Available online at [www.sciencedirect.com](http://www.sciencedirect.com/)

[Electronic Notes in Theoretical Computer Science 352 (2020) 211–232](https://doi.org/10.1016/j.entcs.2020.09.011)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

A Complete Equational Axiomatisation of Partial Differentiation

Gordon D. Plotkin [1](#_bookmark0)

*Google Research Mountain View, United States*

**Abstract**

We formalise the well-known rules of partial differentiation in a version of equational logic with function variables and binding constructs. We prove the resulting theory is complete with respect to polynomial in- terpretations. The proof makes use of Severi’s interpolation theorem that all multivariate Hermite problems are solvable. We also present a number of related results, such as decidability and equational completeness.

*Keywords:* Partial differentiation, second-order equational logic, completeness, Hermite interpolation

# Introduction

There has been recent increasing interest in categorical axiomatisations of differen- tial structure. For example, for forward differentiation, see [[2](#_bookmark20)]; for reverse differen- tiation see [[4](#_bookmark22)]; and for tangent structures, see [[3](#_bookmark23)]. A natural question is whether the axioms are well chosen. The authors generally show they hold in natural structures. For example, in the case of cartesian differential categories they hold for the cate- gory of finite powers of the reals and smooth functions. So the axioms are, in that sense, correct. But one can additionally ask, if, or in what sense, they are complete, that is whether there are missing axioms.

Here we interest ourselves in an allied basic logical question which we hope will help with the categorical one: are the standard rules for manipulating partial derivatives complete? The rules are indeed well known: derivatives of products and sums are given by simple formulas involving their immediate subexpressions and their partial derivatives; derivatives of real constants and variables are 0 or 1; the chain rule takes care of function applications; and partial derivatives with respect to different variables commute. They are surely complete.

1 Email: [plotkin@google.com](mailto:plotkin@google.com)

<https://doi.org/10.1016/j.entcs.2020.09.011>

1571-0661/© 2020 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

To prove this, we proceed by first formalising the standard rules of partial dif- ferentiation using a suitable kind of equational logic. Our axiom system consists of the ring axioms, addition and multiplication tables for the reals, and four axioms for partial differentiation. As partial differentiation involves variable binding, we are outside the scope of standard equational logic. So we instead use an expanded version which allows binding constructs and makes use of function variables. An equational logic of this kind, called *second-order equational logic*, has been pre- sented by Fiore, Hur, and Mahmoud [[6,](#_bookmark25)[7](#_bookmark26)]. We employ a minor, although entirely equivalent, variant of their logic.

The resulting theory of partial differentiation is naturally interpreted using smooth functions, but, for completeness, it turns out that it suffices to use only polynomial functions. Indeed, the theory is complete even if the interpretation of function variables is restricted to natural number polynomial functions, and the in- terpretation of variables to natural numbers. This remains the case if the constants are restricted to the rationals or to 0 and 1 (equivalently the integers).

To establish completeness we employ an interpolation theorem, a well-known theorem of Severi [[12](#_bookmark31)] on the solvability of Hermite interpolation problems. Hermite interpolation generalises Lagrange interpolation. In Lagrange interpolation, one seeks a real polynomial taking prescribed values at prescribed points, and this has an evident generalization to multivariate Lagrange interpolation. In multivariate Hermite interpolation, one additionally prescribes values of (possibly higher-order) partial derivatives.

Theorem [5.4](#_bookmark13) establishes completeness relative to natural number polynomial in- terpretations of function variables and natural number interpretations of variables. (The polynomials are adaptations of real polynomials solving suitable Hermite in- terpolation problems.) We also give a number of other results. Theorem [5.5](#_bookmark14) char- acterises the theorems of our equational theory in terms of an equivalence relation between canonical forms. Theorem [5.8](#_bookmark17) shows that the equivalence relation holds if, and only if, it can be established using just one axiom for partial differentiation, that the order of partial differentiation by two variables does not matter (with a slightly more refined notion of canonical form this axiom can be eliminated, when no axioms for partial differentiation are needed). Theorem [5.9](#_bookmark18) shows that the the- ory is not only complete with respect to standard interpretations over the reals, it is also equationally complete (also known as Hilbert-Post complete), that is, it has no equationally consistent proper extensions, see [[14](#_bookmark33)]. This remains the case if the constants are restricted to the rationals but not if they are restricted to the integers. Theorem [5.10](#_bookmark19) shows that the subtheory in which the constants are restricted to the rationals is decidable, and that, in case an equation does not hold, a counterexam- ple to it can be found. An important remaining question is the complexity of the decision problem.

# Axiomatisation of partial differentiation

We assume disjoint sets of variables, ranged over by *x, y, z, . . .*, and of function variables, ranged over by *f, g, h, . . .*; each function variable has a given arity *n ≥* 0, written *f* : *n*. The set of variables is assumed denumerable, as is the set of function variables of each arity. Expressions have the following forms:

*e* ::= *r* (*r ∈* R) *| x | e*0 + *e*1 *| e*0*e*1 *| f* (*e*0*,..., en—*1) (*f* : *n*) *|* PDiff(*x.e*0*, e*1) The expression PDiff(*x.e*0*, e*1) is read as the partial derivative of *e*0 with respect to

*x*, evaluated at *e*1. In this expression, the variable *x* has binding power over *e*0, but

not over *e* . Below, we use the more familiar, and more natural, expression *∂e*0

1

*∂x*

*x*=*e*1

*∂xn−*1*...∂x*0

(while more natural, this expression obscures the role of the variable *x*). We write

*e* for

*∂ ∂e*

*∂x*

*∂x x*=*x*

for *∂ ...*  *∂ e*.

*∂xn−*1

, and note that the last occurrence of *x* is free; we write *∂ne*

*∂x*0

Free variables and *α*-equivalence, are defined as usual, and, as is also usual, we

identify *α*-equivalent expressions. We write FV(*e*) for the set of free variables of an expression *e* and FnV(*e*) for the set of its function variables. Every expression *e* has a size *|e|*, defined in an evident way.

The simultaneous substitution

*e*[*ej /x*0*,..., ej*

*/xn—*1]

0 *n—*1

of expressions for variables is defined as may be expected, with the clause for PDiff(*x.e*0*, e*1), with its bound variable, being:

PDiff(*x.e*0*, e*1)[*ej /x*0*,..., ej*

*/xn—*1]

0 *n—*1

= PDiff(*x.e*0[*ej /x*0*,..., ej /xn—*1]*, e*1[*ej /x*0*,..., ej*

*/xn—*1])

0 *n—*1 0 *n—*1

where *x ∈/* FV(*ej* )*∪.. .∪*FV(*ej*

). There is also a notion of substitution of *abstracts*

0 *n—*1

(*x*0*,..., xn—*1)*.ej* for function variables. For *f* : *n*, we define

*e*[(*x*0*,..., xn—*1)*.ej/f* ] by structural induction on *e* as follows:

*r*[(*x*0*,..., xn—*1)*.ej/f* ] = *r*

*x*[(*x*0*,..., xn—*1)*.ej/f* ] = *x*

(*e*0 + *e*1)[(*x*0*,..., xn—*1)*.ej/f* ] = *e*0[(*x*0*,..., xn—*1)*.ej/f* ]+ *e*1[(*x*0*,..., xn—*1)*.ej/f* ]

(*e*0*e*1)[(*x*0*,..., xn—*1)*.ej/f* ] = *e*0[(*x*0*,..., xn—*1)*.ej/f* ]*e*1[(*x*0*,..., xn—*1)*.ej/f* ]

*g*(*e*0*,..., en—*1)[(*x*0*,..., xn—*1)*.ej/f* ]

⎧⎪ *ej*[*e*0[(*x*0*,..., xn—*1)*.ej/f* ]*/x*0*,..., en—*1[(*x*0*,..., xn—*1)*.ej/f* ]*/xn—*1]

= ⎪⎪⎨

⎪

(*g* = *f* ) *g*(*e*0[(*x*0*,..., xn—*1)*.ej/f* ]*,..., en—*1[(*x*0*,..., xn—*1)*.ej/f* ])

(*g /*= *f* )

PDiff(*x. e*0*, e*1)[(*x*0*,..., xn—*1)*.e /f* ]

= PDiff(*x. e*0[(*x*0*,..., xn—*1)*.ej/f* ]*, e*1[(*x*0*,..., xn—*1)*.ej/f* ])

⎪⎩ *j*

(*x ∈/* FV(*ej*)*\{x*0*,..., xn—*1*}*)

There is a more general, similarly defined, simultaneous substitution of several abstracts:

*e*[(*x*11*,..., x*1*n* )*.ej /f*1*,...,* (*xk*1*,..., xkn* )*.ej /fk*]

1 1 *k k*

where *fi* : *ni*, for *i* = 1*, k*.

The axioms of the theory of partial differentiation consist of those for commu- tative rings, the addition and multiplication tables for the real constants, and four axioms for partial differentiation. The four axioms are the usual rules for the dif- ferentiation of addition and multiplication, the chain rule for binary functions, and the commutativity of partial differentiation with respect to different variables.

*∂x* + *y*

=1

*∂x*

*∂yx*

= *y*

*∂x*

*∂ f* (*g*0(*x*)*, g*1(*x*))

*∂f*(*x*0*, g*1(*x*))

=

*∂g*0(*x*) *∂f*(*g*0(*x*)*, x*1)

+

*∂g*1(*x*)

*∂x ∂x*0

*x*0=*g*0(*x*) *∂x*

*∂x*1

*x*1=*g*1(*x*) *∂x*

*∂ ∂*

*f* (*x, y*) =

*∂ ∂ f* (*x, y*)

*∂y ∂x ∂x ∂y*

The theorems of the theory of partial differentiation are obtained by closing the axioms under the evident rules for equality, congruence, and both kinds of substitution, following the usual pattern for equational systems. In particular, the congruence rule for partial differentiation is as follows:

*e*0 = *e*1 *ej* = *ej*

0

1

PDiff(*x. e*0*, ej* )= PDiff(*x. e*1*, ej* )

0 1

If an equation *e*0 = *e*1 holds in this theory we write

*▶ e*0 = *e*1

We next establish a number of expected consequences of the axioms, including

a general form of the chain rule. We employ an evident abbreviation Σ*n ei* for

*i*=*m*

finite sum expressions.

**Lemma 2.1** *The following equations are provable in the theory of partial differen- tiation:*

(i)

(ii)

*∂x*

*∂x*

*∂ f* (*x*)+ *g*(*x*)

=1

*∂f* (*x*)

*∂g*(*x*)

(iii)

*∂x*

*∂ f* (*x*)*g*(*x*)

= +

*∂x*

*∂f* (*x*)

*∂x*

*∂g*(*x*)

(iv)

*∂x*

*∂ f* (*g*0(*x*)*,..., gn—*1(*x*))

*∂x*

= *g*(*x*)

*∂x*

+ *f* (*x*)

*∂x*

*n—*1

Σ

*∂f*(*g*0(*x*)*,..., gi—*1(*x*)*, xi, gi*+1(*x*)*,..., gn—*1(*x*))

*∂gi*(*x*)

(v)

**Proof.**

=

*i*=0

*∂xi*

*∂e* =0 (*x ∈/* FV(*e*))

*∂x*

*xi*=*gi*(*x*) *∂x*

* 1. This follows from the axiom for addition, substituting 0 for *y*.
  2. This follows from the axiom for addition and the binary chain rule.
  3. This follows from the axiom for multiplication and the binary chain rule.
  4. First, note that, substituting 0 for *y* in the axiom for multiplication, we obtain

*∂*0 = 0. Next, for the case *n* = 0 of the chain rule, substitute (*x*0*, x*1)*. h*() for *f* and (*z*)*.* 0 for both *g*0 and *g*1 in the binary chain rule to get the following provable equations:

*∂x*

*∂h*()

*∂x*

= *∂h*()

0

*∂x*

*∂*0 *x*0=0 *∂x*

+ *∂h*()

1

*∂x*

*∂*0 *x*1=0 *∂x*

= *∂h*()

*∂x*

0

= 0

*x*0=0

*∂h*()

*∂x*1

0 +

0

*x*1=0

Next, for the unary chain rule, substitute (*x*0*, x*1)*. h*(*x*0) for *f* and (*z*)*.* 0 for

*g*1 in the binary chain rule to get the following provable equations:

*∂h*(*g*0(*x*0))

*∂x*

= *∂h*(*x*0)

0

*∂x*

= *∂h*(*x*0)

*∂x*

0

*x*0=*g*0(*x*) *x*0=*g*0(*x*)

*∂g*0(*x*)

*∂x*

*∂g*0(*x*)

*∂x*

+ *∂h*(*x*0)

1

*∂x*

*∂*0 *x*1=0 *∂x*

Finally, for *n ≥* 2, we proceed by induction. The axiom provides the base

case, so assuming the induction hypothesis for *n*, we prove it for *n* + 1. Sub-

stituting (*z*)*.z* for *g*0 and *g*1 in the binary chain rule, and using *∂x* = 1, we

*∂x*

obtain the following equation:

+

1

*∂ f* (*x, x*)

1

=

*∂x*

*∂f* (*xj , x*)

0

*∂f* (*x, xj* )

0

*∂xj*

1

Then, substituting (*x, y*)*. h*(*g*0(*x*)*, g*1(*y*)*,..., gn*(*y*)) for *f* , we obtain:

0

*∂xj*

*x′* =*x*

*x′* =*x*

0

*∂ h*(*g*0(*x*)*, g*1(*x*)*,..., gn*(*x*))

0

=

*∂x*

*∂h*(*g*0(*xj* )*, g*1(*x*)*,..., gn*(*x*))

*∂xj*

0

+

*x′* =*x*

For the first of these two summands we have:

1

*∂xj*

1

*∂h*(*g*0(*x*)*, g*1(*xj* )*,..., gn*(*xj* ))

1

*x′* =*x*

1

*∂h*(*g*0(*x′* )*,g*1(*x*)*,...,gn*(*x*))

= *∂h*(*g*0(*x′* )*,g*1(*x*)*,...,gn*(*x*)) [*x/xj* ]

0

*∂x′*

0

0

*x′* =*x*

*∂x′*

= *∂h*(*x*0*,g*1(*x*)*,...,gn*(*x*))

0

*∂g*0(*x′* ) [*x/xj* ]

0

0

0

*∂x*0

*x*0=*g*0(*x′* )

*∂x′* 0

0

0

= *∂h*(*x*0*,g*1(*x*)*,...,gn*(*x*))

*∂x*

0

*x*0=*g*0(*x*)

*∂g*0(*x*)

*∂x*

with the second equality being an application of the unary chain rule and the other two equalities being syntactic identities (the second up to *α*-equivalence).

For the second of these, making use of the induction hypothesis (the *n*-ary chain rule) we have:

*∂h*(*g*0(*x*)*,g*1(*x′* )*,...,gn*(*x′* ))

1

*∂x′*

1

*x′* =*x*

= *∂h*(*g*0(*x*)*,g*1(*x′* )*,...,gn*(*x′* )) [*x/xj* ]

1

1

= Σ*n*

1

1

*∂h*(*g*0(*x*)*,g*1(*x′* )*,...,gi−*1(*x′* )*,xi,gi*+1(*x′* )*,...,gn*(*x′* ))

1

*i*=1

1

1

*∂xi*

1

1

1

*∂x′*

1

*∂gi*(*x′* ) [*x/xj* ]

= Σ*n*

*i*=1

*∂h*(*g*0(*x*)*,g*1(*x*)*,...,gi−*1(*x*)*,xi,gi*+1(*x*)*,...,gn*(*x*))

*′*

*∂x′*

1

*∂xi*

*xi*=*gi*(*x*1) 1

*∂gi*(*x*)

*xi*=*gi*(*x*)

*∂x*

Combining the results, we reach the desired conclusion.

* 1. For *n* = 0, the chain rule is *∂f*() = 0, where *f* : 0. Substituting ( )*.e* for *f* we

*∂x*

obtain *∂e* =0 as *x ∈/* FV(*e*).

*∂x*

*2*

It will prove useful to know the interaction between differentiation and substi- tution:

**Lemma 2.2** *The following are equivalent:*

1. *The general chain rule.*
2. *The following substitution principle, that for all expressions e and e*0*,..., en—*1

*we have:*

*▶ ∂e*[*e*0*/x*0*,..., en—*1*/xn—*1]

*∂x*

*n—*1

= *∂e* [*e /x ,...,e /x*

Σ

*∂xi*

0

0

*n—*1

*n—*1

] *∂ei*

*i*=0

*where x*0*,..., xn—*1 *are distinct variables such that x ∈/* FV(*e*)*\{x*0*,..., xn—*1*}.*

*∂x*

**Proof.** In one direction assume the general chain rule. Choose *f* : *n* not in any of the FnV(*ei*). Using the general chain rule we have:

*∂f*(*e*0*,..., en—*1)

*▶*

*n—*1

*∂f*(*e*0*,..., ei—*1*, xi, ei*+1*,..., en—*1)

Σ

=

*∂x*

*∂ei*

*i*=0

*∂xi*

*xi*=*ei*

*∂x*

Then, substituting (*x*0*,..., xn—*1)*.e* for *f* on the left we have the following identities (recall that we identify *α*-equivalent expressions):

*∂f*(*e*0*,...,en−*1) [(*x ,...,x*

)*. e/f* ] = *∂f*(*e*0*,...,en−*1)[(*x*0*,...,xn−*1)*. e/f* ] (as *x ∈/* FV(*e*)*\{x ,...,x }*)

*∂x* 0

*n—*1

*∂x*

= *∂e*[*e*0*/x*0*,...,en−*1*/xn−*1]

0

(as *f ∈/* FnV(*e ,...,e*

*n−*1

))

*∂x* 0

*n−*1

Next, substituting on the right, we have the identities:

Σ*n—*1 *∂f* (*e*0*,...,ei−*1*,xi,ei*+1*,...,en−*1)

*i*=0

*∂xi*

*∂ei* [(*x*0*,..., xn—*1)*. e/f* ]

= Σ*n—*1 *∂f* (*e*0*,...,ei−*1*,xi,ei*+1*,...,en−*1)

*xi*=*ei*

*∂x*

[(*x*0*,..., xn—*1)*. e/f* ] *∂ei*

*i*=0

*∂xi*

*xi*=*ei ∂x*

(as *f ∈/* FnV(*ei*), for *i* = 0*,...,n —* 1)

= Σ*n—*1 *∂f* (*e*0*,...,ei−*1*,xi,ei*+1*,...,en−*1)[(*x*0*,...,xn−*1)*. e/f* ]

*i*=0

*∂xi*

*∂ei*

*xi*=*ei*

*∂x*

(as *xi ∈/* FV(*e*)*\{x*0*,..., xn−*1*}*)

= Σ*n—*1 *∂e*[*e*0*/x*0*,...,ei−*1*/xi−*1*,xi/xi,ei*+1*/xi*+1*,...,en−*1*/xn−*1]

*i*=0

*∂xi*

*xi*=*ei*

*∂ei*

*∂x*

(as *f ∈/* FnV(*e*0*,..., ei−*1*, ei*+1*,..., en−*1))

= Σ*n—*1 *∂e*[*e*0*/x*0*,...,ei−*1*/xi−*1*,ei*+1*/xi*+1*,...,en−*1*/xn−*1]

*i*=0

*∂xi*

*∂ei*

*∂x*

= Σ*n—*1 *∂e* [*e*0*/x*0*,..., en—*1*/xn—*1] *∂ei*

*xi*=*ei*

*i*=0

*∂xi*

*∂x*

and the conclusion follows.

In the other direction, assuming the substitution principle, take *e* to be *f* (*x*0*,..., xn—*1) (when FV(*e*)*\{x*0*,..., xn—*1*}* = *∅*) and *ei* to be *gi*(*x*) for *i* = 0*,n−* 1, to obtain:

*n—*1

*∂x*

*▶ ∂f*(*g*0(*x*)*,..., gn—*1(*x*))

*∂x*

*n—*1

= *∂f*(*x*0*,..., xn—*1) [*g* (*x*)*/x ,...,g*

Σ

*∂xi*

0

0

*n—*1

(*x*)*/x*

] *∂gi*(*x*)

*i*=0

As

*∂f*(*x*0*,..., xn—*1) [*g* (*x*)*/x ,...,g*

(*x*)*/x* ]

and

*∂xi*

0 0 *n—*1

*n—*1

*∂xi*

*xi*=*gi*(*x*)

*∂f*(*g*0(*x*)*,..., gi—*1(*x*)*, xi, gi*+1(*x*)*,..., gn—*1(*x*))

are *α*-equivalent, we are done. *2*

A *(real) polynomial* is (as usual) an expression containing no function variables or partial differentiations; it is a *natural number* polynomial if all its constants are natural numbers. We write *P* (*x*0*,..., xn—*1) to indicate that *P* is a real polynomial whose (necessarily free) variables are included in *{x*0*,..., xn—*1*}*; for expressions *e*0*,..., en—*1 we may then write *P* (*e*0*,..., en—*1) for *P* [*e*0*/x*0*,..., en—*1*/xn—*1]. We write *P ∼ Pj* to mean that the polynomials *P* and *Pj* are equal modulo the ax- ioms for commutative rings and the addition and multiplication tables for the real constants.

We remark that our equational system is an instance of a second-order equa- tional logic for binding. This logic has *operator constants* op of arities of the form (*b*0*,..., bn—*1; *m*) (*bi,m ≥* 0). These are used to form a compound expression from *n* abstracts, of respective arities *b*0*,..., bn—*1, and *m* expressions. With this notation,

+ has arity (; 2) and PDiff has arity (1; 1). This system is single-sorted; there is a natural generalisation to a multisorted version.

In [[6](#_bookmark25)] Fiore and Hur gave a multisorted second-order equational logic, and in [[7]](#_bookmark26) Fiore and Mahmoud considered the single-sorted case. The differences between their systems and ours are inessential: what we term ‘function variables’ they term ‘metavariables’; whereas our function variables come with preassigned arities, theirs do not, and they rather utilise suitable environments; their arities are of the simpler form (*b*0*,..., bn—*1) but are no less general, as the extra arguments become 0-ary abstracts; and whereas we have both a substitution operation for function variables and a context rule, they employ a single equivalent rule. A single substitution version of their equivalent rule is:

*e*0 = *e*1

*e*0[(*x*0*,..., xn—*1)*.ej/f* ]= *e*1[(*x*0*,..., xn—*1)*.ej/f* ]

They employ a simultaneous substitution version of this rule, which is, in any case, equivalent to the single substitution rule.

Returning to our single-sorted version of second-order equational logic, we say that an equational theory is *equationally inconsistent* if *x* = *y* is a theorem (with *x*, *y* different), and we say that an an equational theory is *equationally complete* if any extension of it by a single non-theorem is equationally inconsistent. Note that this is a syntactic criterion, independent of any particular model.

Polynomials provide an example of equational completeness, though one with- out a binding operator. The theory is that of commutative rings with constants for all reals and addition and multiplication tables (i.e., our theory less partial differ- entiation). The idea of the proof is that distinct polynomials have distinct values

for suitable choices of values for their variables. This can be formalised within the logic, and so, given an equation between distinct polynomials, two distinct reals can be proved equal, and that, in turn, enables one to prove *x* = *y*.

The *λβη*-calculus provides a partial example with a binding operator. It can be formalised as a second-order theory with a binary application operator ap:(; 2) and a unary lambda abstraction operator *λ* :(1; 0) and the two equations

ap(*λ*((*x*)*.f* (*x*))*, y*)= *f* (*y*) *λ*((*y*)*.* ap(*x, y*)) = *x*

While not equationally complete, it is partially so in the sense that, by B¨ohm’s theorem [[1](#_bookmark21)], no two distinct *βη*-normal forms can be consistently equated. The theory of the Beta-Bernoulli process [[13](#_bookmark32)] also has binding operators; it is formulated using an equational logic that is a bit different than ours in order to accommodate algebraic effects. The theory is almost equationally complete, in the sense that it has just one consistent extension. In this extension all quantitative information about probabilities is lost, and so can be ruled out for the purposes at hand.

# Semantics

We give a semantics of our partial differentiation expressions using smooth func- tions *h* : R*n →* R. An *environment ρ* is a function from variables to reals; a *function environment ϕ* is a function from function variables to smooth functions that sends function variables of arity *n* to smooth functions on R*n*. We write *ρ*[*r*0*/x*0*,..., rn—*1*/xn—*1] for the environment with value *ri* on *xi* (for *i* = 0*, n−*1) and value *ρ*(*x*) on any other variable *x*; function environments *ϕ*[*h*0*/f*0*,..., hn—*1*/fn—*1], where *hi* is *m*-ary if *fi* : *m*, are defined similarly. We write 0 for both the constantly 0 environment and the function environment yielding constantly 0 functions.

We define the denotation *S*[[*e*]]*ϕρ* of an expression relative to a function environ- ment and an environment by the following clauses:

|  |  |  |
| --- | --- | --- |
| *S*[[*r*]]*ϕρ* | = | *r* |
| *S*[[*x*]]*ϕρ* | = | *ρ*(*x*) |
| *S*[[*e* + *ej*]]*ϕρ* | = | *S*[[*e*]]*ϕρ* + *S*[[*ej*]]*ϕρ* |
| *S*[[*eej*]]*ϕρ* | = | *S*[[*e*]]*ϕρ × S*[[*ej*]]*ϕρ* |
| *S*[[*f* (*e*0*,..., en—*1)]]*ϕρ* | = | *ϕ*(*f* )(*S*[[*e*0]]*ϕρ,..., S*[[*en—*1]]*ϕρ*) |
| *S*[[PDiff(*x. e, ej*)]]*ϕρ* | = | *D*(*r ∈* R *'→ S*[[*e*]]*ϕρ*[*r/x*])(*S*[[*ej*]]*ϕρ*) |

Prima facie this definition may not be proper as the last clause depends on the function *r ∈* R *'→ S*[[*e*]]*ϕρ*[*r/x*] being differentiable. However one proves by struc- tural induction that the definition is proper and, in addition, that for any variables *x*0*,..., xn—*1 the function sending *r*0*,..., rn—*1 *∈* R to *S*[[*e*]]*ϕρ*[*r*0*/x*0*,..., rn—*1*/xn—*1] is smooth.

The denotation *S*[[*e*]]*ϕρ* of an expression depends only on the values ascribed to its free function variables by *ϕ*, and its free variables by *ρ*. When *e* has no function variables, we just write *S*[[*e*]]*ρ* for its denotation, omitting *ϕ*, and when it is closed (has no free variables of either kind) we just write *S*[[*e* **]**. As usual, we write

*|*= *e*0 = *e*1 to mean that *e*0 and *e*1 have the same denotation, given any function environment and any environment.

We omit the proof of the following standard substitution lemma:

**Lemma 3.1** (i) *Variable substitution and denotation commute, that is, we have:*

*S*[[*e*[*e*0*/x*0*,..., en—*1*/xn—*1]]]*ϕρ* = *S*[[*e*]]*ϕρ*[*S*[[*e*0]]*ϕρ/x*0*,..., S*[[*en—*1]]*ϕρ/xn—*1]

(ii) *Function variable substitution and denotation commute, that is, we have:*

*S*[[*e*[(*x*0*,..., xn—*1)*.ej/f* ]]]*ϕρ* =

*S*[[*e*]]*ϕ*[*u*0*,..., un—*1 *'→ S*[[*ej*]]*ϕρ*[*u*0*/x*0*,..., un—*1*/xn—*1]*/f* ]*ρ*

Using this lemma, it is then straightforward to prove consistency:

**Theorem 3.2 (Consistency)** *For any expressions e*0 *and e*1 *we have:*

*▶ e*0 = *e*1 =*⇒ |*= *e*0 = *e*1

Any polynomial *P* (*x*0*,..., xn—*1) defines an *n*-ary polynomial function on the reals, also written *P* (*x*0*,..., xn—*1). Note that

*P* (*r*0*,..., rn—*1) = *S*[[*P* ]]0[*r*0*/x*0*,..., rn—*1*/xn—*1]

and we recall that *P* (*x*0*,..., xn—*1) *∼ Pj*(*x*0*,..., xn—*1) if, and only if, the two func- tions *P* (*x*0*,..., xn—*1) and *Pj*(*x*0*,..., xn—*1) are equal.

We say that a function environment is a *(natural number) polynomial* envi- ronment if all its values are (natural number) polynomial functions and that an environment is a natural number environment if all its values are natural numbers. We write

*|*=N *e*0 = *e*1

to mean that *e*0 and *e*1 have the same denotation, given any natural number poly- nomial function environment, and any natural number environment.

Our semantics of the theory of partial differentiation employs a concrete notion of function, viz. smooth functions over the reals. For the general theory of second- order logic, a more abstract, and thereby more general, notion of function is needed. This can be formulated variously in terms of abstract clones [[5](#_bookmark24)], Lawvere theories, or, more conceptually, in terms of monoids in a presheaf category of contexts, see [[8,](#_bookmark27)[6,](#_bookmark25)[7](#_bookmark26)].

# Canonical forms

To show completeness we need suitable canonical forms *c*; up to a suitable equiva- lence relation *c ≈ cj*, they provide normal forms for our theory.

Let *x*0*, x*1*,...* be a countably infinite sequence of distinct variables. For any function variable *f* : *n*, argument sequence *m* = *i*1*,..., ik ∈* [*n*]*∗* (where [*n*] =

*{*0*,...,n −* 1*}*), and expressions *e*0*,..., en—*1 we set:

*∂kf* (*x*0*,..., xn—*1)

*fm*(*e*0*,..., en—*1) =

We note that

*∂xi*1

*... ∂xik*

[*e*0*/x*0*,..., en—*1*/xn—*1]

*n—*1 *n—*1

FV(*fm*(*e*0*,..., en—*1)) = FV(*ei*) and FnV(*fm*(*e*0*,..., en—*1)) = *{f}∪* FnV(*ei*)

*i*=0

*i*=0

and that

*fm*(*e*0*,..., en—*1)[*e/x*] = *fm*(*e*0[*e/x*]*,..., en—*1[*e/x*])

We remark that if two such expressions *fm*(*e*0*,..., en—*1) and *fj ′* (*ej ,..., ej ′*

) are

*m* 0 *n —*1

equal then so are: *f* and *fj*; *m* and *mj*; *n* and *nj*; and *ei* and *ej*, for *i* = 1*, n*; this is

*i*

used implicitly below to ensure the uniqueness of expression case analyses. We also note that, as may be expected, for any *ϕ* and *ρ* we have:

*S*[[*fm*(*e*0*,..., en—*1)]]*ϕρ* = *Dm*(*ϕ*(*f* ))(*S*[[*e*0]]*ϕρ,..., S*[[*en—*1]]*ϕρ*)

The chain rule takes the following form for such applications of partial derivatives of functions:

**Lemma 4.1** *We have:*

*▶ ∂fm*(*e*0*,..., en—*1)

*∂x*

*n—*1

= *f*

Σ

*im*

(*e ,...,e*

) *∂ei*

*i*=0

**Proof.** Supposing *m* = *i*1*,..., ik ∈* [*n*]*∗*, then we have

0

*n—*1

*∂x*

*∂fm*(*e*0*,...,en−*1) = *∂* *∂kf* (*x*0*,...,xn−*1) [*e /x ,...,e /x* ]

*∂x*

*∂x*

*∂xi*1 *...∂xik*

0

0

*n—*1

*n—*1

= Σ*n—*1 *∂ ∂kf* (*x*0*,...,xn−*1) [*e /x ,...,e /x*

*i*=0

*∂x*

*∂xi*1 *...∂xik*

0

0

*n—*1

*n—*1

*∂x*

] *∂ei*

(by Lemma [2.2](#_bookmark2))

= Σ*n—*1 *fim*(*e*0*,..., en—*1) *∂ei*

*i*=0

*∂x*

where the second identity is a provable equality, and the other two are syntactic.*2* We define a set of *canonical forms c* and a set of *atomic expressions a* (both sets

of expressions) by simultaneous induction:

1. (a) Any atomic expression is a canonical expression.
   1. Any *r ∈* R is a canonical expression, as are *c* + *cj* and *ccj* if *c* and *cj* are.
2. (a) Any variable *x* is an atomic expression.
   1. For any *f* : *n*, *m ∈* [*n*]*∗*, and canonical forms *c*0*,..., cn—*1, *fm*(*c*0*,..., cn—*1) is an atomic expression.

The set of *immediate* atomic subexpressions of a canonical form *c* is defined by structural recursion:

Im(*a*)= *{a}* Im(*r*)= *∅* Im(*c* + *cj*)= Im(*ccj*)= Im(*c*) *∪* Im(*cj*)

We remark that the canonical forms are closed under substitution: *c*[*cj/x*] is a canonical form if *c* and *cj* are, as is straightforwardly proved by structural induction on *c*.

**Lemma 4.2** *For any canonical forms c and c there is a canonical form cj such that:*

*∂c*

*∂x*

*▶*

*x*=*c*

= *cj*

*and with* FnV(*cj*) *⊆* FnV(*c*) *∪* FnV(*c*) *and* FV(*cj*) *⊆* FV(*c*) *∪* FV(*c*)*.*

**Proof.** The proof is by induction on the size of *c*. The cases where *c* is a constant, variable, sum, or product make use of Lemma [2.1.](#_bookmark1) This leaves the case of an atomic expression *fm*(*c*0*,..., cn—*1), where we can prove:

*∂fm*(*c*0*,...,cn−*1)

*∂x*

*x*=*c*

= *∂fm*(*c*0*,...,cn−*1) [*c/x*]

= Σ*n—*1 *fim*(*c*0*,..., cn—*1) *∂ci* [*c/x*] (by Lemma [4.1](#_bookmark5))

*∂x*

*i*=0

*∂x*

= Σ*n—*1 *fim*(*c*0[*c/x*]*,..., cn—*1[*c/x*]) *∂ci*

*i*=0

*∂x*

*x*=*c*

and recall the canonical forms are closed under substitution and apply the induction hypothesis to the *ci*. *2*

**Lemma 4.3 (Canonicalisation)** *For any expression e there is a canonical form* CF(*e*) *such that ▶ e* = CF(*e*) *and with* FnV(CF(*e*)) *⊆* FnV(*e*) *and* FV(CF(*e*)) *⊆* FV(*e*)*.*

**Proof.** We prove this by structural induction on *e*. If *e* is a variable or a constant it is already a canonical form. The cases where *e* is a sum or a product follow from the fact that the canonical forms are closed under sums and products. The case where *e* has the form *f* (*e*0*,..., en—*1) is immediate from the induction hypothesis.

Finally, the case where *e* has the form *∂e*0

*∂x*

*x*=*e*1

is handled using Lemma [4.2](#_bookmark6), and

the induction hypothesis. *2*

We have the following corollary of the canonicalisation lemma:

**Corollary 4.4** *For any closed expression e, we have:*

*▶ e* = *S*[[*e*]]

**Proof.** As *e* is closed, the same, by Lemma [4.3](#_bookmark7), is true of CF(*e*), which latter must therefore have no immediate atomic subexpressions or free variables, and so is a closed polynomial. There is therefore a real *r* such that *▶* CF(*e*)= *r* and so, by the lemma, *▶ e* = *r*. By consistency we have *r* = *S*[[*e*]]. *2*

We next define the equivalence relations that will allow us to characterise prov- able equality between canonical forms. We also define polynomials that will be used to formulate our Hermite interpolation problems.

For any *m, mj ∈* [*n*]*∗* we write *m ∼ mj* to mean that *m* is a permutation of *mj*.

For any atomic expressions *a, aj* we write *a ≈ aj* to mean that

* *a* and *aj* are identical variables, or else
* for some function variable *f* : *n*, and for some *m ∼ mj*, they have the forms

*fm*(*c*0*,..., cn—*1) and *fm′* (*cj ,..., cj* ) with *▶ ci* = *cj* for *i* = 0*,n −* 1.

We evidently have:

0 *n—*1 *i*

*a ≈ aj* =*⇒▶ a* = *aj* (1)

We now fix an assignment *va* of variables (termed *separation* variables) to atomic expressions such that:

*va* = *va′ ⇐⇒ a ≈ a*

*j*

For any canonical form *c*, we define its *node* polynomial *Pc* by structural induction:

*Pr* = *r* (*r ∈* R) *Pa* = *va Pc*+*c′* = *Pc* + *Pc′ Pcc′* = *PcPc′*

The separation variables occurring in *Pc* are the *va* with *a* an immediate atomic subexpression of *c*.

We then define our equivalence relation between canonical forms by:

*c ≈ c ⇐⇒ Pc ∼ Pc′*

*j*

Note that this relation only depends on the choice of the *va* for the immediate atomic expressions of *c* and *cj*. Note too that the relation extends that between atomic expressions, i.e., *a ≈ aj* holds with *a, aj* taken as atomic expressions iff it does when they are taken as canonical forms.

**Lemma 4.5** (i) *For any canonical expression c with node polynomial*

*Pc*(*va*0 *,..., vam−*1 )*, for atomic expressions a*0*,..., am—*1*, we have:*

*▶ c* = *Pc*(*a*0*,..., am—*1)

(ii) *For any canonical expressions c and cj we have:*

*c ≈ cj* =*⇒▶ c* = *cj*

**Proof.** The first part is established by a straightforward structural induction. In the case where *c* is an atomic expression *a*, we have *va* = *Pc* = *va*0 and so *a ≈ a*0 and so *▶ a* = *a*0, i.e., *▶ c* = *Pc*(*a*0).

For the second part, suppose that we have canonical expressions *c* and *cj* such that *c ≈ cj* (and so *Pc ∼ Pc′* ). Let *va ,..., va* (*j* = 0*, m*) include all the variables of *Pc* and *Pc′* . As *Pc ∼ Pc′* , we have:

0 *m−*1

*▶ Pc*[*a*0*/va*0 *,..., am—*1*/vam−*1 ]= *Pc′* [*a*0*/va*0 *,..., am—*1*/vam−*1 ]

Applying the first part to *c* and *cj*, it follows that *▶ c* = *cj*, as required. *2*

# 5 Completeness

To establish completeness we use environments distinguishing non-equivalent canon- ical forms, i.e., canonical forms *c* and *cj* such that *c /≈ cj*. We first need to be able to distinguish finite sets of non-equivalent polynomials (these will be the node poly- nomials of sub-canonical forms of *c* or *cj*):

**Lemma 5.1** *Let Pi*(*x*0*,..., xm—*1) *be n mutually inequivalent polynomials. Then, for some natural number choices of x*0*,.* *, xm—*1*, they take on different values.*

**Proof.** None of the polynomials *Qij* = *Pi − Pj* are identically 0 (*i /*= *j*), and so neither is *Q* = Π*ijQij*. So *Q* is non-zero for some natural number choices of *x*0*,..., xm—*1. That choice differentiates distinct *Pi*. *2*

We next need to be able to solve multivariate Hermite interpolation problems in order to give prescribed values to atomic expressions. A *multivariate Hermite interpolation problem of dimension d ≥* 0 is given by:

* A finite set of *nodes xi ∈* R*d* (*i* = 1*.k*), and

*∗*

* For each node *xi*, finitely many *conditions Dmij* (*h*)(*xi*)= *rij* (*mij ∈* [*d*] *, rij ∈* R)

on the partial derivatives of a function *h* : R*d →* R. The set of conditions must

be *consistent*, in the sense that each *rij* is determined by the node *xi* and the permutation equivalence class of *mij*.

A *solution* to this problem is a *d*-ary polynomial function satisfying all the condi- tions. According to a theorem of Severi [[12](#_bookmark31)] (and see [[9,](#_bookmark28)[10](#_bookmark29)]) every such problem has a solution by a polynomial of degree *k*(max(*|mij|*)+ 1) *−* 1.

A finite set *C* of canonical forms is *saturated* if the following two conditions hold:

1. For any canonical form *c* and immediate atomic subexpression *a* of *c*

*c ∈ C* =*⇒ a ∈ C*

(ii)

*fm*(*c*0*,..., cn—*1) *∈ C* =*⇒ c*0*,..., cn—*1 *∈ C*

Every finite set of canonical forms can evidently be extended to a finite saturated set of canonical forms.

**Theorem 5.2 (Polynomial Separation)** *Let C be a ﬁnite set of canonical forms. Then there is a polynomial function environment ϕ and an environment ρ that distinguish any two inequivalent elements of C.*

**Proof.** We can assume w.l.o.g. that *C* is saturated. By Lemma [5.1](#_bookmark10) we can find an assignment *ρ* of reals to the separation variables of the node polynomials *Pc*, (*c ∈ C*) which separates them, i.e., makes inequivalent ones take on different values, *rc*. As *Pc ∼ Pc′* iff *c ≈ cj*, we have *c ≈ cj* iff *rc* = *rc′* . In the case that *c* is an atomic expression *a* we have *Pa* = *va* and so *ra* = *ρ*(*va*). We define an environment *ρ* by:

*ρ*(*x*) = ⎧⎨ *rx* (*x ∈ C*)

⎩ 0 (otherwise)

We next set up an *n*-dimensional Hermite interpolation problem for each function variable *f* : *n* appearing in the atomic expressions in *C*. For each atomic expression *fm*(*c*0*,..., cn—*1) *∈ C* we add a node

(*rc*0 *,..., rcn−*1 )

and a condition:

*Dm*(*h*)(*rc*0 *,..., rcn−*1 )= *rfm*(*c*0*,...,cn−*1)

Note that the nodes are determined by the values of the node polynomials.

To show these conditions consistent, let *a* = *fm*(*c*0*,..., cn—*1) and *aj* =

*fm′* (*cj ,..., cj* ) be two atomic expressions in *C* such that *m ∼ mj* and *rc* = *rc′* for

0 *n—*1 *i* *i*

*i* = 0*,n−* 1. Then, by the above, we have *ci ≈ cj*, and so, by part (2) of Lemma [4.5](#_bookmark9),

*i*

*▶ ci* = *cj*, for *i* = 0*,n −* 1. So *a ≈ aj* as atomic expressions and so too, therefore, as canonical forms. It follows that *ra* = *ra′* , as required for consistency.

*i*

We can therefore obtain a polynomial function environment by taking *ϕ*(*f* ) to be an *m*-ary interpolating polynomial for the corresponding Hermite interpolation problem, if *f* appears in some *c ∈ C*, and constantly 0, otherwise.

We claim that:

*S*[[*c*]]*ϕρ* = *rc*

for every *c ∈ C*. We establish the claim by structural induction on the expressions in *C*.

First, consider a variable *x ∈ C*. We have *S*[[*x*]]*ϕρ* = *ρ*(*x*)= *rx*, as required.

Next, consider an atomic expression *fm*(*c*0*,..., cn—*1) *∈ C*. By saturation we have *c*0*,..., cn—*1 *∈ C*. So, by the induction hypothesis, we have *S*[[*ci*]]*ϕρ* = *rci* for *i* = 0*,n −* 1. We then have:

*S*[[*fm*(*c*0*,..., cn—*1)]]*ϕρ* = *Dm*(*ϕ*(*f* ))(*S*[[*c*0]]*ϕρ,..., S*[[*cn—*1]]*ϕρ*)

= *Dm*(*ϕ*(*f* ))(*rc*0 *,..., rcn−*1 )

= *rfm*(*c*0*,...,cn−*1)

with the last line holding as *ϕ*(*f* ) solves the Hermite interpolation problem for *f*

set up above.

Lastly, consider a canonical form *c ∈ C* which is not an atomic expression.

Let *va′ ,..., va′* be the variables of *Pc*. For every *aj* there is an immediate atomic

0 *n−*1 *i*

subexpression *ai* of *c* such that *ai ≈ aj* (and so *va′* = *va* ). By part (1) of Lemma [4.5](#_bookmark9)

we then have

*i i* *i*

*▶ c* = *Pc*[*a*0*/va*0 *,..., an—*1*/van−*1 ]

As *C* is saturated, it contains the *ai*. Since *c* is not itself an atomic expression, we can apply the induction hypothesis to the *ai* and so we have *£*[[*ai*]]*ϕρ* = *rai* .

We may then calculate that:

*£*[[*c*]]*ϕρ* = *£*[[*Pc*[*a*0*/va*0 *,..., an—*1*/van−*1 ]]]*ϕρ*

= *£*[[*Pc*(*va*0 *,..., van−*1 )]]*ϕρ*[*£*[[*a*0]]*ϕρ/va*0 *,..., £*[[*an—*1]]*ϕρ/van−*1 ]

= *£*[[*Pc*(*va*0 *,..., van−*1 )]]*ϕρ*[*ra*0 */va*0 *,..., ran−*1 */van−*1 ]

= *£*[[*Pc*(*va*0 *,..., van−*1 )]]*ϕρ*[*ρ*(*va*0 )*/va*0 *,..., ρ*(*van−*1 )*/van−*1 ]

= *rc*

with the last line holding by the definition of *r**c*. This concludes the inductive proof. Finally, taking *c /≈ cj ∈ C*, we have *rc /*= *rc′* , and so, by the above, *£*[[*c*]]*ϕρ /*=

*£*[[*cj*]]*ϕρ*, and so, as required, *ϕ* and *ρ* distinguish any two inequivalent elements of

*C*, concluding the proof. *2*

We next strengthen the polynomial separation theorem to natural number sep- aration:

**Theorem 5.3 (Natural Number Separation)** *Let C be a ﬁnite set of canonical forms. Then there is a natural number polynomial function environment ϕ*N *and a natural number environment ρ*N *that distinguish any two inequivalent elements of C.*

**Proof.** By polynomial separation (Theorem [5.2](#_bookmark11)), there is a polynomial function environment *ϕ* and an environment *ρ* such that distinguish any two inequivalent elements of *C*.

Let *fi* : *mi* (for *i* = 0*,n —* 1) be the function variables occurring in the *c ∈ C*, and, for *i* = 0*,n —* 1, let *Pi*(*x*0*,..., xmi—*1) be polynomials defining *ϕ*(*fi*). Let *r*0*,..., rq—*1 be the constants occurring in these *Pi* and, for *i* = 0*,n —* 1, let *Qi*(*x*0*,..., xmi—*1*, y*0*,.* *, yq—*1) be natural number polynomials such that

*Qi*[*r*0*/y*0*,.* *, rq—*1*/yq—*1]= *Pi*

and where no *yk* occurs freely in any *c ∈ C*. (The *Qi* can be obtained from the *Pi*

by replacing *rk* by *yk*, for *k* = 0*,q —* 1.)

For *c ∈ C*, set *dc* = CF(*c*[*...,* (*x*0*,..., xmi* )*. Qi/fi,.. .*]). As *fi* is a list of function variables including all those of *c*, and as FnV(*c*) *⊆* FnV(*e*) by Lemma [4.3](#_bookmark7), we see that *c*[*...,* (*x*0*,..., xmi* )*. Qi/fi,.. .*] has no function variables and so, again by Lemma [4.3,](#_bookmark7) neither does its canonical form *dc*. As any canonical form with no function variables is a polynomial, it follows that *dc* is a polynomial.

Setting *ρ* = *ρ*[*..., rk/yk,.. .*], we calculate:

*£*[[*c*]]*ϕρ* = *£*[[*c*]]*ϕ*[*...,* (*u*0*,..., umi ∈* R *'→ £*[[*Pi*]]*ρ*[*..., uj/xj,.. .*])*/fi,.. .*]*ρ*

= *£*[[*c*]]*ϕ*[*...,* (*u*0*,..., umi ∈* R *'→ £*[[*Qi*[*..., rk/yk,.. .*]]]*ρ*[*..., uj/xj,.. .*])*/fi,.. .*]*ρ*

= *£*[[*c*]]*ϕ*[*...,* (*u*0*,..., umi ∈* R *'→ £*[[*Qi*]]*ρ*[*..., uj/xj,..., rk/yk,.. .*])*/fi,.. .*]*ρ*

= *£*[[*c*]]*ϕ*[*...,* (*u*0*,..., umi ∈* R *'→ £*[[*Qi*]]*ρ*[*..., uj/xj,.. .*])*/fi,.. .*]*ρ*

= *£*[[*c*[*...,* (*x*0*,..., xmi* )*. Qi/fi,.. .*]]]*ϕρ*

= *£*[[*dc*]]*ρ*

where we have used Lemma [3.1](#_bookmark3), canonicalisation, and consistency (Theorem [3.2](#_bookmark4)). As *ϕ*, *ρ* separate inequivalent *c ∈ C* we therefore see that the corresponding *dc* are inequivalent as functions and so as polynomials. By Lemma [5.1](#_bookmark10) there is a natural number environment *ρ*N separating any two inequivalent *dc*, and so any *dc* and *dj*

*c*

with *c* and *cj* inequivalent.

Next, define a natural number polynomial function environment *ϕ*N by setting:

*ϕ* (*f* ) = ⎧⎨ *u*0*,..., uni ∈* R *'→ £*[[*Qi*]]*ρ*N[*u*0*/x*0*,..., uni /xni* ] (*f* = *fi*)

N

⎩

For any *c ∈ C* we have:

0 (otherwise)

*£*[[*c*]]*ϕ*N*ρ*N = *£*[[*c*]]0[*...,* (*u*0*,..., uni ∈* R *'→ £*[[*Qi*]]*ρ*N[*..., uj/xj,.. .*])*/fi,.. .*]*ρ*N

= *£*[[*c*[*...,* (*x*0*,..., xmi* )*. Qi/fi,.. .*]]]0*ρN*

= *£*[[*dc*]]*ρN*

again making use of Lemma [3.1](#_bookmark3), canonicalisation, and consistency.

So as *ρN* separates *dc* and *dc′* whenever *c* and *cj* are inequivalent, we see that the natural number polynomial function environment *ϕ*N and the natural number environment *ρ*N separate any two inequivalent elements of *C*, concluding the proof.*2*

We can now prove completeness relative to natural number polynomial function environments and natural number environments:

**Theorem 5.4 (Natural number completeness)** *For any expressions e and ej*

*we have:*

*|*=N *e* = *ej* =*⇒▶ e* = *ej*

**Proof.** Suppose that *|*=N *e* = *ej* but, for the sake of contradiction, that */▶ e* = *ej*. By canonicalisation (Lemma [4.3](#_bookmark7)) we then have */▶ c* = *cj*, where *c* = CF(*e*) and *cj* = CF(*ej*); so, by part (2) of Lemma [4.5](#_bookmark9), we have *c /≈ cj*. Therefore, by

Theorem [5.3](#_bookmark12), there is a natural number polynomial function environment *ϕ*N and a natural number environment *ρ*N such that *£*[[*c*]]*ϕ*N*ρ*N */*= *£*[[*cj*]]*ϕ*N*ρ*N, obtaining the required contradiction. *2*

We next upgrade part (2) of Lemma [4.5](#_bookmark9) to an equivalence and use that to analyse the theorems of our theory in terms of canonical forms and their *≈* relation; we also obtain a useful characterisation of the equivalence of atomic expressions in terms of the equivalence of their canonical subexpressions:

**Theorem 5.5** (i) *For any expressions e and ej we have:*

*▶ e* = *ej ⇐⇒* CF(*e*) *≈* CF(*ej*)

1. *For any canonical forms c and cj we have:*

*▶ c* = *cj ⇐⇒ c ≈ cj*

1. *For any atomic forms a and aj, a ≈ aj holds iff:*
   1. *a and aj are identical variables, or else*
   2. *for some function variable f* : *n, and for some m ∼ mj, they have the forms*

*fm*(*c*0*,..., cn—*1) *and fm′* (*cj ,..., cj* ) *with ci ≈ cj for i* = 0*,n —* 1*.*

0 *n—*1 *i*

**Proof.** For the second part, we already have the implication from right to left, by part (2) of Lemma [4.5](#_bookmark9). The other direction follows from consistency and separation. For the first part, by canonicalisation we have that *▶ e* = *ej* holds iff *▶* CF(*e*) = CF(*ej*) does, and conclusion follows from the second part. The third part follows immediately from the second part and the definition of the equivalence relation between atomic expressions. *2*

We next seek a local criterion for canonical expression equivalence. Let *A* be a set of atomic forms, and let *wa* (*a ∈ A*) be an *A*-indexed set of variables. We define polynomials *Pw,c* for canonical forms whose immediate atomic subexpressions are included in *A* by structural induction on *c*:

*Pw,r* = *r* (*r ∈* R) *Pw,a* = *wa Pw,c*+*c′* = *Pw,c* + *Pw,c′ Pw,cc′* = *Pw,cPw,c′*

and we say that *w* is *equivalence-characterising* if, for all *a, aj ∈ A*, we have:

*wa* = *wa′ ⇐⇒ a ≈ a*

*j*

**Lemma 5.6** *Let c, cj be canonical forms, and let A be a set of atomic forms in- cluding the immediate atomic subexpressions of c and cj and let wa* (*a ∈ A*) *be an equivalence-characterising A-indexed set of variables. Then:*

*c ≈ cj ⇐⇒ Pw,c ∼ Pw,c′*

**Proof.** It suffices to show that *Pc ∼ Pc′* holds iff *Pw,c ∼ Pw,c′* does. Choose *a*1*,..., ak ∈ A* such that *wa*1 *,..., wak* enumerates *{wa | a ∈ A}*. Then, as both *w* and *v* are equivalence-characterising, *va*1 *,..., vak* enumerates *{va|a ∈ A}*. A straightforward induction shows that *Pc*[*wa*1 */va*1 *,..., wak /vak* ]= *Pw,c*, and similarly for *cj*. Assuming *Pc ∼ Pc′* , we then find:

*Pw,c* = *Pc*[*wa*1 */va*1 *,..., wak /vak* ] *∼ Pc′* [*wa*1 */va*1 *,..., wak /vak* ]= *Pw,c′*

that is, *Pc ∼ Pc′* implies *Pw,c ∼ Pw,c′* . The converse is proved similarly. *2*

As we now show, the equivalence relation on canonical forms can be viewed as a combination of polynomial equivalence and the commutativity of partial differen- tiation with respect to different variables. We write

*▶*RTC *e* = *ej*

to mean that *e* and *ej* can be proved equal using only the ring axioms, the addition and multiplication tables, and the commutativity axiom for partial differentiation (i.e., without using any partial differentiation axioms other than commutativity).

**Lemma 5.7** *For any canonical form c, and set of atomic expressions A with* Im(*c*) *⊆ A and such that for all a, aj ∈ A if a ≈ aj then ▶*RTC *a* = *aj, and for any ≈-characterising A-indexed variable assignment w we have*

*▶*RTC *c* = *Pw,c*[*a*1*/va*1 *,.* *, an/van* ]

*where va*1 *,..., van are the variables of Pw,c for a*1*,.* *, an ∈ A.*

**Proof.** The proof is a simple structural induction on *c*. For an atomic form *a*, we have *Pw,c* = *va* and *va* = *va*1 , by assumption. So as *w* is *≈*-characterising, we have *a ≈ a*1 and so *▶*RTC *a* = *a*1, by assumption. *2*

**Theorem 5.8** *For any canonical forms c and cj we have:*

*▶ c* = *cj ⇐⇒ c ≈ cj ⇐⇒ ▶*RTC *c* = *cj*

**Proof.** We prove that for all *c, cj*

*c ≈ cj* =*⇒ ▶*RTC *c* = *cj*

by induction on max(*|c|, |cj|*), when the conclusion follows immediately from Theo- rem [5.5](#_bookmark14).

In the case where *c* (say) is an atomic expression *a*, then *Pc′ ∼ Pc* = *va*, and so

*▶*RTC *cj* = *aj* for some *aj ∈* Im(*cj*) with *va′* = *va*. As *va′* = *va*, we have *a ≈ aj*, and we then use the induction hypothesis to prove *▶*RTC *a* = *aj* using the characterisation of equivalence of atomic forms given by part (3) of Theorem [5.5](#_bookmark14).

Otherwise neither *c* nor *cj* is an atomic form and so, setting *A* = Im(*c*) *∪* Im(*cj*), by induction we have *▶*RTC *a* = *aj* whenever *a ≈ aj*, for *a, aj ∈ A*. Let *w* be any

*≈*-characterising *A*-indexed variable assignment. Applying Lemma [5.7](#_bookmark16) we find *▶*RTC *c* = *Pw,c*[*a*1*/va ,..., an/va* ] and *▶*RTC *cj* = *Pw,c′* [*a*1*/va ,..., an/va* ]. Further, by Lemma [5.6](#_bookmark15) , we have *Pw,c ~ Pw,c′* . It follows that *▶*RTC *Pw,c*[*a*1*/va*1 *,..., an/van* ]=

1 *n* 1 *n*

*j*

*Pw,c′* [*a*1*/va*1 *,..., an/van* ]. Putting these three things together we obtain *▶*RTC *c* = *c* ,

as desired. *2*

As may be evident, one can further strengthen the definition of canonical forms so that fewer axioms are needed to prove equivalence. For example, the partial differentiations in atomic expressions can be put in a standard order using the commutativity axiom. One can show that two such canonical forms are provably equal if, and only, if they can be proved so without using any axioms for partial differentiation; the proof of this fact parallels that of Theorem [5.8](#_bookmark17). Should one wish, one can go further and rewrite polynomials as sums of distinct multinomials; two such canonical forms are provably equal if, and only if, they can be proved so using only the associativity and commutativity of +.

We turn next to equational completeness. Let *fi* : *mi* (*i* = 1*, k*) and *xj* (*j* = 1*, l*) be the function variables and the free variables of two expressions *e* and *ej*. Then a *natural number counterexample* to the equation *e* = *ej* con- sists of natural number polynomials *Pi*(*y*0*,..., ymi* ) and natural numbers *k*1*,.* *, kl*

such that, the closed terms *e* =def *e*[*...,* (*y*0*,..., ymi* )*. Pi/fi,.. .*][*... kj/xj,.. .*] and

*ej* =def *ej*[*...,* (*y*0*,..., ym* )*. Pi/fi,.. .*][*... kj/xj,.. .*] are provably equal to different constants. As we now see, the natural number separation theorem yields such counterexamples, and thereby enables us to establish equational completeness:

*i*

**Theorem 5.9** *The theory of partial differentiation is equationally complete.*

**Proof.** Suppose that */▶ e* = *ej*. We show that adding the equation *e* = *ej* makes the theory equationally inconsistent. As */▶ e*0 = *e*1, by natural number polynomial completeness (Theorem [5.4](#_bookmark13)), there is a natural number polynomial function environ- ment *ϕ*, and a natural number environment *ρ* such that *£*[[*e*]]*ϕρ /*= *£*[[*ej*]]*ϕρ*. We can then define a natural number counterexample to the equation. Let *fi* : *mi* (*i* = 1*, k*) and *xj* (*j* = 1*, l*) be the function variables and the free variables of *e* and *ej*. For *i* = 1*, k*, take *Pi*(*y*0*,.* *, ymi* ) to be a natural number polynomial defining *ϕ*(*fi*) and

take *kj* to be *ρ*(*xj*), and define *e* and *ej* as above. By the substitution rule we

have *▶ e* = *ej*. By Lemma [3.1](#_bookmark3), we have *£*[[*e*]] = *£*[[*e*]]*ϕρ* and *£*[[*ej*]] = *£*[[*ej*]]*ϕρ*. So

*£*[[*e*]] */*= *£*[[*ej*]], and therefore by Corollary [4](#_bookmark8) and consistency, *e* and *ej* are provably equal to different constants.

Thus, assuming *▶ e* = *ej*, we can prove distinct real constants *£*[[*e* **]** and *£*[[*ej*]] equal. Using the addition and multiplication tables and the ring axioms, we then find that *▶* 1=0 and so *▶ x* = 0 and so *▶ x* = *y*. *2*

Instead of allowing all reals as constants, one could restrict them, for example to Q or none, except for the ring constants (equivalently N). The above development goes through straightforwardly in either of these cases, except for equational com- pleteness. The proof of equational completeness for Q goes through as the required polynomials, being over the natural numbers, and the required reals, being natural

numbers, are definable, and the argument from the equality of two distinct rationals to that of two distinct variables also goes through.

However, the last part of that argument does not go through for N. For example it is then consistent to add the equation 1 + 1 = 0. As a model, one can employ the *boolean differential calculus* [[11](#_bookmark30)]. One works over the boolean ring and defines the partial differentiation of boolean functions *h* : B*n →* B by:

*∂h*(*x*0*,...,xn−*1)

=

*∂xi*

*h*(*x*0*,..., xi—*1*,* 0*, xi*+1*,..., xn—*1)+ *h*(*x*0*,..., xi—*1*,* 1*, xi*+1*,..., xn—*1)

This can be equivalently written, somewhat more transparently, as:

*∂h*(*x*0*,...,xn−*1)

=

*∂xi*

*h*(*x*0*,..., xi—*1*, xi* + d*xi, xi*+1*,..., xn—*1) *— h*(*x*0*,..., xi—*1*, xi, xi*+1*,..., xn—*1)

taking d*xi* = 1. Another model of 1 + 1 = 0 can be constructed from the clone of polynomials in several variables over the boolean ring, with partial differentiation defined as usual on polynomials over a ring.

With the constants restricted to the rationals, the question of decidability of the equational theory of partial differentiation makes sense, and we have:

**Theorem 5.10** *With the constants restricted to the rationals, the equational theory of partial differentiation is decidable, and, further, natural number counterexamples to unprovable equations can be effectively obtained.*

**Proof.** We interleave two search procedures. One searches for a proof, the other searches through all possible natural number counterexamples. Since we know from the proof of equational completeness (Theorem [5.9](#_bookmark18)) that a natural number coun- terexample exists if an equation is not provable, this procedure will terminate, yielding either a proof or a counterexample. *2*

There is another, more direct, way to prove decidability and to find counterex- amples. For decidability, one uses the effectiveness of the proof of canonicalisation (Lemma [4.3](#_bookmark7)) to find canonical forms, and the characterisations of the equivalence of atomic expressions and canonical forms given by Theorem [5.5](#_bookmark14) and Lemma [5.6](#_bookmark15), to obtain a recursive algorithm to decide the equivalence of canonical forms.

With this decision procedure in hand, and with the observation that the proof in [[9](#_bookmark28)] of the Severi theorem is effective, one observes that the proof of polynomial separation (Theorem [5.2](#_bookmark11)) is effective. So too, therefore is the proof of natural num- ber separation (Theorem [5.3](#_bookmark12)), and, following the proof of equational completeness (Theorem [5.9](#_bookmark18)), we finally see that, if they exist, counterexamples can be found effectively.

# Acknowledgements

I thank Mart´ın Abadi, Jonathan Gallagher, and Tarmo Uustalu for useful discus- sions, and Sam Staton for helpful remarks including indicating how the axioms for partial differentiation could be made finitary.

# References

1. Barendregt, Hendrik P., “The Lambda Calculus: Its Syntax and Semantics”, Studies in Logic and the Foundations of Mathematics, **103**, 2nd edition, North-Holland, 1985.
2. Blute, Richard F., Robin Cockett, and Robert A.G. Seely, *Cartesian Differential Categories*, Theory and Applications of Categories, **22** (2009), 622–672.
3. Cockett, Robin and Geoff Cruttwell, *Differential Structure, Tangent Structure, and SDG*, Applied Categorical Structures, **22**(2) (2014), 331–417.
4. Cockett, Robin, Geoff Cruttwell, Jonathan Gallagher, Jean-Simon Pacaud Lemay, Benjamin MacAdam, Gordon Plotkin, and Dorette Pronk, *Reverse Derivative Categories*, Proc. 28th. Annual Conference on Computer Science Logic (eds. Maribel Fern´andez and Anca Muscholl), LIPIcs, **152**, 18:1-18:16, Schloss Dagstuhl, 2020.
5. Cohn, Paul Moritz, “Universal algebra”, Mathematics and Its Applications, **6**, 2nd edn., Dordrecht: Reidel, 1981.
6. Fiore, Marcelo and Chung-Kil Hur, *Second-order equational logic*, Proc. 24th. International Workshop on Computer Science Logic (eds. Anuj Dawar and Helmut Veith), LNCS **6247**, 320–335, Springer, 2010.
7. Fiore, Marcelo and Ola Mahmoud, *Second-order algebraic theories*, Proc. 35th. International Symposium on Mathematical Foundations of Computer Science (eds. Hlinˇeny´ P., Kuˇcera A.), LNCS **6281**, 368–380, Springer, 2010.
8. Fiore, Marcelo, Gordon Plotkin, and Daniele Turi, *Abstract syntax and variable binding*, Proc. 14th. Annual IEEE Symposium on Logic in Computer Science, 193–202, 1999.
9. Lorentz, Rudolph A., “Multivariate Birkhoff Interpolation”, Lecture Notes in Mathematics, **1516**, Springer, 1992.
10. Lorentz, Rudolph A., *Multivariate Hermite interpolation by algebraic polynomials: A survey*, Journal of Computational and Applied Mathematics, **122** (2000), 167–201.
11. Posthoff, Christian, and Bernd Steinbach, “Logic Functions and Equations”, Springer, 2004.
12. Severi, Francesco, “Vorlesungen u¨ber Algebraische Geometrie”, Teubner, Berlin, 1921.
13. Staton, Sam, Dario Stein, Hongseok Yang, Nathanael L. Ackerman, Cameron E. Freer, and Daniel M. Roy, *The Beta-Bernoulli process and algebraic effects*, Proc. 45th. ICALP (eds. Ioannis Chatzigiannakis, Christos Kaklamanis, D´aniel Marx, and Donald Sannella), LIPIcs, **107**, 141:1–141:15, Schloss Dagstuhl, 2018.
14. Taylor, Walter, *Survey 79: Equational logic*, Houston Journal of Mathematics, **5**(S) (1979), 1–83.