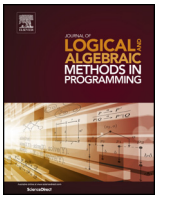
Journal of Logical and Algebraic Methods in Programming 135 (2023) 100892

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

journal homepage: www.elsevier.com/locate/jlamp



An axiomatic approach to differentiation of polynomial circuits

Paul Wilson [a,*], Fabio Zanasi [b,c,*]

[a] University of Southampton, University Road, Southampton, SO17 1BJ, United Kingdom
[b] University College London, Gower Street, London, WC1E 6BT, United Kingdom
[c] University of Bologna, Via Zamboni, Bologna, 40126, Italy

ARTICLE INFO

Article history:
Received 31 December 2022
Received in revised form 21 June 2023
Accepted 22 June 2023
Available online 30 June 2023

ABSTRACT

Reverse derivative categories (RDCs) have recently been shown to be a suitable semantic framework for studying machine learning algorithms. Whereas emphasis has been put on training methodologies, less attention has been devoted to particular *model classes*: the concrete categories whose morphisms represent machine learning models. In this paper we study presentations by generators and equations of classes of RDCs. In particular, we propose *polynomial circuits* as a suitable machine learning model class. We give an axiomatisation for these circuits and prove a functional completeness result. Finally, we discuss the use of polynomial circuits over specific semirings to perform machine learning with discrete values.

1. Introduction

Reverse Derivative Categories [10] have recently been introduced as a formalism to study abstractly the concept of differentiable functions. As explored in [11], it turns out that this framework is suitable to give a categorical semantics for gradient-based learning. In this approach, models–as for instance neural networks–correspond to morphisms in some RDC. We think of the particular RDC as a 'model class'–the space of all possible definable models.

However, much less attention has been directed to actually defining the RDCs in which models are specified: existing approaches assume there is some chosen RDC and morphism, treating both essentially as a black box. In this paper, we focus on classes of RDCs which we call 'polynomial circuits', which may be thought of as a more expressive version of the boolean circuits of Lafont [17], with wires carrying values from an arbitrary semiring instead of $\mathbb{Z}_2$. Because we ensure polynomial circuits have RDC structure, they are suitable as machine learning models, as we discuss in the second part of the paper.

Our main contribution is to provide an algebraic description of polynomial circuits and their reverse derivative structure. More specifically, we build a presentation of these categories by operations and equations. Our approach will proceed in steps, by gradually enriching the algebraic structures considered, and culminate in showing that a certain presentation is *functionally complete* for the class of functions that these circuits are meant to represent.

An important feature of our categories of circuits is that morphisms are specified in the graphical formalism of *string diagrams*. This approach has the benefit of making the model specification reflects its combinatorial structure. Moreover, at a computational level, the use of string diagrams makes available the principled mathematical toolbox of *double-pushout*

* Corresponding authors.
E-mail addresses: paul@statusfailed.com (P. Wilson), f.zanasi@ucl.ac.uk (F. Zanasi).

https://doi.org/10.1016/j.jlamp.2023.100892

rewriting, via an interpretation of string diagrams as hypergraphs [6–8]. Finally, the string diagrammatic presentation suggests a way to encode polynomial circuits into datastructures: an important requirement for being able to incorporate these models into tools analogous to existing deep learning frameworks such as TensorFlow [1] and PyTorch [19]. Concretely, because polynomial circuits are presented by generators and equations, it is straightforward to represent them using datastructures such as those described in [23] and [26], with the latter also giving a scalable algorithm for taking reverse derivatives.

Tool-building is not the only application of the model classes we define here. Recent neural networks literature [4,9] proposes to improve model performance (e.g. memory requirements, power consumption, and inference time) by 'quantizing' network parameters. One categorical approach in this area is [24], in which the authors define learning directly over boolean circuit models instead of training with real-valued parameters and then quantizing. The categories in our paper can be thought of as a generalisation of this approach to arbitrary semirings.

This generalisation further yields another benefit: while neural networks literature focuses on finding particular 'architectures' (i.e. specific morphisms) that work well for a given problem, our approach suggests a new avenue for model design: changing the underlying semiring (and thus the corresponding notion of arithmetic). To this end, we conclude our paper with some examples of finite semirings which may yield new approaches to model design.

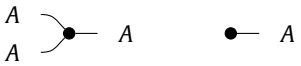
Synopsis We recall the notion of RDC in Section 2, and then study presentations of RDCs by operations and equations in Section 3. We define categories of polynomial circuits in Section 4, before showing how they can be made *functionally complete* in Section 5. Finally, we close by discussing some case studies of polynomial circuits in machine learning, in Section 6.

This work extends the conference paper [25] with more detailed background on reverse derivative categories (Sections 2.1 - 2.3) additional results on cartesian distributive categories (Section 3), a new part describing polynomial circuits over rings (Section 4.1, Appendix C) as well as graphical proofs omitted from [25] (Appendices B and D).

2. Reverse derivative categories

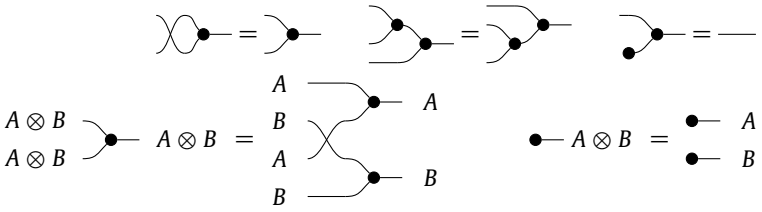
We recall the notion of reverse derivative category [10] in two steps. First we introduce the simpler structure of cartesian left-additive categories. We make use of the graphical formalism of *string diagrams* [20] to represent morphisms in our categories.

Definition 2.1. A **Cartesian Left-Additive Category** ([10], [5]) is a cartesian category in which each object A is equipped with a commutative monoid and zero map:

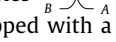


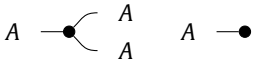
(1)

so that



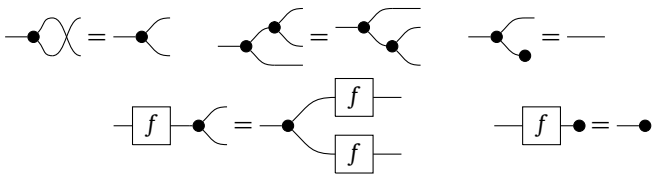
(2)

Note that the category being cartesian means that: (I) it is symmetric monoidal, namely for each object A and B there are symmetries  and identities  satisfying the laws of symmetric monoidal categories [20]; (II) each object A comes equipped with a *copy* and a *discard* map:

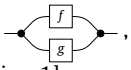


(3)

satisfying the axioms of commutative comonoids and natural with respect to the other morphisms in the category:

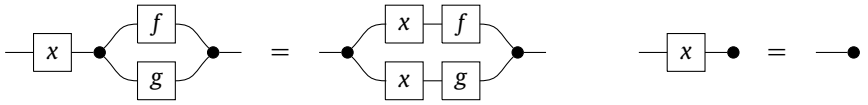


(4)

Remark 2.2. Definition 2.1 is given differently than the standard definition of cartesian left-additive categories [10, Definition 1], which one may recover by letting addition of morphisms be $f + g :=$  , and the zero morphism be $0 :=$. Equations of cartesian left-additive categories as given in [10, Definition 1]

$$x \fatsemi (f + g) = (x \fatsemi f) + (x \fatsemi g) \qquad x \fatsemi 0 = 0$$

are represented by string diagrams



and follow from Definition 2.1 thanks to the naturality of  and  , respectively. We refer to [5, Proposition 1.2.2 (iv)] for more details on the equivalence of the two definitions.

2.1. Reverse derivative categories

We can now recall the original definition of Reverse Derivative Categories first given in [10, Definition 13]. Note that we will give an equivalent, string-diagrammatic reformulation of the same definition in Definition 2.5.

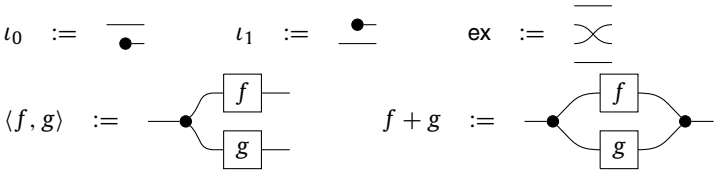
Definition 2.3 *(Reverse Differential Category [10]).* A **Reverse Differential Category** (RDC) is a cartesian left-additive category equipped with a combinator R of the following type

$$\frac{A \xrightarrow{f} B}{A \times B \xrightarrow[\mathsf{R}[f]]{} A}$$

satisfying axioms [**RD.1**] – [**RD.7**]:

[**RD.1**] $R[f+g] = R[f] + R[g]$ and $R[0] = 0$
[**RD.2**] $\langle a, b+c\rangle \fatsemi \mathsf{R}[f] = \langle a, b\rangle \fatsemi \mathsf{R}[f] + \langle a, c\rangle \fatsemi \mathsf{R}[f]$ and $\langle a, 0\rangle \fatsemi \mathsf{R}[f] = 0$
[**RD.3**] $\mathsf{R}[\mathsf{id}] = \pi_1$ and $\mathsf{R}[\pi_0] = \pi_1 \fatsemi \iota_0$ and $\mathsf{R}[\pi_1] = \pi_1 \fatsemi \iota_1$
[**RD.4**] $\mathsf{R}[\langle f, g\rangle] = (\mathsf{id} \times \pi_0) \fatsemi \mathsf{R}[f] + (\mathsf{id} \times \pi_1) \fatsemi \mathsf{R}[g]$ and $\mathsf{R}[!_A] = 0$
[**RD.5**] $\mathsf{R}[f \fatsemi g] = \langle \pi_0, \langle \pi_0 \fatsemi f, \pi_1\rangle\rangle \fatsemi (\mathsf{id} \times \mathsf{R}[g]) \fatsemi \mathsf{R}[f]$
[**RD.6**] $\langle \mathsf{id} \times \pi_0, 0 \times \pi_1\rangle \fatsemi (\iota_0 \times \mathsf{id}) \fatsemi \mathsf{R}[\mathsf{R}[\mathsf{R}[f]]] \fatsemi \pi_1 = (\mathsf{id} \times \pi_1) \fatsemi \mathsf{R}[f]$
[**RD.7**] $(\iota_0 \times \mathsf{id}) \fatsemi \mathsf{R}[\mathsf{R}[(\iota_0 \times \mathsf{id}) \fatsemi \mathsf{R}[\mathsf{R}[f]] \fatsemi \pi_1]] \fatsemi \pi_1$
$= \mathsf{ex} \fatsemi (\iota_0 \times \mathsf{id}) \fatsemi \mathsf{R}[\mathsf{R}[(\iota_0 \times \mathsf{id}) \fatsemi \mathsf{R}[\mathsf{R}[f]] \fatsemi \pi_1]] \fatsemi \pi_1$

This definition makes reference to several morphisms which we define below in string-diagrammatic form, omitting types for the sake of readability.



Note that, as observed in [10], the maps ι_0 and ι_1 are not in general injections of the coproduct.

Intuitively, for a morphism $f : A \to B$ we think of its reverse derivative $R[f] : A \times B \to A$ as approximately computing the change of *input* to f required to achieve a given change in *output*. That is, if f is a function, we should have

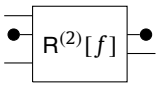
$$f(x) + \delta_y \approx f(x + \mathsf{R}[f](x, \delta_y))$$

The authors of [10] go on to show that any reverse derivative category also admits a *forward* differential structure: i.e., it is also a Cartesian Differential Category (CDC). This means the existence of a forward differential operator satisfying various axioms, and having the following type:

$$\frac{A \xrightarrow{f} B}{A \times A \xrightarrow[\mathsf{D}[f]]{} B}$$

In an RDC, the forward differential operator D is defined in terms of the reverse differential operator R as follows.

Definition 2.4 *(Induced Forward Differential Operator [10]).* In a Cartesian Reverse Differential category with reverse derivative operator R, the induced *forward* differential operator D is defined as

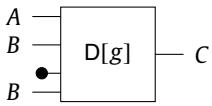
$\mathsf{D}[f] \quad := \quad$ 

where $R^{(2)}[f]$ denotes the 2-fold application of R, i.e., the map $R[R[f]]$.

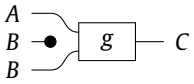
In contrast to the R operator, we think of D as computing a change in *output* from a given change in *input*, whence 'forward' and 'reverse' derivative:

$$f(x + \delta_x) \approx f(x) + \mathsf{D}[f](x, \delta_x)$$

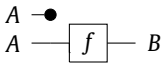
The final pieces we need to state our definition of RDCs are the (cartesian differential) notions of *partial derivative* and *linearity* defined in [10]. Graphically, the partial derivative of $g : A \times B \to C$ with respect to B is defined as follows:

$\mathsf{D}_B[g] \quad := \quad$ 

Finally we say that g is *linear in* B when

$\mathsf{D}_B[g] \quad = \quad$ 

and that $f : A \to B$ is *linear* when

$\mathsf{D}[f] \quad = \quad$ 

2.2. Alternative axiomatisation of RDCs

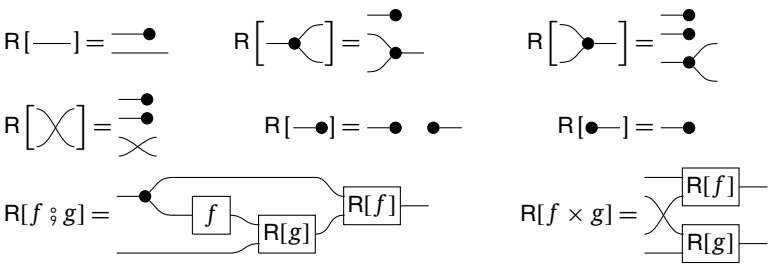
We can now formulate our alternative definition of RDCs. Note that in the following definition and proofs we treat D purely as a syntactic shorthand for its definition in terms of R. We avoid use of CDC axioms to prevent a circular definition, although one can derive them as corollaries of the RDC axioms.

Definition 2.5. A **Reverse Derivative Category** is a cartesian left-additive category equipped with a *reverse differential combinator* R:

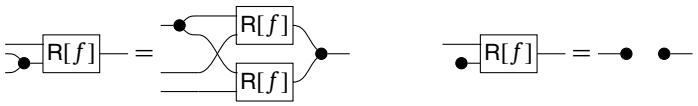
$$\frac{A \xrightarrow{f} B}{A \times B \xrightarrow[\mathsf{R}[f]]{} A}$$

satisfying the following axioms:

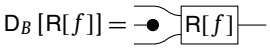
[ARD.1] (Structural axioms, equivalent to RD.1, RD.3-5 in [10])



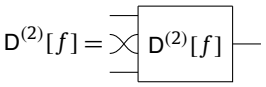
[ARD.2] (Additivity of change, equivalent to RD.2 in [10])



[ARD.3] (Linearity of change, equivalent to RD.6 in [10])



[ARD.4] (Symmetry of partials, equivalent to RD.7 in [10])



Remark 2.6. Note that we may alternatively write axioms ARD.3 and ARD.4 directly in terms of the R operator by simply expanding the syntactic definition of D.

Note that axioms ARD.1 and ARD.2 are quite different to that of [10], while ARD.3 and ARD.4 are essentially direct restatements in graphical language of RD.6 and RD.7 respectively.

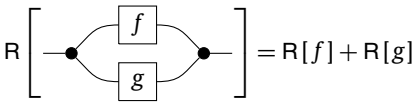
The definition we provide best suits our purposes, although it is different than the standard one provided in [10, Definition 13]. We can readily verify that they are equivalent.

Theorem 2.7. *Definitions 2.3 and 2.5 are equivalent.*

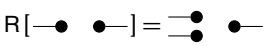
Proof. Axioms ARD.3-4 are direct statements of axioms RD.6-7, so it suffices to show that we can derive axioms ARD.1-2 from RD.1.5 and vice-versa. The structural axioms ARD.1 follow directly from RD.1 and RD.3-5.

- For $\mathsf{R}[\,\_\,]$ use RD.3 directly.
- For $\mathsf{R}\left[\times\right]$, apply RD.4 to $\langle \pi_1, \pi_0 \rangle$
- For $\mathsf{R}\left[\right]$, apply RD.1 to $\pi_0 + \pi_1$
- For $\mathsf{R}[\,\bullet\!\!-\,]$, apply RD.1 directly.
- For $\mathsf{R}\left[\right]$, apply RD.4 to $\langle \mathsf{id}, \mathsf{id} \rangle$
- For $\mathsf{R}[\,\_\bullet\,]$, apply RD.4 directly.
- For composition $f \fatsemi g$, apply RD.5 directly
- For tensor $f \times g$, apply RD.4 to $\langle \pi_0 \fatsemi f, \pi_1 \fatsemi g \rangle$

In the reverse direction, we can obtain RD.1 and RD.3-5 by simply constructing each equation and showing it holds given the structural equations. For example, RD.1 says that $\mathsf{R}[f+g] = \mathsf{R}[f] + \mathsf{R}[g]$ and $\mathsf{R}[0] = 0$, which we can write graphically as:



and



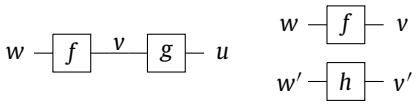
ARD.2 can be derived from RD.2 by setting a, b, c to appropriate projections, and in the reverse direction we can obtain RD.2 simply by applying ARD.2 to its left-hand-side and using naturality of . □

A main reason to give an alternative formulation of cartesian left-additive and reverse derivative categories is being able to work with a more 'algebraic' definition, which revolves around the interplay of operations , , , and . This perspective is particularly useful when one wants to show that the free category on certain generators and equations has RDC structure. We thus recall such free construction, referring to [27, Chapter 2] and [3, Section 5] for a more thorough exposition.

Definition 2.8. Given a set *Obj* of *generating objects*, we may consider a set Σ of *generating morphisms* $f : w \to v$, where the *arity* $w \in Obj^\star$ and the *coarity* $v \in Obj^\star$ of f are *Obj*-words. Cartesian left-additive Σ-*terms* are defined inductively:

- Each $f : w \to v$ is a Σ-term.
- For each $A \in Obj$, the generators (1) and (3) of the cartesian left-additive structure are Σ-terms.

- If $f\colon w \to v$, $g\colon v \to u$, and $h\colon w' \to v'$ are Σ-terms, then $f \fatsemi g\colon w \to u$ and $f \otimes h\colon ww' \to vv'$ are Σ-terms, represented as string diagrams



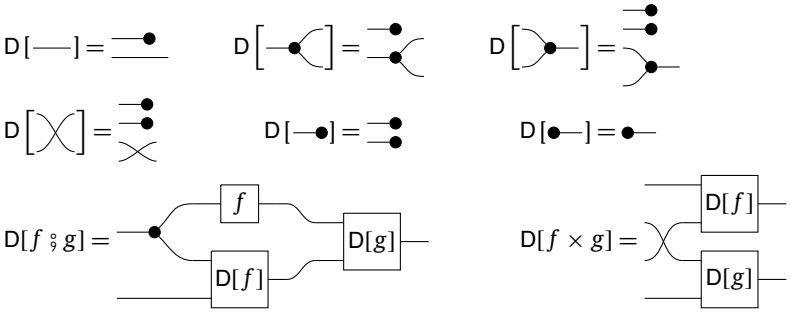
Let us fix Obj, Σ and a set E of equations between Σ-terms. The cartesian left-additive category $\mathscr{C}$ freely generated by (Obj, Σ, E) is the monoidal category with set of objects $Obj^\star$ and morphisms the Σ-terms quotiented by the axioms (2) and (4) of cartesian left-additive categories and the equations in E. The monoidal product in $\mathscr{C}$ is given on objects by word concatenation. Identities, monoidal product and sequential composition of morphisms are given by the corresponding Σ-terms and their constructors $f \otimes h$ and $f \fatsemi g$.

One may readily see that $\mathscr{C}$ defined in this way is indeed cartesian left-additive. We say that $\mathscr{C}$ is *presented* by generators (Obj, Σ) and equations E.

2.3. Forward derivative structure

Although we stop short of a complete proof that reverse differential structure induces forward differential structure, we will now show that the forward differential operator D can nevertheless be viewed as being defined *inductively* on the generating operations of an RDC presented by generators and equations. The following proposition is therefore analogous to our alternative reaxiomatisation of RDCs in Definition 2.5.

Proposition 2.9. *In an RDC with forward differential operator* D *defined in terms of* R *as in Definition 2.4, we have the following equalities.*



Proof. One may calculate the action of D on generators of cartesian left-additive categories by applying Definition 2.4 directly to each generator and simplifying the result. One can then show inductively that the action of D on composition and tensor product of morphisms is as given above. Concretely, assume that the equality in Definition 2.4 holds for two arbitrary morphisms. Then it also holds for their composition and tensor product, which we prove in Appendix B (see Propositions B.2 and B.1 respectively). □

Note however that this proposition is *not* a definition: it is an immediate consequence of the definition of D in terms of R given in Definition 2.4.

3. Reverse derivatives and algebraic presentations

As we will see in Section 5, our argument for functional completeness relies on augmenting the algebraic presentation of polynomial circuits with an additional operation. To formulate such result, we first need to better understand how reverse differential combinators may be defined compatibly with the generators and equations presenting a category.

Theorem 3.1. *Let $\mathscr{C}$ be the cartesian left-additive category presented by generators (Obj, Σ) and equations E. If for each $s \in \Sigma$ there is some $\mathsf{R}[s]$ which is well-defined (see Remark 3.2) with respect to E, and which satisfies axioms ARD.1-4, then $\mathscr{C}$ is a reverse derivative category.*

Proof. Observe that axioms ARD.1 fix the definition of R on composition, tensor product and the cartesian and left-additive structures. It therefore suffices to show that axioms ARD.2-4 are preserved by composition and tensor product. That is, for morphisms f, g of appropriate types, both $f \fatsemi g$ and $f \otimes g$ preserve axioms ARD.2-4. Thus, any morphism constructed from

generators must also satisfy the axioms ARD.1-4, and $\mathscr{C}$ must be an RDC. We provide the full graphical proofs that ARD.2-4 are preserved by composition and tensor product in Appendix A. □

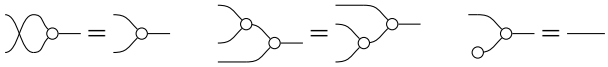
Remark 3.2. In the statement of Theorem 3.1, strictly speaking $s \in \Sigma$ is just a representative of the equivalence class of Σ-terms (modulo E plus the laws of left-additive cartesian categories) defining a morphism in $\mathscr{C}$. Because of this, we require $\mathsf{R}[s]$ to be 'well-defined', in the sense that if s and t are representatives of the same morphisms of $\mathscr{C}$, then the same should hold for $\mathsf{R}[s]$ and $\mathsf{R}[t]$. In a nutshell, we are allowed to define R directly on Σ-terms, provided our definition is compatible with E and the laws of left-additive cartesian categories.

An immediate consequence of Theorem 3.1 is that if we have a presentation of a category $\mathscr{C}$ with RDC structure, we can 'freely extend' it with an additional operation s, a chosen reverse derivative $\mathsf{R}[s]$, and equations E', so long as R is well-defined with respect to E' and the axioms ARD.2-4 hold for $\mathsf{R}[s]$. Essentially, this gives us a simple recipe for adding new 'gadgets' to existing RDCs and ensuring they retain RDC structure.

One particularly useful such 'extension' is the addition of a *multiplication* morphism  that distributes over the addition  . We define categories with such a morphism as an extension of cartesian left-additive categories as follows:

Definition 3.3. A **Cartesian Distributive Category** is a cartesian left-additive category such that each object A is equipped with a commutative monoid  and unit  which distributes over the addition  . More completely, it is a category having morphisms

satisfying the *cartesianity* equations (4), the *left-additivity* equations (2), the *multiplicativity* equations

 (5)

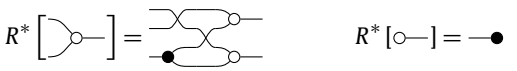
and the *distributivity* and *annihilation* equations

 (6)

Just as for cartesian left-additive categories, one may construct cartesian distributive categories freely from a set of objects *Obj*, a signature Σ, and equations E, the difference being that Σ-term will be constructed using also and , and quotiented also by (5)-(6). The main example of cartesian distributive categories are *Polynomial Circuits*, which we define in Section 4 below.

Reverse derivative categories define a reverse differential combinator on a left-additive cartesian structure. As cartesian distributive categories extend left-additive ones, it is natural to ask how we may extend the definition of the reverse differential combinator to cover the extra operations and . The following theorem provides a recipe, which we will use in the next section to study RDCs with a cartesian distributive structure. Note that the definition of R^* below is a string diagrammatic version of the reverse derivative combinator defined on POLY in [10].

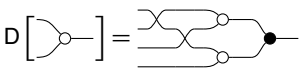
Theorem 3.4. *Suppose $\mathscr{C}$ is a cartesian left-additive category presented by (Obj, Σ, E), and assume $\mathscr{C}$ is also an RDC, say with reverse differential combinator R. Then the cartesian distributive category $\mathscr{C}^*$ presented by (Obj, Σ, E), with reverse differential combinator R^* defined as R on the left-additive cartesian structure, and as follows*

 (7)

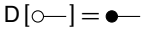
on the extra distributive structure, is also an RDC.

Proof. It suffices to check that R is well-defined with respect to the additional equations of cartesian distributive categories, and that the new generators and satisfy axioms ARD.2-4. We give diagrammatic proofs of the latter in Appendix D.1 □

Finally, observe that Definition 2.4 fixes the forward differential operator on as



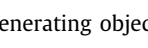
and on as

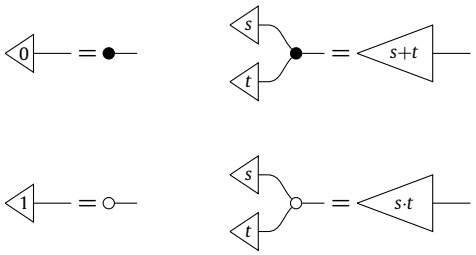


We give graphical proofs of these equalities in Propositions D.6 and D.7, respectively.

4. Polynomial circuits

Our motivating example of cartesian distributive categories is that of *polynomial circuits*, whose morphisms can be thought of as representing polynomials over a commutative semiring. We define them as follows:

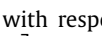
Definition 4.1. Let S be a commutative semiring. We define $\mathsf{PolyCirc}_S$ as the cartesian distributive category presented by (I) one generating object 1, (II) for each $s \in S$, a generating morphism  : $0 \to 1$, (III) the 'constant' equations



(8)

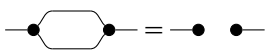
for $s, t \in S$, intuitively saying that the generating morphisms respect addition and multiplication of S.

Proposition 4.2. $\mathsf{PolyCirc}_S$ *is an RDC with* $\mathsf{R}\left[\text{  }\right] =$  .

Proof. The type of $\mathsf{R}\left[\text{  }\right] : 1 \to 0$ implies that there is only one choice of reverse derivative, namely the unique discard map  . Furthermore, R is well-defined with respect to the constant equations (8) for the same reason. Finally, observe that the axioms ARD.2-4 hold for $\mathsf{R}\left[\text{  }\right]$, precisely in the same way as for $\mathsf{R}[$  $]$, and so $\mathsf{PolyCirc}_S$ is an RDC. □

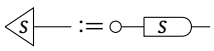
Although our Definition 4.1 of $\mathsf{PolyCirc}_S$ requires that we add an axiom for each possible addition and multiplication of constants, for some significant choices of S an equivalent smaller finite axiomatisation is possible. We demonstrate this with some examples.

Example 4.3. In the case of $\mathsf{PolyCirc}_{\mathbb{Z}_2}$, the equations of Definition 4.1 reduce to the single equation



expressing that $x + x = 0$ for both elements of the field $\mathbb{Z}_2$.

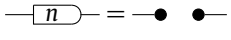
Example 4.4. In the case $\mathsf{PolyCirc}_{\mathbb{N}}$ of the semiring of natural numbers, with the usual addition and multiplication, no extra generating morphisms or equations are actually necessary: all those appearing in Definition 4.1 may be derived from the cartesian distributive structure. To see why, notice that we may define each constant $s \in S$ as repeated addition:



where we define inductively as

The equations expressing addition and multiplication in $\mathbb{N}$ are then a consequence of those of cartesian distributive categories. In fact, from this observation we have that $\mathsf{PolyCirc}_{\mathbb{N}}$ is the free cartesian distributive category on one generating object.

Example 4.5. In a straightforward generalization of $\mathsf{PolyCirc}_{\mathbb{Z}_2}$, we can define $\mathsf{PolyCirc}_{\mathbb{Z}_n}$ in the same way, but with the only additional equation as



which says algebraically that $(1 + .^{n}. + 1) \cdot x = n \cdot x = 0 \cdot x = 0$.

It is important to note that $\mathsf{PolyCirc}_S$ is isomorphic to the category POLY_S, defined as follows:

Definition 4.6. POLY_S is the symmetric monoidal category with objects the natural numbers and arrows $m \to n$ the n-tuples of polynomials in m indeterminates:

$$\langle p_1(\vec{x}), \ldots, p_n(\vec{x}) \rangle : m \to n$$

with each

$$p_i \in S[x_1, \ldots, x_m]$$

where $S[x_1, \ldots x_m]$ denotes the polynomial ring in m indeterminates over S.

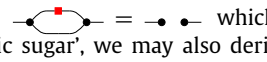
The isomorphism $\mathsf{PolyCirc}_S \cong \mathsf{POLY}_S$ is constructed by using that homsets $\mathsf{PolyCirc}_S(m,n)$ and $\mathsf{POLY}_S(m,n)$ have the structure of the free module over the polynomial ring $S[x_1 \ldots x_m]^n$ which yields a unique module isomorphism between them. We do not prove this isomorphism here, other than to say that it follows by the same argument as presented in [24, Appendix A].

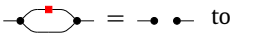
Remark 4.7. Note in [10] POLY_S is proven to be a reverse derivative category, meaning that we could have derived Proposition 4.2 as a corollary of the isomorphism $\mathsf{PolyCirc}_S \cong \mathsf{POLY}_S$. We chose to provide a 'native' definition of the reverse differential combinator of $\mathsf{PolyCirc}_S$ because–as we will see shortly–we will need to extend it with an additional generator. The reason for this is to gain the property of 'functional completeness', which will allow us to express any function $S^m \to S^n$. This new derived category will in general no longer be isomorphic to POLY_S, and so we must prove it too is an RDC: we do this straightforwardly using Theorem 3.1.

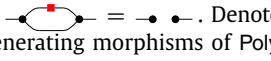
4.1. Polynomial circuits over a ring

We now discuss polynomial circuits over rings. In particular, we will see that by freely augmenting $\mathsf{PolyCirc}_S$ with an additional negation operation and equation we 'upgrade' a semiring S to a ring.

When S is already a ring, we may regard $\mathsf{PolyCirc}_S$ as having a morphism representing negation. We define this morphism as multiplication by -1:

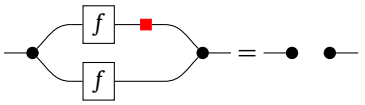
One can then derive (as in the proof of Proposition C.2) that  which says that negation  functions as an additive inverse. Since this definition is merely 'syntactic sugar', we may also derive its reverse derivative $\mathsf{R}[-\blacksquare-] =$  and forward derivative $\mathsf{D}[-\blacksquare-] =$  .

More interestingly, given a *semiring* S we may also *freely add* a generator  and equation  to $\mathsf{PolyCirc}_S$.

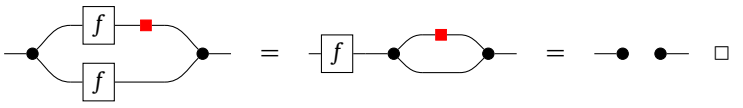
Definition 4.8 ($\mathsf{PolyCirc}_S^-$). Let S be a semiring. $\mathsf{PolyCirc}_S^-$ is generated by the operations and equations of $\mathsf{PolyCirc}_S$ with the addition of a single operation  and equation  . Denote by $\mathsf{Ring} : \mathsf{PolyCirc}_S \to \mathsf{PolyCirc}_S^-$ the functor defined inductively with $F(g) = g$ for each of the generating morphisms of $\mathsf{PolyCirc}_S$.

Observe that the hom-sets $\mathsf{PolyCirc}_S^-$ are upgraded from semirings to rings:

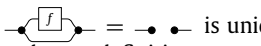
Proposition 4.9. *Let $f : A \to B$ be a morphism in $\mathsf{PolyCirc}_S^-(A, B)$. Then*  *is the additive inverse of f, i.e.:*



Proof.



□

We may then derive the $\mathsf{PolyCirc}_S^-$ equivalents of elementary ring properties (see Appendix C). An important example of such a property is the uniqueness of negation (Proposition C.1). This says that any morphism  satisfying the equation  is unique. From this property, we can derive in Proposition C.2 that which we formerly took as a definition.

The inclusion of the additional generator amounts to the group completion [21, Chapter 2] of the additive structure of each hom-set $\mathsf{PolyCirc}_S(A, B)$. To see why, first note that cartesian left-additive functors [5, Definition 1.3.1] preserve negation.

Proposition 4.10 *(Cartesian Left-Additive functors preserve negation). Let $\mathscr{C}$ and $\mathscr{D}$ be cartesian left-additive categories with a negation morphism satisfying the equation . If $F : \mathscr{C} \to \mathscr{D}$ is a cartesian left-additive functor, then $F(-) = -$.*

Proof. For F to be well-defined, we must have that

F $=$ F

Moreover, since F is *cartesian left-additive*, it preserves the generating morphisms of cartesian left-additive categories, so we may apply F to both sides of this equation to obtain

$F(-)$ $=$

which says algebraically that $x + (-x) = 0$. That is, $F(-)$ is an additive inverse, and so by the uniqueness of negations (Proposition C.1), we can conclude that

$F(-)$ $=$ $-$ □

We may then express the idea of group completion in terms of polynomial circuits as follows.

Proposition 4.11 *(Group Completion of Polynomial Circuits). Let S be a semiring and R a ring, and suppose $F : \mathsf{PolyCirc}_S \to \mathsf{PolyCirc}_R$ is a cartesian left-additive functor that also preserves cartesian distributive structure. That is, assume that $F(\,\cdot\,) = \,\cdot\,$ and $F(\circ-) = \circ-$. Then there is a unique functor $\tilde{F} : \mathsf{PolyCirc}_S^- \to \mathsf{PolyCirc}_R$ preserving cartesian distributive structure such that $\mathsf{Ring}\,;\tilde{F} = F$.*

Proof. The definition of F and $\tilde{F}$ is fixed on cartesian distributive structure, and so we can define $\tilde{F}$ solely in terms of the 'constant' generators of $\mathsf{PolyCirc}_S$. Therefore, define $\tilde{F}(\,\cdot\,) := F(\,\cdot\,)$ and so

$\tilde{F}(\mathsf{Ring}(\,\cdot\,))$ $=$ $\tilde{F}(\,\cdot\,)$ $=$ $F(\,\cdot\,)$

The uniqueness of $\tilde{F}$ follows because its definition must agree with F on each or else $\mathsf{Ring}\,;\tilde{F} = F$ does not hold. Moreover, its definition on cartesian distributive structure is fixed, and finally by Proposition 4.10 there is exactly one possible definition of $\tilde{F}(-)$. □

Example 4.12. The extension of polynomial circuits over the naturals $\mathsf{PolyCirc}_{\mathbb{N}}$ with a negation operation yields the category $\mathsf{PolyCirc}_{\mathbb{N}}^-$, which is precisely $\mathsf{PolyCirc}_{\mathbb{Z}}$: polynomial circuits over the *integers*. In other words, the cartesian distributive category with a single generating object, negation, and the equation $\quad = \quad$ is the category $\mathsf{PolyCirc}_{\mathbb{Z}}$.

Finally, we should verify that the extension of $\mathsf{PolyCirc}_S$ to $\mathsf{PolyCirc}_S^-$ retains its reverse derivative structure. Once again, we can appeal to Theorem 3.1, and define R on negation as follows.

$\mathsf{R}[-] :=$

In Appendix C we show that this extension is well-defined (Proposition C.7) and satisfies axioms ARD.2-4 (Propositions C.8, C.9, and C.10, respectively).

Finally, note that the forward derivative $\mathsf{D}[-] =$ is completely in terms of R, and so no additional proofs are required.

5. Functional completeness

We are now ready to consider the *expressivity* of the model class of polynomial circuits. More concretely, for a given commutative semiring S, we would like to be able to represent any function between sets $S^m \to S^n$ as a string diagram in $\mathsf{PolyCirc}_S$. This property, which we call 'functional completeness', is important for a class of machine learning models to satisfy because it guarantees that we may always construct an appropriate model for a given dataset. It has been studied, for instance, in the context of the various 'universal approximation' theorems for neural networks (see e.g. [16], [18]).

To formally define functional completeness, let us fix a finite set S. We denote by FinSet_S the cartesian monoidal category whose objects are natural numbers and a morphism $m \to n$ is a function of type $S^m \to S^n$.

Definition 5.1. We say a category $\mathscr{C}$ is **functionally complete** with respect to a finite set S when there exists a full identity-on-objects functor $\mathsf{F} : \mathscr{C} \to \mathsf{FinSet}_S$.

The intuition for Definition 5.1 is that we call a category $\mathscr{C}$ 'functionally complete' when it suffices as a syntax for FinSet_S — that is, by fullness of F we may express any morphism in FinSet_S. Note however that two distinct morphisms in $\mathscr{C}$ may represent the same function — F is not necessarily faithful.

In general, $\mathsf{PolyCirc}_S$ is not functionally complete with respect to S. Take for example the boolean semiring $\mathbb{B}$ with multiplication and addition as AND and OR respectively. It is well known [22] that one cannot construct every function of type $\mathbb{B}^m \to \mathbb{B}^n$ from only these operations.

Nonetheless, we claim that in order to make $\mathsf{PolyCirc}_S$ functionally complete it suffices to add to its presentation just one missing ingredient: the 'comparator' operation, which represents the following function:

$$\mathsf{compare}(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

The following result clarifies the special role played by the comparator.

Theorem 5.2. *Let S be a finite commutative semiring. A category $\mathscr{C}$ is functionally complete with respect to S iff there is a monoidal functor $\mathsf{F} : \mathscr{C} \to \mathsf{FinSet}_S$ in whose image are the following functions:*

- $\langle\rangle \mapsto s$ *for each* $s \in S$ *(constants)*
- $\langle x, y\rangle \mapsto x + y$ *(addition)*
- $\langle x, y\rangle \mapsto x \cdot y$ *(multiplication)*
- compare

Proof. Suppose $\mathscr{C}$ is functionally complete with respect to S, where S is a finite commutative semiring. Then by definition there is a functor $\mathsf{F} : \mathscr{C} \to \mathsf{FinSet}_S$ with each of the required functions in its image.

Now in the reverse direction, we will show that any function can be constructed only from constants, addition, multiplication, and comparison. The idea is that because S is finite, we can simply encode the function table of any function $f : S^m \to S$ as the following expression:

$$x \mapsto \sum_{s \in S^m} \mathsf{compare}(s, x) \cdot f(s) \tag{9}$$

Further, since $\mathscr{C}$ is cartesian, we may decompose any function $f : S^m \to S^n$ into an n-tuple of functions of type $S^m \to S$. More intuitively, for each of the n outputs, we simply look up the appropriate output in the encoded function table. □

It follows immediately that $\mathsf{PolyCirc}_S$ is functionally complete with respect to S if and only if one can construct the compare function in terms of constants, additions, and multiplications. We illustrate one such case below.

Example 5.3. $\mathsf{PolyCirc}_{\mathbb{Z}_p}$ is functionally complete for prime p. To see why, recall Fermat's Little Theorem [12], which states that

$$a^{p-1} \equiv 1 (\mathrm{mod}\ p)$$

for all $a > 0$. Consequently, we have that

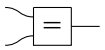
$$(p-1) \cdot a^{p-1} + 1 = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$$

We denote this function as $\delta(a) := (p-1) \cdot a^{p-1} + 1$ to evoke the dirac delta 'zero indicator' function. To construct the compare function is now straightforward:

$$\mathsf{compare}(x_1, x_2) = \sum_{s \in S} \delta(x_1 + s) \cdot \delta(x_2 + s)$$

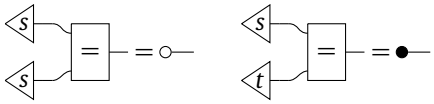
However, as we already observed, it is not possible in general to construct the compare function in terms of multiplication and addition. Therefore, to guarantee functional completeness we must *extend* the category of polynomial circuits with an additional comparison operation.

Definition 5.4. We define by $\mathsf{PolyCirc}_S^{=}$ as the cartesian distributive category presented by the same objects, operations, and equations of $\mathsf{PolyCirc}_S$, with the addition of a 'comparator' operation



(10)

and equations



(11)

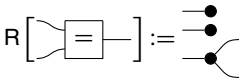
for $s, t \in S$ with $s \neq t$.

To make $\mathsf{PolyCirc}_S^{=}$ a reverse derivative category, we can once again appeal to Theorem 3.1. However, we must choose an apropriate definition of R[compare] which is well-defined and satisfies axioms ARD.1-4.

A suggestion for this choice comes from the machine learning literature. In particular, the use of the 'straight-through' estimator in quantized neural networks, as in e.g. [4]. Typically, these networks make use of the dirac delta function in the forward pass, but this causes a catastrophic loss of gradient information in the backwards pass since the gradient is zero almost everywhere. To fix this, one uses the *straight-through estimator*, which instead passes through gradients directly from deeper layers to shallower ones.

In terms of reverse derivatives, this amounts to setting $\mathsf{R}[\delta] = \mathsf{R}[\mathsf{id}]$. Of course, we need to define R for the full comparator, not just the zero-indicator function δ, and so we make the following choice:

Theorem 5.5. $\mathsf{PolyCirc}_S^{=}$ *is an RDC with* R *as for* $\mathsf{PolyCirc}_S$*, and*

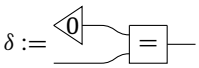


Proof. R is well-defined with respect to the equations (11) since both sides of each equation must equal the unique discard morphism —●. Further, R[=] satisfies axioms ARD.2-4 in the same way that R[●—] does, and so by Theorem 3.1 $\mathsf{PolyCirc}_S^{=}$ is a reverse derivative category. □

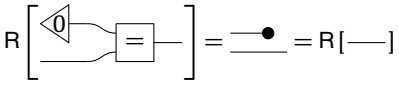
From Theorem 5.2, we may derive:

Corollary 5.6. $\mathsf{PolyCirc}_S^{=}$ *is functionally complete with respect to* S.

Finally, note that we recover the dirac delta function by 'capping' one of the comparator's inputs with the zero constant:



whose reverse derivative is equivalent to the 'straight-through' estimator:

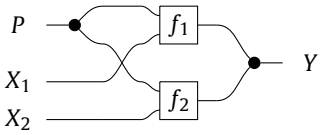


6. Polynomial circuits in machine learning: case studies

We now discuss the implications of some specific choices of semiring from a machine learning perspective. Let us begin with two extremes: neural networks, and the boolean circuit models of [24].

Neural networks We may think of a neural network as a circuit whose wires carry values in $\mathbb{R}$. Of course, in order to compute with such circuits we must make a finite approximation of the reals–typically using floating-point numbers. However, this approximation introduces two key issues. First, floating point arithmetic is significantly slower than integer arithmetic. Second, the floating point operations of addition and multiplication are not even associative, which introduces problems of *numerical instability*. Although attempts exist to address issues of floating point arithmetic (such as 'posits' [15]), these still do not satisfy the ring axioms; to properly account for these approximations would require additional work.

Boolean circuits and $\mathbb{Z}_2$ One may note that since we must always eventually deal with finite representations of values, we may as well attempt to define our model class directly in terms of them. This is essentially the idea of [24]: the authors use the category $\mathsf{PolyCirc}_{\mathbb{Z}_2}$ (which they call simply **PolyCirc**) as a model class since it is already functionally complete[1] and admits a reverse derivative operator. However, using a semiring of modular arithmetic in general introduces a different problem: one must be careful to construct models so that gradients do not 'wrap around'. Consider for example the model below, which can be thought of as two independent sub-models f_1 and f_2 using the same parameters[2] but applied to different parts of the input X_1 and X_2



Since $\mathsf{R}\left[\text{—}\right) = \mathsf{R}\left[\text{—}\right]$, when we compute the gradient update for P we will sum the gradients of f_1 and f_2. In the extreme case when the underlying semiring is $\mathbb{Z}_2$, then when the gradients of f_1 and f_2 are both 1, the result will 'wrap around' to 0 and P will not be updated. This is clearly undesirable: here we should prefer that $1+1=1$ to $1+1=0$.

Saturating arithmetic Another possible solution is to use the semiring Sat_n as a model of *saturating unsigned integer arithmetic* for a given 'precision' n. The underlying set is simply the finite set $\bar{\mathbf{n}}$, with addition and multiplication defined as for the naturals, but 'truncated' to at most $n-1$. We define Sat_n as follows, noting that it is equivalent to the semiring $B(n, n-1)$ first defined in [2, Example 3] (see also [14]).

Definition 6.1. The semiring Sat_n has as addition and multiplication the operations

$$x_1 + x_2 := \min(n-1, x_1 + x_2) \qquad x_1 \cdot x_2 := \min(n-1, x_1 \cdot x_2)$$

over the set $\bar{\mathbf{n}} := \{0 \dots n-1\}$

Note that while Sat_n is a commutative semiring, it is certainly *not* a ring: the introduction of inverses means that the associativity axiom of semirings is violated.

Finally, note that for each of these choices of semiring S, in general $\mathsf{PolyCirc}_S$ is not functionally complete. Thus, in order to obtain a model class which is functionally complete and is a reverse derivative category, we must use $\mathsf{PolyCirc}_S^{=}$.

7. Conclusions and future work

In this paper, we studied in terms of algebraic presentations categories of polynomial circuits, whose reverse derivative structure makes them suitable for machine learning. Further, we showed how this class of categories is functionally complete for finite number representations, and therefore provides sufficient expressiveness. There remain however a number of opportunities for theoretical and empirical work.

On the empirical side, we plan to use this work combined with data structures and algorithms like that of [23] as the basis for practical machine learning tools. Using these tools, we would like to experimentally verify that models built using semirings like those presented in Section 6 can indeed be used to develop novel model architectures for benchmark datasets.

There also remains a number of theoretical avenues for research. First, we want to generalise our approach to functional completeness to the continuous case, and then to more abstract cases such as polynomial circuits over the Burnside semiring. Second, we want to extend the developments of Section 3 in order to provide a reverse derivative structure for circuits with notions of feedback and delay, such as the stream functions described in [13].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

[1] We discuss why in Example 5.3.

[2] This approach is called 'weight-tying' in neural networks literature.

Appendix A. Graphical proofs of extension theorem

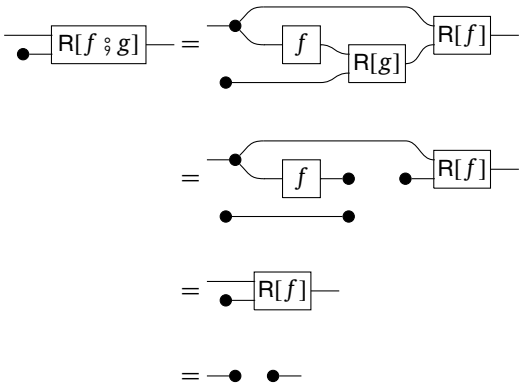
We now prove Theorem 3.1. We split the proof into the following lemmas:

1. ARD.2 is preserved by composition
2. ARD.2 is preserved by tensor product
3. ARD.3/RD.6 is preserved by composition
4. ARD.3/RD.6 is preserved by tensor product
5. ARD.4/RD.7 is preserved by composition
6. ARD.4/RD.7 is preserved by tensor product

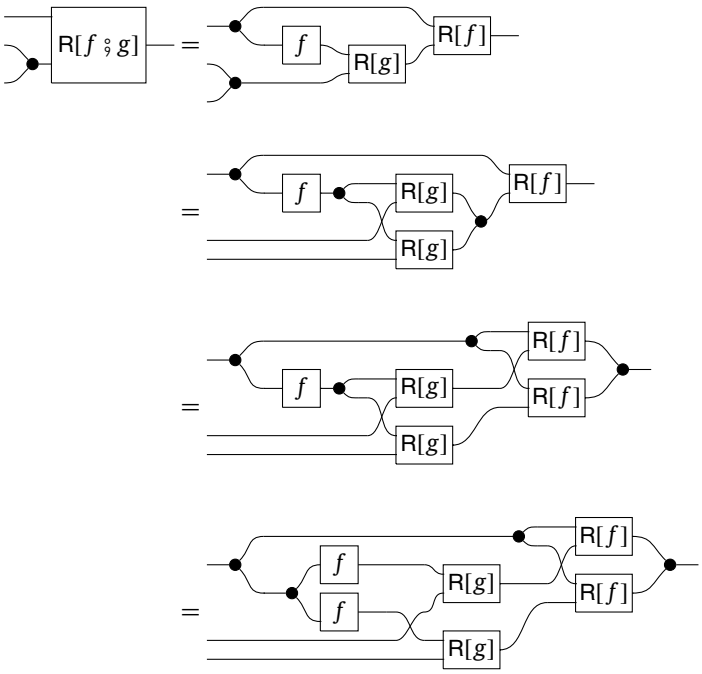
In each case, when we say 'ARD.x is preserved by composition' we mean that if f and g satisfy ARD.x, then so too does $f \fatsemi g$, and likewise for tensor product. Let us now address these lemmas in order.

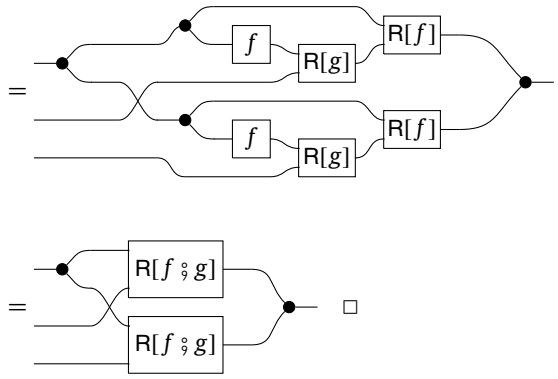
Lemma A.1. *ARD.2 is preserved by composition.*

Proof. Assume that ARD.2 holds for $f : A \to B$ and $g : B \to C$. For the zero case, apply the chain rule and use the hypothesis twice to obtain the result as follows:

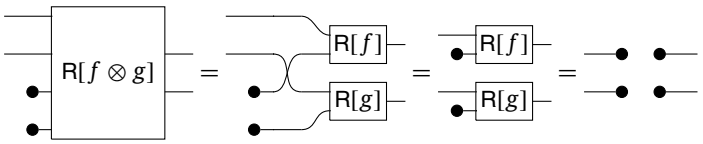


In the additive case we proceed similarly by expanding definitions, applying the hypothesis, and then using associativity and commutativity of to obtain the final result:

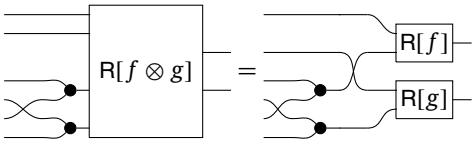


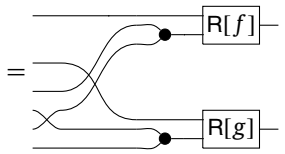


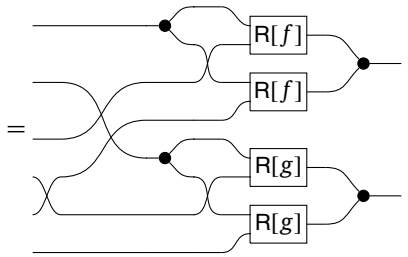
Lemma A.2. *ARD.2 (RD.2) is preserved by tensr product For the zero case,*

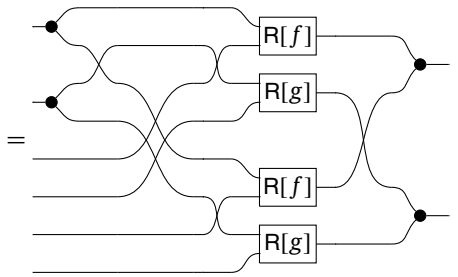


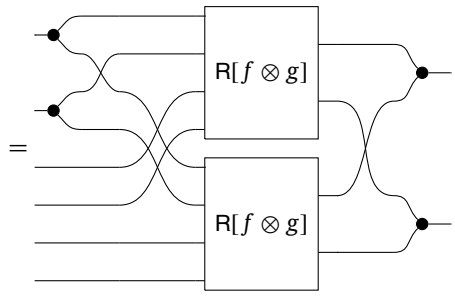
And now in the additive case,





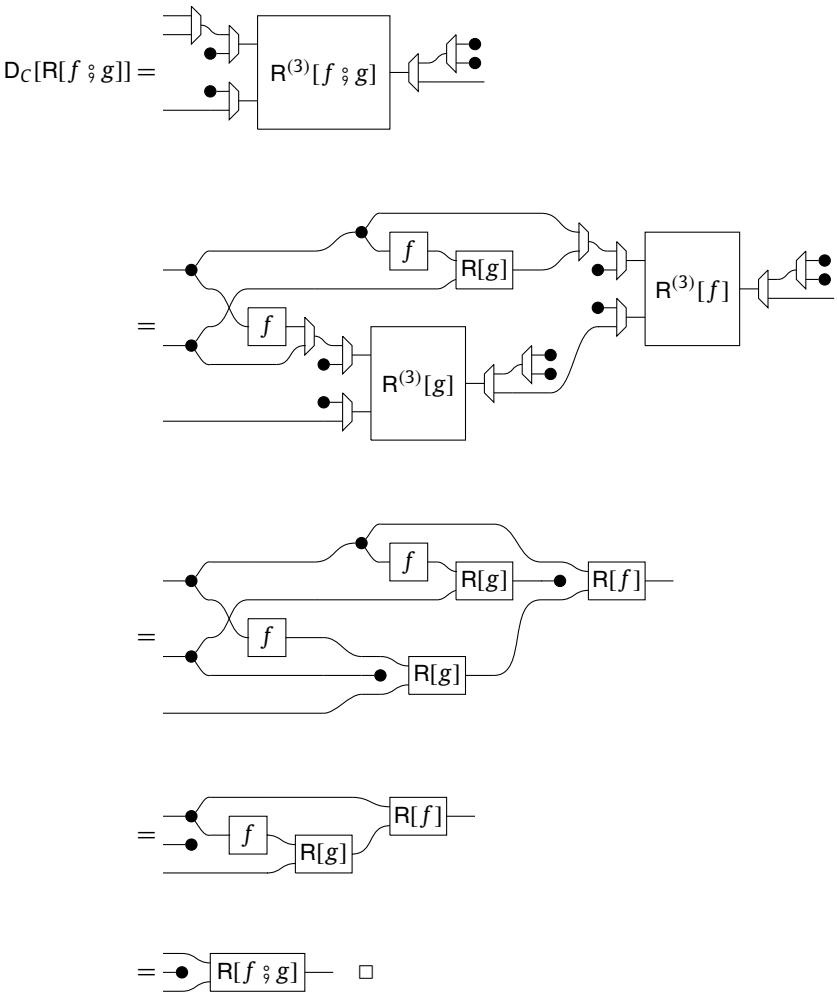






Lemma A.3. *ARD.3 (RD.6) is preserved by composition.*

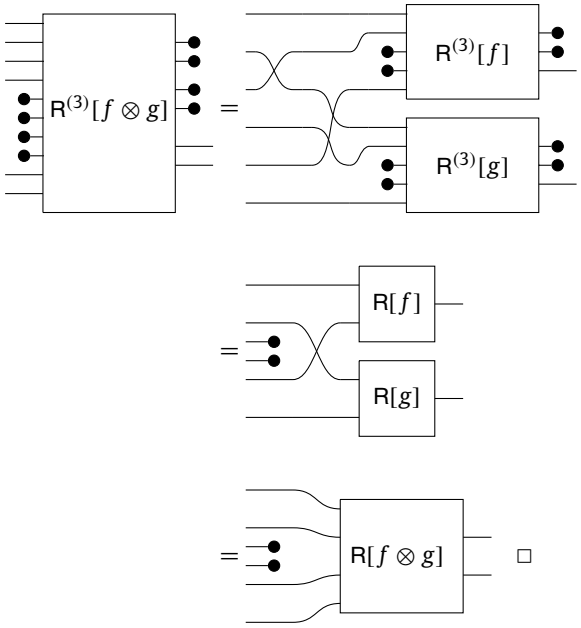
Proof. Assume that ARD.3 holds for $f : A \to B$ and $g : B \to C$. Now calculate:



Note that in the first step, we must expand $\mathsf{R}^{(3)}$ using repeated application of the chain rule- we have omitted much of this tedious calculation. In the second step where we apply the inductive hypothesis, then naturality of $\multimap$ to finally obtain the result.

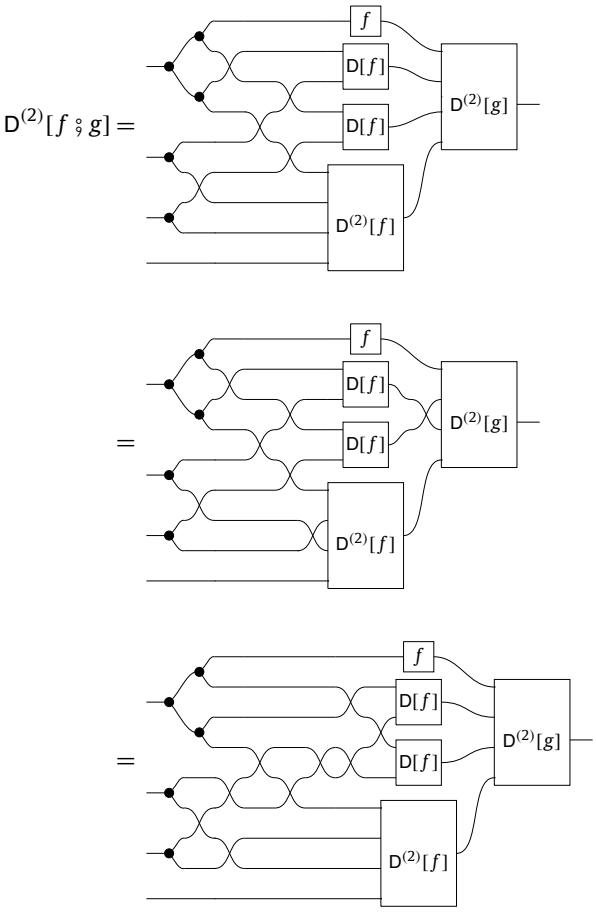
Lemma A.4. *ARD.3 (RD.6) is preserved by tensor product.*

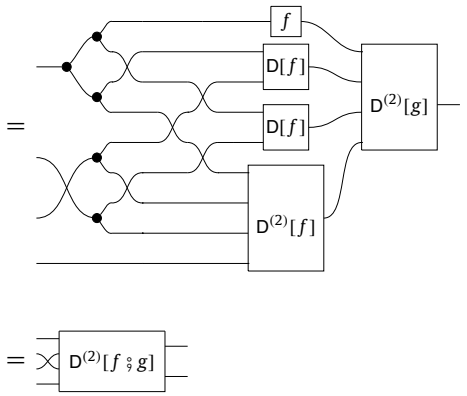
Proof. Assume that ARD.3 holds for f and g. Then it holds for $f \otimes g$ as follows:



Lemma A.5. *ARD.4 (RD.7) is preserved by composition.*

Proof. Assume that ARD.4 holds for $f : A \to B$ and $g : B \to C$. Now we can calculate as follows:



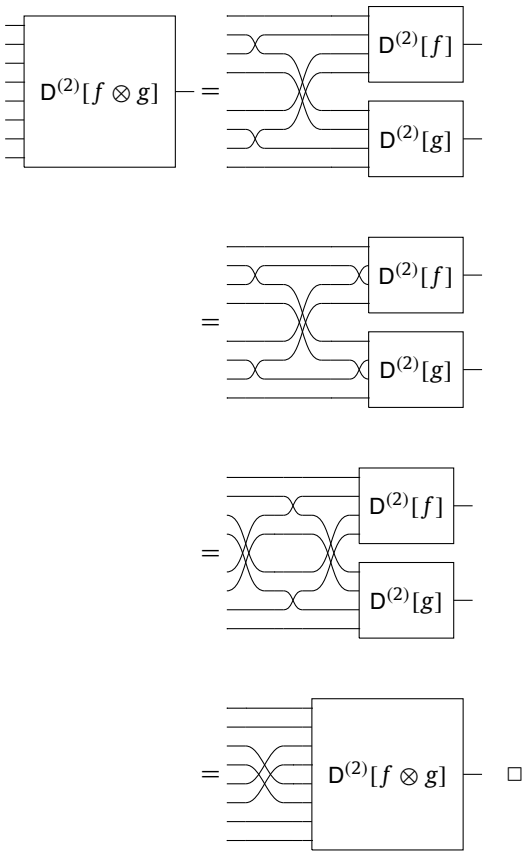


As with the proof for RD.6 we omit a great deal of tedious expansion and calculation from the first step of the proof, which is obtained simply by expanding $\mathsf{D}^{(2)}[f \fatsemi g]$ in terms of R and using naturality of $\multimap$ to simplify the result. In remaining steps, we apply of the assumption that ARD.4 holds for f and g, before finally using naturality of $\multimap$. □

Note that although the above derivation is written in terms of D, each step of the proof treats the D operator merely as a syntactic sugar for its definition in terms of R. Each step of the proof thus uses only axioms of RDCs, rather than the forward differential structure defined in terms of it.

Lemma A.6. *ARD.4 (RD.7) is preserved by tensor product.*

Proof. Assume that ARD.4 holds for $f : A_1 \to B_1$ and $g : A_2 \to B_2$. Then we may calculate as follows, first expanding the definition of D, and then using the inductive hypothesis to obtain the result:



□

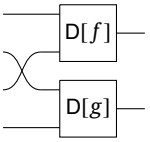
It is now straightforward to prove Theorem 3.1.

Proof. (Proof of Theorem 3.1)
Suppose $\mathscr{C}$ is a cartesian left-additive category presented by generators and relations which is equipped with a (well-defined) R operator such that axioms ARD.1-4 hold for each generator, and that R is defined on tensor and composition of morphisms as in ARD.1. By the lemmas above, composition and tensor product preserve the remaining axioms ARD.2-4, and so $\mathscr{C}$ is an RDC. □

Appendix B. Graphical proofs of forward derivative structure

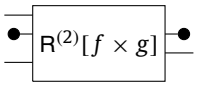
We now give proofs of the two inductive steps for the proof of Proposition 2.9. We first give the inductive step for tensor of morphisms:

Proposition B.1. *Assume that the equality in Definition 2.4 holds for f and g. Then* $\mathsf{D}[f \times g] =$

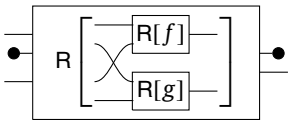


Proof.

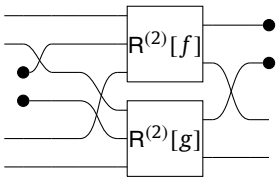
$\mathsf{D}[f \times g] =$



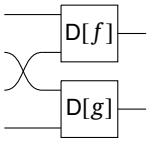
$=$



$=$



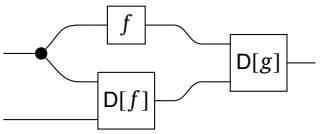
$=$



□

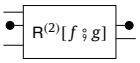
Finally we prove the inductive step for composition.

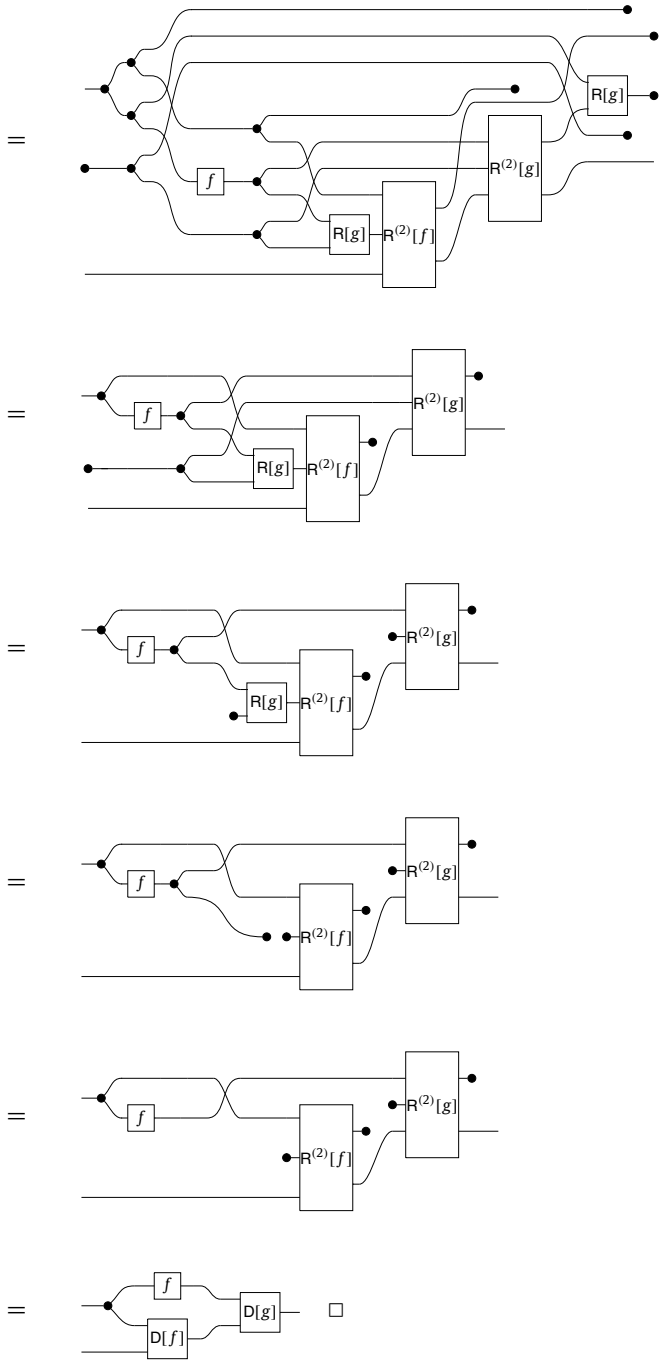
Proposition B.2. *Assume that the equality in Definition 2.4 holds for f and g. Then* $\mathsf{D}[f \fatsemi g] =$



Proof.

$\mathsf{D}[f \fatsemi g] =$



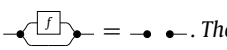


Appendix C. Polynomial circuits over a ring

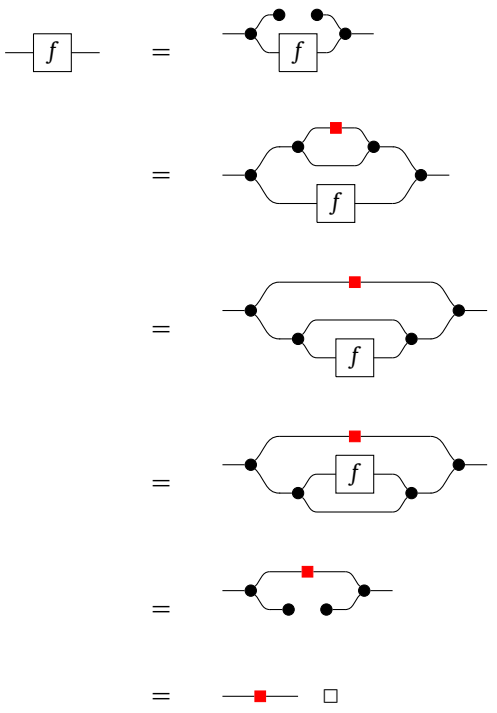
In this section we give several omitted proofs for the category of polynomial circuits with negation $\mathsf{PolyCirc}_S^-$. In particular, we will prove that negations are unique, and that $\mathsf{PolyCirc}_S^-$ has RDC structure.

C.1. Elementary properties of rings

We begin by reframing some well-known elementary properties of rings in the graphical language of polynomial circuits.

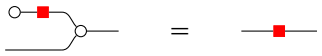
Proposition C.1 *(Uniqueness of negation). Suppose $f : A \to A$ is a morphism such that*  *. Then*  *.*

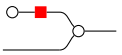
Proof.

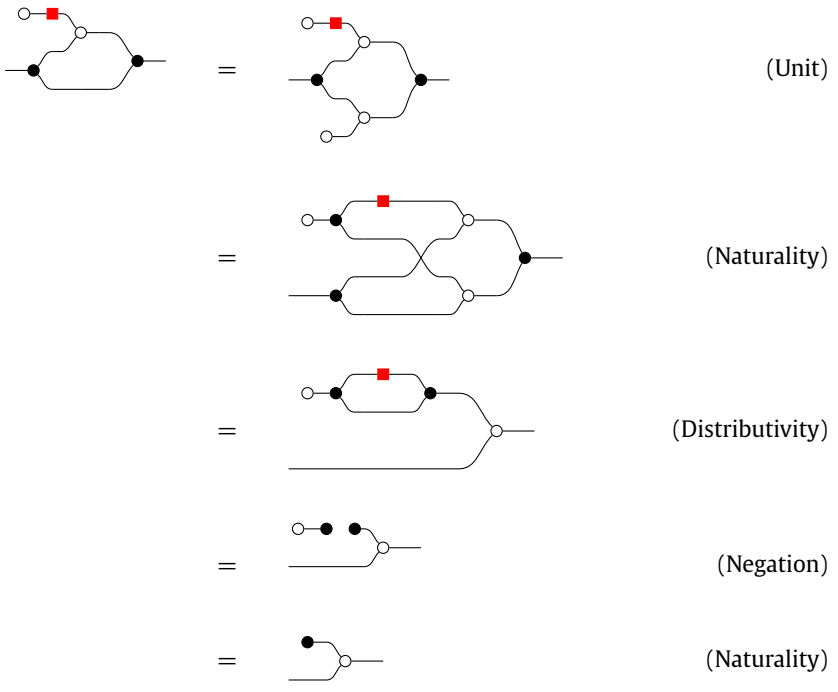


The next proposition amounts to saying $-x = (-1) \cdot x$

Proposition C.2 *(Multiplication by -1 is negation).*

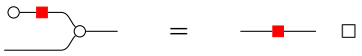


Proof. We first observe that  functions as an additive inverse:

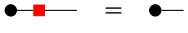



(Annihilation)

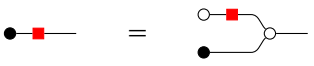
And so by Proposition C.1, we can conclude that


□

The next proposition says that the zero element is its own negation, i.e., that $-0 = 0$.

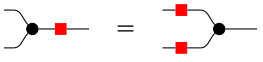
Proposition C.3. 

Proof.

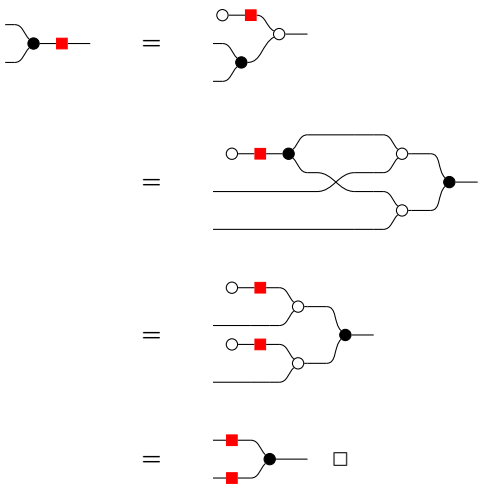

(Proposition C.2)


(Annihilation) □

In addition, the negation of addition is the addition of negations:

Proposition C.4. 

Proof.


□

C.2. Reverse derivatives and negation

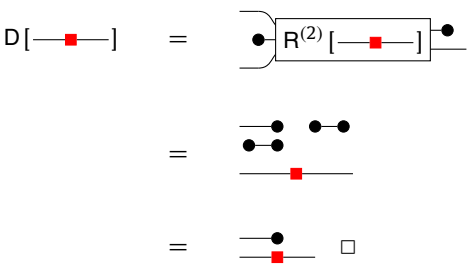
We now prove that $\mathsf{PolyCirc}_S^-$ has reverse derivative structure. Thanks to Theorem 3.1, it will suffice to define a reverse derivative for negations, and show that it is well-defined and respects axioms ARD.2-4.

Definition C.5. $\mathsf{R}[$  $] :=$ 

This choice of reverse derivative leads to the following induced forward derivative.

Proposition C.6. $\mathsf{D}[$  $] =$ 

Proof.

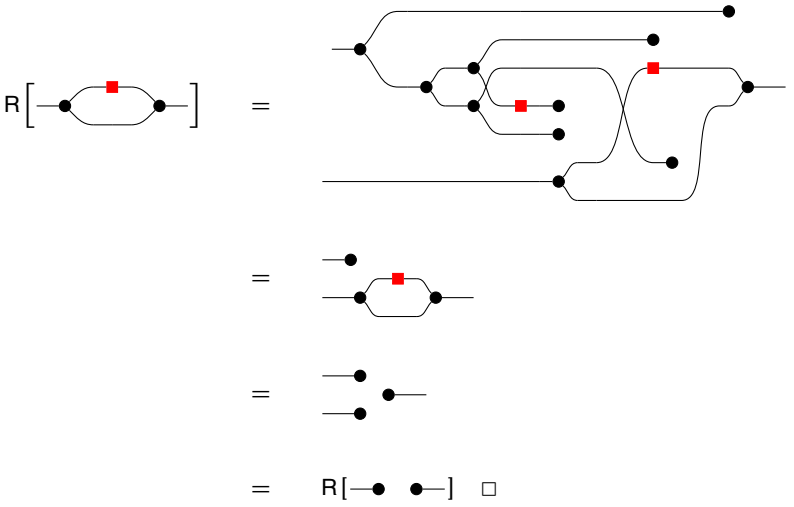

□

Before showing that this definition of R is well-defined and satisfies the axioms ARD.2-4, we first give some elementary properties of rings.

Before proving properties ARDC.2-4, we must check that the definition of R[—■—] is well-defined with respect to the equation —●— = —● ●—.

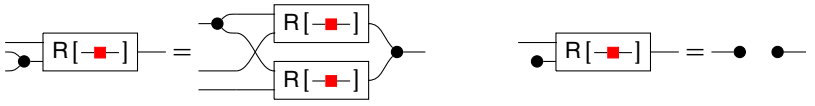
Proposition C.7 *(Well-definedness of —■—).* R[—●—●—] = R[—● ●—].

Proof.

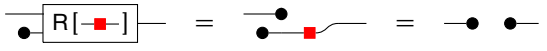


We can now proceed to show that this definition of R satisfies axioms ARDC.2-4. We begin with axiom ARD.2.

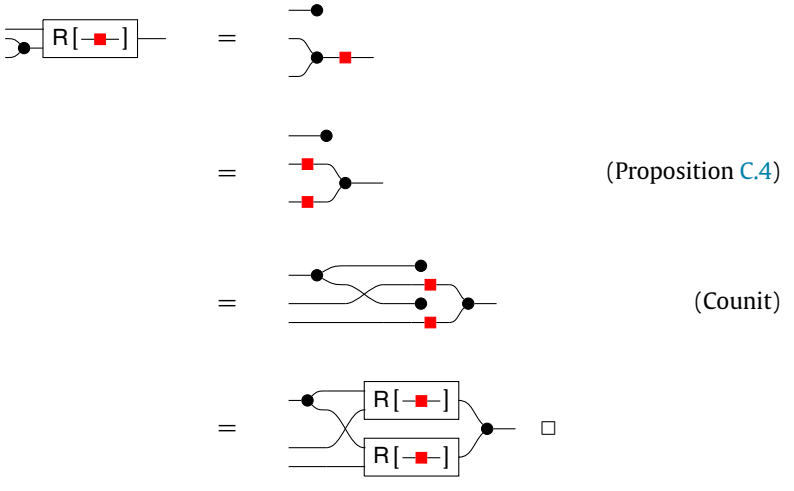
Proposition C.8.



Proof. The latter case follows immediately by Proposition C.3:

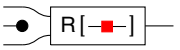


And in the former, we calculate as follows:



We now prove that —■— satisfies axiom ARD.3.

Proposition C.9 *(ARD.3 holds for negation).* $\mathsf{D}_A\,[\mathsf{R}\,[\,-\blacksquare-\,]]$ =

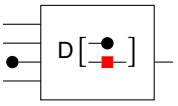


Proof.

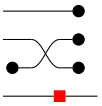
$\mathsf{D}_A\,[\mathsf{R}\,[\,-\blacksquare-\,]]$ = $\mathsf{D}_A\,[$

$]$

=

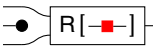


=



=

=

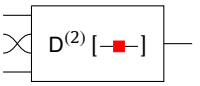


□

Finally, we show that

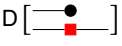
satisfies axiom ARD.4.

Proposition C.10 *(ARD.4 holds for negation).* $\mathsf{D}^{(2)}\,[\,-\blacksquare-\,] =$



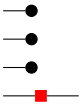
Proof.

$\mathsf{D}^{(2)}\,[\,-\blacksquare-\,]$ = $\mathsf{D}\,[$

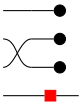


$]$

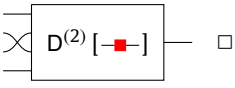
=



=



=



□

Appendix D. Graphical proofs for Cartesian distributive categories

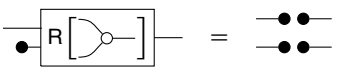
In this section we derive the forward derivative structure of cartesian distributive categories, and show that axioms ARD.2-4 are satisfied.

D.1. RDC axioms for Cartesian distributive structure

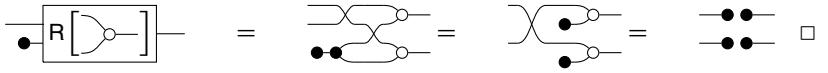
We now verify the axioms of RDCs hold for $\mathsf{R}\,[$

$]$. ARD.2 holds by Propositions D.1 and D.2 below.

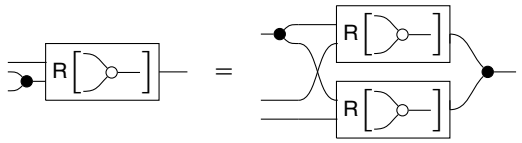
Proposition D.1.



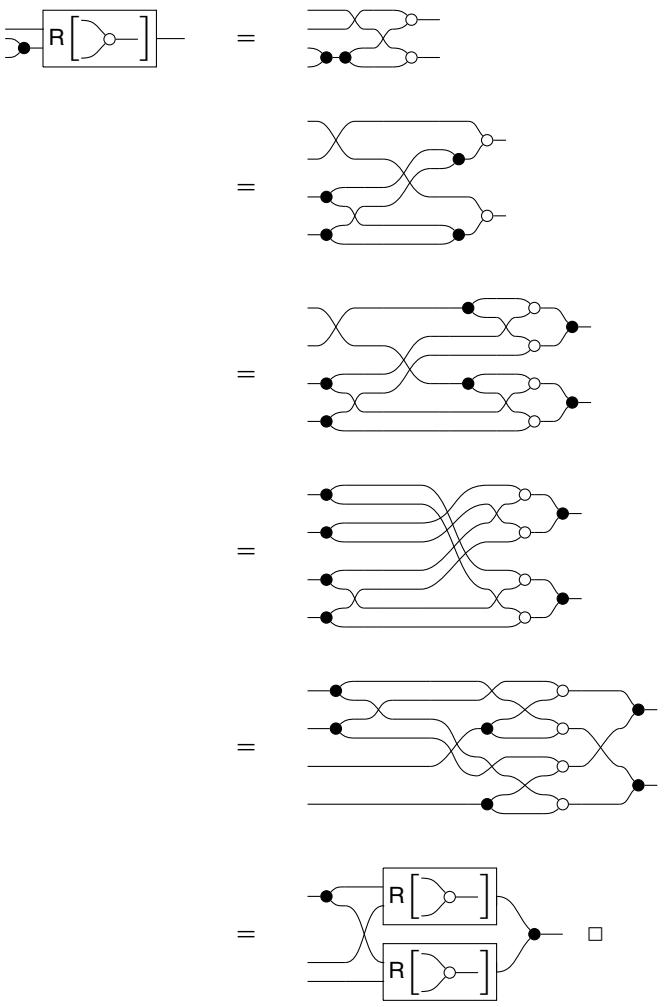
Proof.



Proposition D.2.

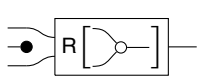


Proof.

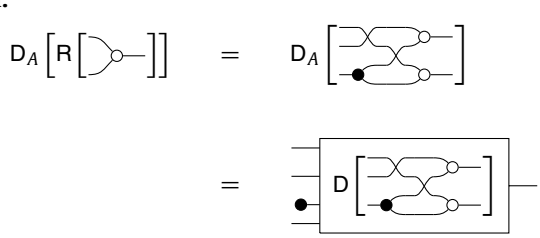


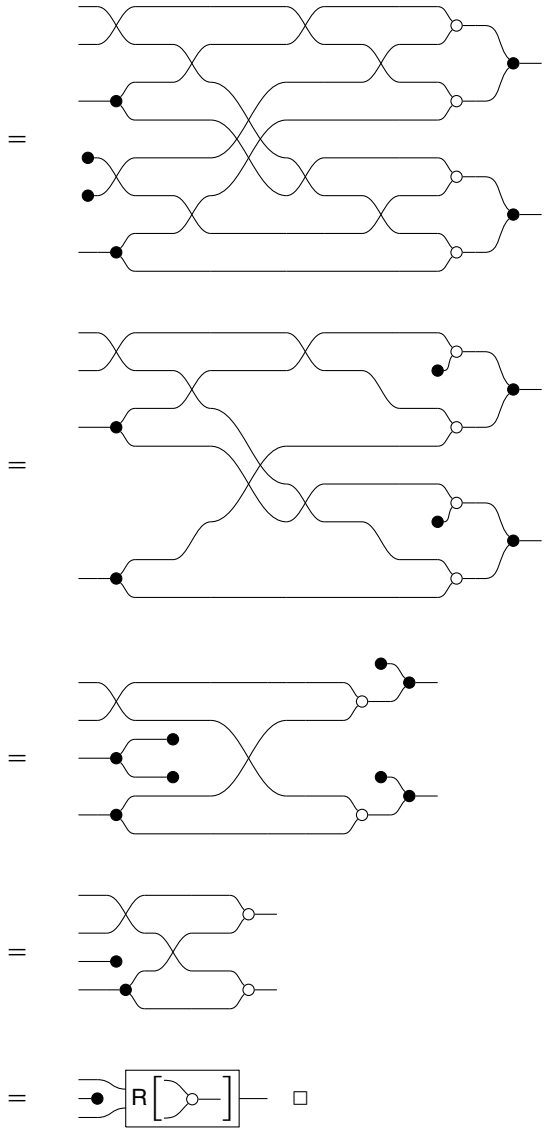
ARD.3 holds by Proposition D.3:

Proposition D.3. $\mathsf{D}_A\left[\mathsf{R}\left[\quad\right]\right]$ =



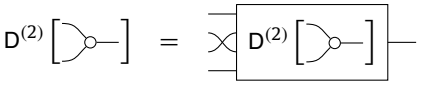
Proof.



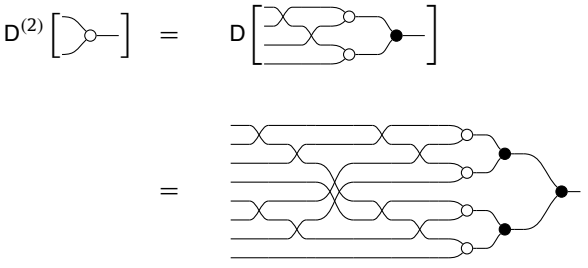


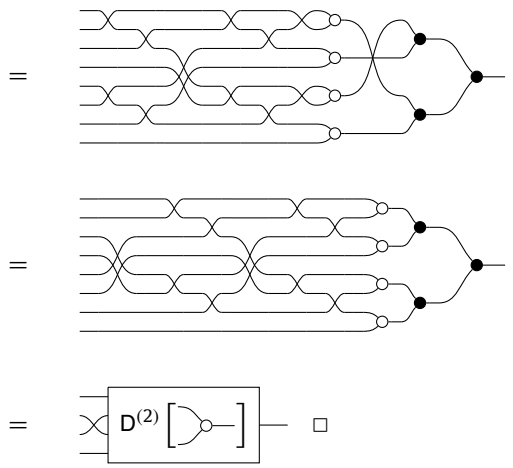
Finally, ARD.4 holds by Proposition D.4:

Proposition D.4.



Proof.

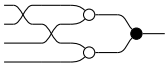




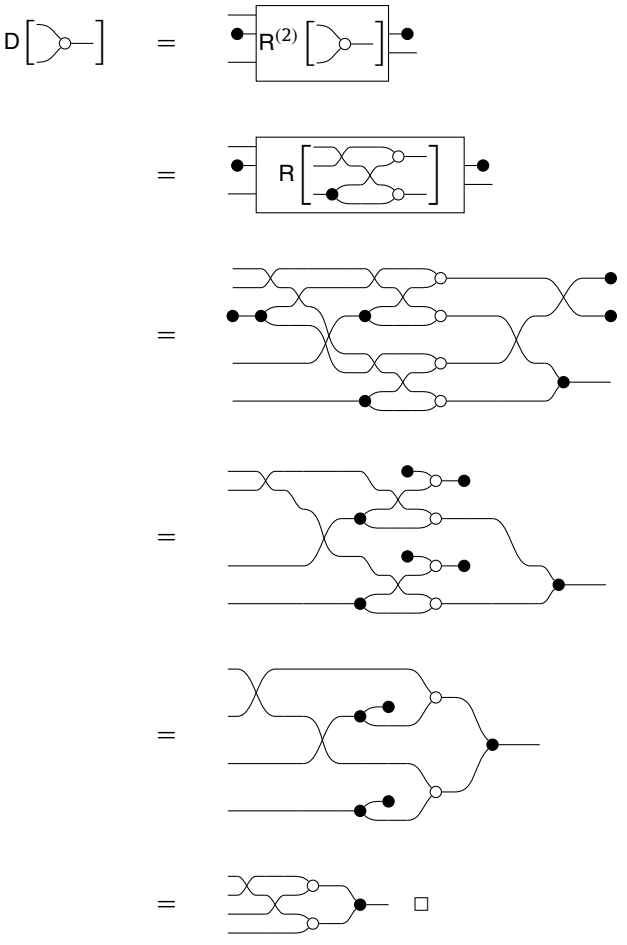
Remark D.5. Note that ARD.4 is the only axiom requiring that is commutative.

D.2. Forward derivative structure

We begin by deriving the values of $\mathsf{D}\left[\right]$ and $\mathsf{D}[\circ\!-]$ for an operator D defined in terms of R as in Definition 2.4.

Proposition D.6 *(Forward Derivative on).* $\mathsf{D}\left[\right] =$ 

Proof.



Proposition D.7 *(Forward Derivative on $\circ\!-$).* $\mathsf{D}[\circ\!-] = \bullet\!-$

Proof. Observe that because $\mathsf{R}[\circ\!-] = \mathsf{R}[\bullet\!-]$, we also have $\mathsf{R}^{(2)}[\circ\!-] = \mathsf{R}^{(2)}[\bullet\!-]$ and therefore $\mathsf{D}[\circ\!-] = \mathsf{D}[\bullet\!-] = \bullet\!-$. □

References

[1] M. Abadi, et al., TensorFlow: large-scale machine learning on heterogeneous systems, https://www.tensorflow.org/, 2015.
[2] F. Alarcón, D. Anderson, Commutative semirings and their lattices of ideals, Houst. J. Math. 20 (1994), https://www.math.uh.edu/~hjm/vol20-4.html.
[3] J.C. Baez, B. Coya, F. Rebro, Props in network theory, http://www.tac.mta.ca/tac/volumes/33/25/33-25abs.html, 2017.
[4] Y. Bengio, N. Léonard, A. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, https://doi.org/10.48550/ARXIV.1308.3432, https://arxiv.org/abs/1308.3432, 2013.
[5] R.F. Blute, J.R.B. Cockett, R.A.G. Seely, Cartesian differential categories, Theory Appl. Categ. 22 (2009), https://emis.univie.ac.at/journals/TAC/volumes/22/23/22-23abs.html.
[6] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobocinski, F. Zanasi, String diagram rewrite theory i: rewriting with Frobenius structure, https://doi.org/10.48550/ARXIV.2012.01847, 2020.
[7] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobocinski, F. Zanasi, String diagram rewrite theory ii: rewriting with symmetric monoidal structure, https://doi.org/10.48550/ARXIV.2104.14686, 2021.
[8] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, F. Zanasi, String diagram rewrite theory iii: confluence with and without Frobenius, https://doi.org/10.48550/ARXIV.2109.06049, 2021.
[9] J. Choi, et al., Accurate and efficient 2-bit quantized neural networks, in: A. Talwalkar, V. Smith, M. Zaharia (Eds.), Proceedings of Machine Learning and Systems, vol. 1, 2019, pp. 348–359, https://proceedings.mlsys.org/paper/2019/file/006f52e9102a8d3be2fe5614f42ba989-Paper.pdf.
[10] R. Cockett, G. Cruttwell, J. Gallagher, J.S.P. Lemay, B. MacAdam, G. Plotkin, D. Pronk, Reverse derivative categories, https://doi.org/10.4230/LIPIcs.CSL.2020.18, 2019.
[11] G.S.H. Cruttwell, B. Gavranović, N. Ghani, P. Wilson, F. Zanasi, Categorical foundations of gradient-based learning, https://doi.org/10.48550/ARXIV.2103.01931, 2021.
[12] P. de Fermat, Letter to frénicle de bessy (1640).
[13] D.R. Ghica, G. Kaye, D. Sprunger, Full abstraction for digital circuits, https://doi.org/10.48550/ARXIV.2201.10456, 2022.
[14] J.S. Golan, Semirings and Their Applications, Springer, Dordrecht, Netherlands, 2010.
[15] J.L. Gustafson, I.T. Yonemoto, Beating floating point at its own game: posit arithmetic, Supercomput. Front. Innov. 4 (2) (2017), https://doi.org/10.14529/jsfi170206.
[16] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366, https://doi.org/10.1016/0893-6080(89)90020-8.
[17] Y. Lafont, Towards an algebraic theory of Boolean circuits, J. Pure Appl. Algebra 184 (2–3) (2003) 257–310, https://doi.org/10.1016/S0022-4049(03)00069-0.
[18] M. Leshno, et al., Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (6) (1993) 861–867, https://doi.org/10.1016/s0893-6080(05)80131-5.
[19] A. Paszke, et al., Pytorch: An Imperative Style, High-Performance Deep Learning Library, Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[20] P. Selinger, A survey of graphical languages for monoidal categories, Lect. Notes Phys. (2010) 289–355, https://doi.org/10.1007/978-3-642-12821-9_4.
[21] C.A. Weibel, The K-Book, Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, Jul 2013.
[22] W. Wernick, Complete sets of logical functions, Trans. Am. Math. Soc. 51 (1) (1942) 117, https://doi.org/10.2307/1989982.
[23] P. Wilson, F. Zanasi, The cost of compositionality: a high-performance implementation of string diagram composition, https://doi.org/10.48550/ARXIV.2105.09257, 2021.
[24] P. Wilson, F. Zanasi, Reverse derivative ascent: a categorical approach to learning boolean circuits, Electron. Proc. Theor. Comput. Sci. 333 (2021) 247–260, https://doi.org/10.4204/eptcs.333.17.
[25] P. Wilson, F. Zanasi, Categories of differentiable polynomial circuits for machine learning, in: Graph Transformation: 15th International Conference, ICGT 2022, Held as Part of STAF 2022, Nantes, France, July 7–8, 2022, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2022, pp. 77–93.
[26] P. Wilson, F. Zanasi, Data-parallel algorithms for string diagrams, 2023.
[27] F. Zanasi, Interacting Hopf algebras: the theory of linear systems, https://doi.org/10.48550/ARXIV.1805.03032, 2018.