lambda x.f(x) \pm r$. Moreover, given a rational number c we denote with $\oplus_c$ the weighted average operation on reals, that is $x \oplus_c y = (1-c)\cdot x + c\cdot y$, and with B_c the functional
$\lambda l, f : I \to I.\, \lambda x : I.\, \max(l(x) - c, \min(f(x), l(x) + c))$.
With some abuse of notation given a PCDF constant c representing a function on reals, we will use the symbol c to denote the functional obtained by pointwise application of the function c. For example, min denotes the functional
$\lambda f, g : \iota \to \iota\,.\,\lambda x : \iota\,.\,\mathsf{min}\,(fx)(gx)$.

For start, we present a series of functions, and functionals definable by PCDF expressions.

- It is easy to see that any non-expansive piecewise rational linear function l is definable using the functions $\mathsf{dig}\,, \mathsf{opp}\,, \mathsf{add}\,, \mathsf{min}\,, \mathsf{max}\,$, in the sense that there exists a PCDF function expression l such that $l = (\mathcal{E}[\![\mathsf{l}]\!]_\rho)_v$ and $\frac{dl}{dx} = (\mathcal{E}[\![\mathsf{l}]\!]_\rho)_d$. For example a piecewise linear interpolation of the function $\lambda x.x^3/2$ coinciding with the function on the points with x equal to $-1, -1/2, 0, 1/2, 1$ can be defined as

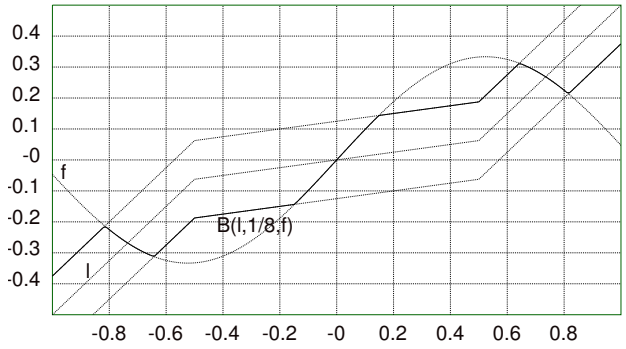
$$\begin{array}{l}\lambda x.\mathsf{max}\ \ (\mathsf{min}\,(\mathsf{add}\,3\,8\,(\mathsf{dig}\,0\,7\,8\,x))(\mathsf{dig}\,0\,1\,8\,x))\\ \qquad\qquad (\mathsf{sub}\,3\,8\,(\mathsf{dig}\,0\,7\,8\,x))\end{array}$$

  We use $\mathsf{l}_1, \mathsf{l}_2, \ldots$ as metavariables over expressions defining piecewise rational linear functions, with $l_1, l_2, \ldots$ denoting the corresponding functions on reals, i.e., $l_1 = (\mathcal{E}[\![\mathsf{l}_1]\!]_\rho)_v$
- In the following we will use the functional:

$$\begin{array}{l}\mathsf{B} = \lambda l : \iota \to \iota.\,\lambda p, q : \nu.\,\lambda f : \iota \to \iota.\,\lambda x : \iota.\\ \qquad \mathsf{max}\,(\mathsf{sub}\,p\,q\,(l\,x))(\mathsf{min}\,(f\,x)(\mathsf{add}\,p\,q\,(l\,x))).\end{array}$$

  Note that, given an expression l defining a (piecewise linear) function l, $\mathcal{E}[\![\mathsf{B}\,\mathsf{l}\,\mathsf{p}\,\mathsf{q}]\!]_\rho fx$ is the interval $(l(x) \pm \frac{p}{q}) \cap f(x)$, if the interval $l(x) \pm \frac{p}{q}$ and

$f(x)$ intersect, otherwise $\mathcal{E}[\![\mathsf{B\,l\,p\,q}]\!]_\rho f x$ coincides with one of the two bounds of the interval $l(x) \pm \frac{p}{q}$. The following diagram illustrates the behaviour of the functional B on maximal points for a function preserving these points.



On partial elements, $\mathcal{E}[\![\mathsf{B\,l\,p\,q}]\!]_\rho f$ is a sort of *projection* of the function f on the function $\lambda x\,.\,l(x) \pm \frac{p}{q}$. Given an expression Ω denoting the completely undefined function, the value part of $\mathcal{E}[\![\mathsf{B\,l\,p\,q}\;\Omega]\!]_\rho)$ is the function $\lambda x\,.\,l(x) \pm \frac{p}{q}$, while the derivative part $(\mathcal{E}[\![\mathsf{B\,l\,p\,q}\Omega]\!]_\rho)_d$ is the completely undefined function.

– We will use also the functional:

$$\mathsf{L} = \lambda l : \iota \to \iota.\, \lambda p, q : \nu.\, \lambda f : \iota \to \iota.\, \lambda x : \iota.(\mathsf{av}\,(q - p)\,q\,(f\,x)\,x).$$

Given an expression l defining a piecewise linear function l, it is readily seen that the value part of $\mathcal{E}[\![\mathsf{L\,l\,p\,q}\Omega_{\iota\to\iota}]\!]_\rho$ is the function $\lambda x\,.\,(1 - \frac{p}{q}) \cdot l(x) \pm \frac{p}{q}$, while $(\mathcal{E}[\![\mathsf{L}\,l\,p\,q\,\Omega_{\iota\to\iota}]\!]_\rho)_d$ is the function $\lambda x\,.\,(1 - \frac{p}{q}) \cdot \frac{d\,l}{d\,x}(x) \pm \frac{p}{q}$.

It is easy to show that for any non-expansive function $f : I \to I$ there exists a sequence of piecewise linear functions $\langle l_i \rangle_{i\in\mathbb{N}}$ converging fast to f, in the sense that for any i, we have $f \in l_i \pm 2^{-i+1}$.

Let $\mathsf{exp} : \nu \to \nu$ be a suitable PCDF term implementing the function $\lambda n.\, 2^n$, if the sequence of piecewise linear functions $\langle l_i \rangle_{i\in\mathbb{N}}$ is definable in the sense that there exists a term l such that $\mathsf{l\,n}$ defines the function l_n, then the term $\mathsf{f} = (\mathsf{Y}\ \lambda\mathsf{F}.\lambda\mathsf{n}.\mathsf{B}(\mathsf{l\,n})\,\mathsf{1}\,(\mathsf{exp}\,(\mathsf{n}+\mathsf{1}))\,(\mathsf{F}(\mathsf{n}+\mathsf{1})))\mathsf{0}$ is such that:

$$\mathcal{E}[\![\mathsf{f}]\!]_\rho = \bigsqcup_{i\in\mathbb{N}} \mathcal{E}[\![\mathsf{B}(\mathsf{l\,0})\,\mathsf{1\,2}\,(\mathsf{B}(\mathsf{l\,1})\,\mathsf{1\,4}\,(\ldots\mathsf{B}(\mathsf{l\,i})\,\mathsf{1\,2}^{\mathsf{i}+\mathsf{1}}\,\Omega)\ldots)]\!]_\rho$$

It is then not difficult to see that $f = (\mathcal{E}[\![\mathsf{f}]\!]_\rho)_v$. However, $(\mathcal{E}[\![\mathsf{f}]\!]_\rho)_d$ is the bottom function, i.e., the completely undefined approximation of the derivative of the function f. We now proceed in three steps of increasing complexity to define various classes of Lipschitz functions in PCDF.

4.7 Piecewise continuously differentiable

If $f : I \to I$ is piecewise continuously differentiable, then there exists a sequence of piecewise linear functions $\langle l_i \rangle_{i \in \mathbb{N}}$ such that for all i the function $l_1 \oplus_{1/2} (l_2 \oplus_{1/2} (\ldots l_i \oplus_{1/2} 0) \ldots)$ approximates the function f with precision 2^{-i}, both for the value and for the derivative part. If the sequence of piecewise linear function is definable by a term l then we can construct a term f such that:

$$\mathcal{E}[\![\mathsf{f}]\!]_\rho = \bigsqcup_{i \in \mathbb{N}} \mathcal{E}[\![\mathsf{L}(\mathsf{l}\,0)\,1\,2(\mathsf{L}(\mathsf{l}\,1)\,1\,2(\ldots \mathsf{L}(\mathsf{l}\,\mathsf{i})\,1\,2(\Omega)\ldots))]\!]_\rho$$

and one can prove that $\mathcal{E}[\![\mathsf{f}]\!]_\rho$ describes both the value part and the derivative part of f.

4.8 Piecewise continuously differentiable except for isolated points

The above construction can be applied only if the function $f : I \to I$, together with its derivative, is globally approximable by a sequence of piecewise linear functions. In general, if f is not piecewise continuously differentiable, this is not always possible. For example consider $f(x) = x^2 \cdot \sin(1/x)/4$, in $[-1, 1]$. Then f has a Lipschitz constant $3/4$ and is differentiable at every point, but in any neighbourhood of 0 its derivative assumes all the values between $-1/4$, $1/4$, i.e., the left and right derivatives at 0 do not exist. It follows that there is no piecewise linear function, whose derivative part approximates the derivative of part of f with an error smaller that $1/4$. To overcome this, we now present a construction where the problem of defining a function on the whole interval $[-1, 1]$ is reduced to the problem of defining suitable approximations to the function on smaller and smaller intervals.

It works as follows: given a non-expansive function $f : I \to I$, we first obtain a piecewise linear function $l_{0,0}$, and a rational number $c_{0,0} \in [0, 1)$ such that $(1 - c_{0,0}) \cdot l_{0,0}$ globally approximates the value and derivative part of f with an error $c_{0,0}$. Given an expression $\mathsf{l}_{0,0}$ defining the function $l_{0,0}$, an expression defining f can be written in the form $\mathsf{av}\,(\mathsf{q} - \mathsf{p})\,\mathsf{q}\,\mathsf{l}_{0,0}\,\mathsf{g}_{0,0}$, where $c_{0,0} = p/q$ and $\mathsf{g}_{0,0}$ is a suitable expression defining the non-expansive function $g_{0,0} = (f - (1 - c_{0,0}) \cdot l_{0,0})/c_{0,0}$. In other words we reduce the problem of defining f to the problem of defining $g_{0,0}$. At this stage we do not look for a global piecewise linear approximation of $g_{0,0}$, but we split the domain of $g_{0,0}$ in two overlapping intervals $J_{1,0}$ and $J_{1,0}$, and consider two functions $f_{1,0}$ and $f_{1,1}$ defined as the least non-expansive functions that coincide with $g_{0,0}$ on the intervals $J_{1,0}$ and $J_{1,0}$ respectively. The function $g_{0,0}$ can then be expressed as $\mathsf{max}\,(f_{1,0}, f_{1,1})$. In this way, the problem of defining $g_{0,0}$, it is split into the problem of defining two functions $f_{1,0}, f_{1,1}$ each of them having a complex behaviour just in one restricted part of the domain and in the remaining part behaving as piecewise linear functions. Corecursively, we apply the apply the procedure consider for the function f to the functions $f_{1,0}$ and $f_{1,1}$, constructing an infinitary tree of linear approximations, each of which considers the behaviour of the function f in smaller and smaller intervals.

A more formal presentation of the construction is the following.

First we define two sequences of coverings, I_i, J_i, with $i > 0$, of the interval I by rational intervals. To any pair i, j of non-negative integers with $2^i > j \geq 0$, we associate the real intervals

$$I_{i,j} = [(j - 2^{i-1})/2^{i-1}, (j + 1 - 2^{i-1})/2^{i-1}],$$

and

$$J_{i,j} = [(2j - 1 - 2^i)/2^i, (2j + 3 - 2^i)/2^i] \cap [-1, 1].$$

As a numerical example, the covering I_2 is formed by the intervals

$$[-1, -1/2], [-1/2, 0], [0, 1/2], [1/2, 1]$$

while the overlapping covering J_2 is formed by the intervals

$$[-1, -1/4], [-3/4, 1/4], [-1/4, 3/4], [1/4, 1].$$

By simultaneous induction on $i \geq 0$ we construct three families of double indexed maps $f_{i,j}$, $l_{i,j}$ and $g_{i,j}$, and a double indexed family of rational $c_{i,j}$ as follows:

- A family of functions $f_{i,j}$ from I to I, with $0 \leq i$ and $0 \leq j < 2^i$, is defined by:
  - $f_{0,0} = f_v$
  - $f_{i+1,j}$ is the smallest (wrt the real line order) non-expansive function coinciding with the $g_{i,\lfloor j/2 \rfloor}$ on the interval $J_{i+1,j}$, formally:
    $f_{i+1,j}(x) = \mathsf{min}\,(g_{i,\lfloor j/2 \rfloor}(x),\, x + a_{i+1,j},\, -x + b_{i+1,j})$.
    where, denoting by $J^-_{i,j}$ and $J^+_{i,j}$, respectively, the left and right bound of the interval $J_{i,j}$, we put $a_{i,j} = g_{i,\lfloor j/2 \rfloor}(J^-_{i,j}) - J^-_{i,j}$ and $b_{i,j} = g_{i,\lfloor j/2 \rfloor}(J^+_{i,j}) + J^+_{i,j}$.

  As anticipated above, the definability of $g_{i,j}$ to the definability of $f_{i+1,2j}$ and $f_{i+1,2j+1}$, each of them consider a different region portion of the function domain of $g_{i,j}$.
- A family of piecewise linear functions $l_{i,j}$ and the rational numbers $c_{i,j} \in (0, 1]$, with $0 \leq i$ and $0 \leq j < 2^i$, such that: $f_{i,j} \in (1 - c_{i,j}) \cdot l_{i,j} \pm c_{i,j}$ and $\frac{d\, f_{i,j}}{d\, x} \in (1 - c_{i,j}) \cdot \frac{d\, l_{i,j}}{d\, x} \pm c_{i,j}$.

  The functions $l_{i,j}$ and the rationals $c_{i,j}$ are not uniquely defined, the construction just chooses them in such a way that $(1 - c_{i,j}) \cdot l_{i,j}$ is a piecewise approximation of, value and derivative part of, $f_{i,j}$, with error $c_{i,j}$.
- The family of functions $g_{i,j}$ from I to I, with $0 \leq i$ and $0 \leq j < 2^i$ are defined such that $f_{i,j} = l_{i,j} \oplus_{c_{i,j}} g_{i,j}$; the conditions pose on the function $l_{i,j}$ assure that the function $g_{i,j}$ exists and it is non-expansive.

  After having generated the approximation $l_{i,j}$ of the functions $f_{i,j}$, one is left with the problem of defining the function $g_{i,j}$.

As an example of the above construction, consider the function $f = x^2/2$. We can choose, in the first step of approximation, the function $l_{0,0}(x) = \mathsf{max}\,(-x, x)$ and the constant $c_{0,0} = 1/2$. This choice induces the functions
$g_{0,0}(x) = \mathsf{min}\,((x^2+x), (x^2-x))$ and $f_{1,0}(x) = \mathsf{min}\,((x^2+x), (x^2-x), (-x+1/4)$. Proceeding with the construction, using similar choices for the next steps, leads to the function
$f_{2,1}(x) = \mathsf{min}\,((2x^2 + 3x + 1), (2x^2 + x), (2x^2 - x), (x + 5/8), (-x + 1/8))$. A piecewise linear approximation of function $f_{2,1}$, with precision $1/2$ is given by the function $l_{2,1}(x) = \mathsf{max}\,(\mathsf{min}\,(x + 1/2,\ -x - 1/2),\ \mathsf{min}\,(x, -x))$. The following diagram depicts the functions $f_{2,1}$ and $l_{2,1}/2$.

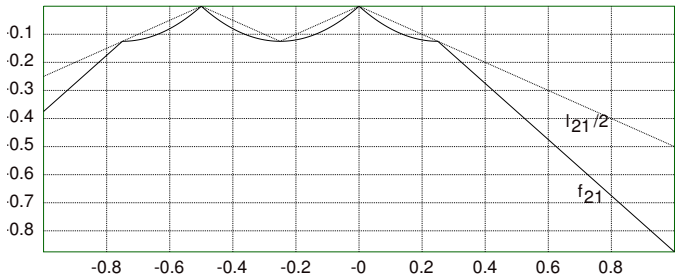


Fig. 1. Functions used in approximating the square function

The function $g_{2,1}$ with $f_{2,1} = l_{2,1}/2 + g_{2,1}/2$ and the function $f_{3,2}(x) = \mathsf{min}\,((4x^2 + 5x + 3/2), (4x^2 + 3x + 1/2), (4x^2 + x), (x + 9/16), (-x - 3/16))$ are illustrated by the following diagram:

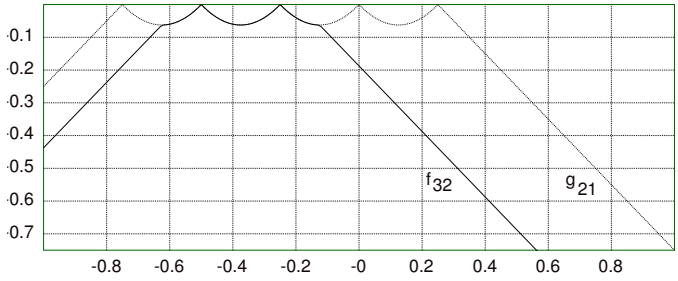


Fig. 2. Approximation of the square function

Coming back to the general construction, at any point on the interval I, we have that $f_{i,j} \geq (1 - c_{i,j}) \cdot l_{i,j} + c_{i,j} \cdot \max(f_{i+1,2j}, f_{i+2,2j})$ while on the interval $J_{i+1,2j} \cap J_{i+1,2j+1}$ equality holds: $f_{i,j} = (1-c_{i,j}) \cdot l_{i,j} + c_{i,j} \cdot \max(f_{i+1,2j}, f_{i+2,2j})$. Thus, the following infinitary form gives a correct approximation of the function

f:

$$l_{0,0} \oplus_{c_{0,0}} \max((l_{1,0} \oplus_{c_{1,0}} \max((l_{2,0} \oplus_{c_{2,0}} \max ldots), \\ (l_{2,1} \oplus_{c_{2,1}} \max ldots))), \\ (l_{1,1} \oplus_{c_{1,1}} \max((l_{2,2} \oplus_{c_{2,2}} \max \ldots), \\ (l_{2,3} \oplus_{c_{2,3}} \max \ldots)))).$$

If the families $c_{i,j}$ and $l_{i,j}$ are definable, then it is possible to construct a PCDF expression whose semantics coincides with the formula. Given a real number $x \in I$, denote with $\langle J_{i,h(i)} \rangle_{i \in \mathbb{N}}$ a sequence of J intervals converging to x such that $\forall i.h(i) = \lfloor h(i+1)/2 \rfloor$ (if x is not a dyadic rational this sequence is unique, if x is a dyadic rationals there are two such a sequences). The above formula defines a function converging on x iff $\Pi_{i \in \mathbb{N}} c_{i,h(i)} = 0$, for any such a sequence (each level reduces the inaccuracy by a factor $c_{i,j}$). If there exists an index k such that the function f is continuously differentiable in any interval in the form $J_{k,j}$ containing x, then on these intervals f can be approximated with arbitrary precision by a piecewise linear function and therefore there exists a choice for the constants $c_{i,j}$ making the above construction converge on x. But if x is a point of essential discontinuity for the derivative, there is a limit on the level of the precision for any choice for the constants $c_{i,j}$, and we need to consider the next construction to obtain convergence to the value of the function and its derivative at x.

4.9 General Lipschitz functions

In the previous construction, the finite approximations of the above displayed formula define both the value part and the derivative part of the function with the same level of precision. But there are non-expansive functions whose Clarke gradients (L-derivatives) are partial elements at all points [16, 7]. When applied to this class of functions the above construction can only lead to expressions whose semantics is a partial function also for the value part. To define functions in this class, we have to add an extra ingredient to the construction and to use the "projection" operator B, which increases the information contained in the value part of the partial function without necessarily modifying the information contained in the derivative part. To apply the operator B, it is necessary to build a list of piecewise linear functions $l'_{i,j}$ and rational numbers $c'_{i,j}$, with $0 \leq j \leq 2^i - 1$ satisfying the following three conditions: $g_{i,j} \in l'_{i,j} \pm c'_{i,j}/4$, $c'_{0,0} \cdot c_{0,0} \leq \frac{1}{2}$ and $c'_{i+1,j} \cdot c_{i+1,j} \leq \frac{c'_{i,j/2}}{2}$, that is $l'_{i,j}$ is a piecewise linear approximation of the function $g_{i,j}$ such that the value part of $g_{i,j}$ is approximated within an error $c'_{i,j}/4$, while there is no condition on the derivative part of $l'_{i,j}$.

The function f can then be expressed as

$$l_{0,0} \oplus_{c_{0,0}} (B_{c'_{0,0}} l'_{0,0}(\mathsf{max}\ (l_{1,0} \oplus_{c_{1,0}} (B_{c'_{1,0}} l'_{1,0}(\mathsf{max}\ \ldots))), \\ (l_{1,1} \oplus_{c_{1,1}} (B_{c'_{1,1}} l'_{1,1}(\mathsf{max}\ \ldots))))).$$

The conditions on the constants $c'_{i,j}$ are such that the expansion of the above formula until the level i describes the value part of f with precision 2^{-i}. The

conditions on $l'_{i,j}$ are such that a further application of the B operator determines the value of the function with an error strictly smaller than the application above it.

Given a maximal computable element f in the function domain $\mathcal{D}_{\iota\to\iota}$, the value part f_v is a total functions. Moreover, by the computability of f, it is possible to effectively generate, with an arbitrary precision, the graphs of the functions f_v and f_d. Therefore it is possible to effectively generate the families of rationals $c_{i,j}$, $c'_{i,j}$ and the piecewise linear functions $l_{i,j}$, $l'_{i,j}$ of the construction above. To ensure the convergence of the derivative part, we also require that given a recursive enumeration of the finite elements below f, the rational number $c_{i,j}$ is chosen as the largest number in the form $\frac{k}{2^i}$ that can be generated after examining the first 2^i elements in the enumeration of f. Since the construction is effective, by Turing completeness of PCF, there exist two PCDF terms $\mathsf{l}, \mathsf{l}' : \nu \to \nu \to \iota \to \iota$ generating the above families $l_{i,j}$, $l'_{i,j}$, and four PCDF terms $\mathsf{nc}, \mathsf{dc}, \mathsf{nc}', \mathsf{dc}' : \nu \to \nu \to \nu$ generating numerators and denominators of the rational numbers $c_{i,j}$, $c'_{i,j}$.

Let $\mathsf{f} : \nu \to \nu \to \iota \to \iota$ be the expression

$$\begin{aligned}\mathsf{f} = Y\ \lambda F : \nu \to \nu \to \iota \to \iota.\, \lambda i, j : \iota \\ \mathsf{L}(\mathsf{l}\, i\, j)(\mathsf{nc}\, i\, j)(\mathsf{dc}\, i\, j)(\mathsf{B}(\mathsf{l}'\, i\, j))(\mathsf{nc}'\, i\, j)(\mathsf{dc}'\, i\, j)(\mathsf{max}\ (F(i+1)(2j)) \\ (F(i+1)(2j+1)))\end{aligned}$$

The expression $\mathsf{f}\,0\,0$ defines the function f; in the sense that for any real number $x \in I$, we have $f_v(x) = (\mathcal{E}[\![\mathsf{f}_{0,0}]\!]_\rho)_v(x)$ and $f_d(x) = (\mathcal{E}[\![\mathsf{f}_{0,0}]\!]_\rho)_d(x)$.

Note that above definability result outlines a program expression that computes a function similar to the tradition of numerical analysis: the function f is expressed as the limit of a sequence of piecewise linear functions and the program that computes the value of the function at a given point actually also computes the values of the derivative at that point. Note moreover that the definability constructions do not use the parallel if operator pif.