



# Graphical Piecewise-Linear Algebra

Guillaume Boisseau<sup>1</sup> and Robin Piedeleu<sup>2</sup>

<sup>1</sup> University of Oxford, Oxford, UK [guillaume.boisseau@cs.ox.ac.uk](mailto:guillaume.boisseau@cs.ox.ac.uk)

<sup>2</sup> University College London, London, UK [r.piedeleu@ucl.ac.uk](mailto:r.piedeleu@ucl.ac.uk)

**Abstract.** Graphical (Linear) Algebra is a family of diagrammatic languages allowing to reason about different kinds of subsets of vector spaces compositionally. It has been used to model various application domains, from signal-flow graphs to Petri nets and electrical circuits. In this paper, we introduce to the family its most expressive member to date: Graphical Piecewise-Linear Algebra, a new language to specify piecewise-linear subsets of vector spaces.

Like the previous members of the family, it comes with a complete axiomatisation, which means it can be used to reason about the corresponding semantic domain purely equationally, forgetting the set-theoretic interpretation. We show completeness using a single axiom on top of Graphical Polyhedral Algebra, and show that this extension is the smallest that can capture a variety of relevant constructs.

Finally, we showcase its use by modelling the behaviour of stateless electronic circuits of ideal elements, a domain that had remained outside the remit of previous diagrammatic languages.

**Keywords:** string diagrams · piecewise-linear · prop · axiomatisation

## 1 Introduction

Functional thinking underpins most scientific models. Nature, however, does not distinguish inputs and outputs—physical systems are governed by laws that merely express *relations* between their observable variables. While influential scientists, like the famous control theorist J. Willems, have pointed out the blind spots of functional thinking [11], it has remained the dominant paradigm in science and engineering. Arguably, our mathematical practice, especially the foundational emphasis on sets and functions, and the limitations of standard algebraic syntax, are partially to blame for the persistence of this status quo. But there are also alternative approaches, that take relations seriously as the primitive building blocks of our mathematical models. Category theory in particular is agnostic about what constitutes a morphism and can accommodate relations as easily as functions.

Relations, with their usual composition and the cartesian product of sets, form a monoidal category—a category in which morphisms can be composed in two different ways. As a result, they admit a natural two-dimensional syntax of *string diagrams*. This notation has several advantages when it comes to

reasoning about open and interconnected systems [1]: string diagrams naturally keep track of structural properties, such as interconnectivity; they factor out irrelevant topological information that standard algebraic syntax needs to keep explicit; variable-sharing—the relational form of composition for systems—is depicted simply by wiring different components together.

As a result, a wealth of recent developments in computer science and beyond have adopted relations and their diagrammatic notation as a unifying language to reason about a broad range of systems, from electrical circuits to Petri nets [2,6,5]. Many of these follow the same methodology. 1) Given a class of systems, find a set of diagrammatic generators from which any system can be specified, using the two available forms of composition. 2) Interpret each of them as a relation between the observable variables of the system that they describe. This defines a structure-preserving mapping—a monoidal functor—from the diagrammatic syntax to the semantics, from the two-dimensional representation of a system to its behaviour. 3) Finally, identify a convenient set of equations between diagrams, from which any semantic equality between the behaviour of the corresponding systems may be derived.

Graphical linear algebra (GLA) is a paradigmatic example of this approach. It provides a diagrammatic syntax to reason compositionally about different types of linear dynamical systems (including for instance traditional *signal flow graphs*) and prove their behavioural equivalence purely diagrammatically. The syntax of GLA is generated by the following primitive components:

$$\text{---}\bigcap\text{---} \mid \text{---}\bullet\text{---} \mid \text{---}\bigcup\text{---} \mid \bullet\text{---} \mid \text{---}\bigcap\text{---} \mid \text{---}\bigcirc\text{---} \mid \text{---}\bigcup\text{---} \mid \bigcirc\text{---} \mid \text{---}\boxed{x}\text{---} \quad (x \in \mathbb{K})$$

As relations, the black nodes force all of their ports to share the same value; the white nodes constrain their left ports and the right ports to sum to the same value (or to zero when there are no left/right ports); the final generator, parameterised by an element of the chosen field  $\mathbb{K}$ , behaves as an amplifier: its right value is  $x$  times the left value. Following point 3) of the methodology sketched above, GLA enjoys a sound and complete equational theory for the specified semantics, called the theory of *Interacting Hopf Algebras* (IH). In summary, string diagrams with  $n$  ports on the left and  $m$  ports on the right, quotiented by the axioms of IH, are precisely linear relations, i.e., linear subspaces of  $\mathbb{K}^n \times \mathbb{K}^m$ .

GLA was the starting point of different extensions, two of which play a prominent role in this paper. First, Graphical Affine Algebra, which adds to the syntax a generator  $\text{---}1\text{---}$  for the constant 1. This allows it to express *affine* relations, i.e. affine subspaces of  $\mathbb{K}^n \times \mathbb{K}^m$ . A corresponding complete equational theory was presented in [6]. Then, Graphical Polyhedral Algebra (GPA), which assumes that  $\mathbb{K}$  is an ordered field and adds a generator  $\text{---}\geq\text{---}$  for this order. The resulting graphical calculus can express all polyhedral relations, i.e., polyhedra<sup>3</sup> in  $\mathbb{K}^n \times \mathbb{K}^m$ , and also comes with its own complete axiomatisation.

In this paper, we define the most expressive member of the GLA family tree to date: Graphical Piecewise-Linear Algebra (GPLA) is a hybrid of symbolic and

<sup>3</sup> For the case of  $\mathbb{R}$ , these include the usual polytopes, which are bounded subsets of  $\mathbb{R}^n \times \mathbb{R}^m$ , as well as proper polyhedra, which may have unbounded faces.

diagrammatic syntax for piecewise-linear (pl) relations—finite unions of polyhedra in  $\mathbb{K}^n \times \mathbb{K}^m$ —and a corresponding complete equational theory. We argue below that the proposed language strikes a convincing balance between structure and expressiveness. It is a simple extension of GPA [4], yet for  $\mathbb{K} = \mathbb{R}$ , it is sufficiently powerful to approximate any submanifold of  $\mathbb{R}^n$  arbitrarily closely.

Furthermore, this extension completes a research program initiated in parallel with the birth of GLA [2,6,3]: its chief purpose was to give the informal graphical notation for electrical circuits a formal, compositional interpretation, with a corresponding equational theory.

Until now however, the category-theoretic setting could only accommodate components with a linear (more precisely, affine) behaviour, such as resistors, inductors, capacitors, voltage and current sources. GPLA finally makes it possible to reason equationally about electronic components, such as ideal diodes and transistors. Even when the idealised physical behaviour of these components is not necessarily piecewise-linear, GPLA is theoretically expressive enough to approximate it as closely as necessary. Indeed, piecewise-linear approximations of transistor behaviour have been proposed to bypass the unavoidable abstraction leaks of purely digital circuits [9]. In this context, GPLA can serve as a form of abstract interpretation for electronic circuits, with adjustable precision to allow for the intended semantics to be as physically realistic as desired. Of course, in practice, working with large diagrams can be prohibitive. But this is a limitation shared by all members of the Graphical Algebra family, and developing convenient tools and techniques for diagrammatic reasoning is an active research area. Our main thrust is that piecewise-linearity provides the appropriate level of structure, where general relations are too flexible to come with a useful equational theory, and linear relations are too rigid to accommodate diodes and other electronic components.

Finally, a remark about syntax. While it is possible to make the language purely diagrammatic, we found that what one gains in purity one loses in complexity. Ultimately, the hybrid syntax of union and diagrams is more convenient to manipulate and intuitive to read. In fact, this is not the first time that sums of diagrams appear in the literature [8]. Nevertheless, one of our central technical contributions is the rigorous definition of a syntax blending diagrams and binary joins, and the corresponding notion of equational theory.

*Outline.* In Section 2 we recall the necessary mathematical background, the fundamentals of diagrammatic syntax, and the language of Graphical Polyhedral Algebra (GPA). In Section 3, we extend the diagrammatic syntax with unions and define the notion of symmetric monoidal semi-lattice theory. From there, in Section 4, we extend GPA with unions, to capture piecewise-linear relations, and give this new language a theory that we prove is complete (Theorem 2). This is our main technical contribution. In Section 5, we explore alternative languages for piecewise-linear relations, and show that they are all equally expressive. Finally, in Section 6, we extend the compositional re-interpretation of electrical circuits from [3] to include electronic components, namely diodes and transistors.

## 2 Preliminaries

Informally, our starting point is a simple diagrammatic language of circuits built from the following generators:

$$\begin{array}{cccccccccccc} \bullet & | & \bullet & | & \cup & \bullet & | & \bullet & | & \cap & \circ & | & \circ & | & \cup & \circ & | & \vdash & | & \geq & | & x \\ \text{---} & & \text{---} & & \text{---} & & \text{---} & & \text{---} & & \text{---} & & \text{---} & & \text{---} & & \text{---} & & \text{---} & & \text{---} \end{array} \quad (x \in \mathbb{K}) \quad (1)$$

We will explain how these basic components can be wired together and give them a formal interpretation.

## 2.1 Props and Symmetric Monoidal Theories


The mathematical backbone of our approach is the notion of product and permutations category ( $\text{prop}$ ), a structure which generalises standard algebraic theories [7]. Formally, a *prop* is a strict symmetric monoidal category (SMC) whose objects are the natural numbers and where the monoidal product  $\oplus$  on objects is given by addition. Equivalently, it is a strict SMC whose objects are all monoidal products of a single generating object. *Prop morphisms* are strict symmetric monoidal functors that act as the identity on objects.

Following an established methodology, we will define two props: **Syn** and **Sem**, for the syntax and semantics respectively. To guarantee a compositional interpretation, we require  $\llbracket \cdot \rrbracket : \mathbf{Syn} \rightarrow \mathbf{Sem}$ , the mapping of terms to their intended semantics, to be a prop morphism.

Typically, the syntactic prop **Syn** is freely generated from a *monoidal signature*  $\Sigma$ , i.e. a set of arrows  $g : m \rightarrow n$ . In this case, we use the notation  $\mathbf{P}\Sigma$  and **Syn** interchangeably. Morphisms of  $\mathbf{P}\Sigma$  are terms of an  $(\mathbb{N}, \mathbb{N})$ -sorted syntax, whose constants are elements of  $\Sigma$  and whose operations are the usual composition  $(-); (-) : \mathbf{Syn}(n, m) \times \mathbf{Syn}(m, l) \rightarrow \mathbf{Syn}(n, l)$  and the monoidal product  $(-) \oplus (-) : \mathbf{Syn}(n_1, m_1) \times \mathbf{Syn}(n_2, m_2) \rightarrow \mathbf{Syn}(n_1 + n_2, m_1 + m_2)$ , quotiented by the laws of SMCs. But this quotient is cumbersome and unintuitive to work with.

This is why we will prefer a different representation. With their two forms of composition, monoidal categories admit a natural two-dimensional graphical notation of *string diagrams*. The idea is that an arrow  $c : n \rightarrow m$  of  $\mathbf{P}\Sigma$  is better represented as a box with  $n$  ordered wires on the left and  $m$  on the right. We can compose these diagrams in two different ways: horizontally, by connecting the right wires of one diagram to the left wires of another, and vertically by juxtaposing two diagrams:

$$c; d = \begin{array}{c} n \\ \text{---} \end{array} \boxed{c} \begin{array}{c} m \\ \text{---} \end{array} \boxed{d} \begin{array}{c} l \\ \text{---} \end{array} \quad d_1 \oplus d_2 = \begin{array}{c} n_1 \\ \text{---} \end{array} \boxed{d_1} \begin{array}{c} m_1 \\ \text{---} \end{array} \quad \begin{array}{c} n_2 \\ \text{---} \end{array} \boxed{d_2} \begin{array}{c} m_2 \\ \text{---} \end{array}$$

where the labelled wire  $-^n$  is syntactic sugar for a stack of  $n$  wires. The identity  $id_1 : 1 \rightarrow 1$  is denoted as a plain wire  $-$ , the unit for  $\oplus$ ,  $id_0 : 0 \rightarrow 0$ , as the empty diagram , and when the category is symmetric, the symmetry  $\sigma_{1,1} : 2 \rightarrow 2$  is

denoted as a wire crossing  $\bowtie$ . With this representation the laws of SMCs become diagrammatic tautologies.

Once we have defined  $\llbracket \cdot \rrbracket : \mathbf{Syn} \rightarrow \mathbf{Sem}$ , it is natural to look for equations to reason about semantic equality directly on the diagrams themselves. Given a set of equations  $E$ , i.e., a set containing pairs of arrows of the same type, we write  $\stackrel{E}{=}$  for the smallest congruence wrt the two composition operations  $;$  and  $\oplus$ . We say that  $\stackrel{E}{=}$  is *sound* if  $c \stackrel{E}{=} d$  implies  $\llbracket c \rrbracket = \llbracket d \rrbracket$ . It is moreover *complete* when the converse implication holds. We call a pair  $(\Sigma, E)$  a *symmetric monoidal theory* (SMT) and we can form the prop  $\mathbf{P}\Sigma_{/E}$  obtained by quotienting  $\mathbf{P}\Sigma$  by  $\stackrel{E}{=}$ . There is then a prop morphism  $q : \mathbf{P}\Sigma \rightarrow \mathbf{P}\Sigma_{/E}$  witnessing this quotient.

We may also wonder what the expressive power of our diagrammatic language is. In terms of props we look to characterise precisely the image  $\text{Im}(\llbracket \cdot \rrbracket)$  of the syntax via  $\llbracket \cdot \rrbracket$ .

The situation for a sound and complete SMT is summarised in the commutative diagram below right.

Soundness simply means that  $\llbracket \cdot \rrbracket$  factors as  $s \circ q$  through  $\mathbf{P}\Sigma_{/E}$  and completeness means that  $s$  is a faithful prop morphism.

$$\begin{array}{ccc} \mathbf{P}\Sigma_{/E} & \xrightarrow{\cong} & \text{Im}(\llbracket \cdot \rrbracket) \\ q \uparrow & \searrow s & \downarrow i \\ \mathbf{Syn} = \mathbf{P}\Sigma & \xrightarrow{\llbracket \cdot \rrbracket} & \mathbf{Sem} \end{array}$$

Typically, our semantic prop  $\mathbf{Sem}$  will be (a subcategory of) the category of sets and relations.

**Definition 1.** Let  $\mathbb{K}$  be a field.  $\text{Rel}_{\mathbb{K}}$  is the prop

- whose arrows  $n \rightarrow m$  are relations  $R \subseteq \mathbb{K}^n \times \mathbb{K}^m$ ,
- with composition given by  $R; S = \{(x, z) \mid \exists y. (x, y) \in R \wedge (y, z) \in S\}$ , for  $R : n \rightarrow m$ ,  $S : m \rightarrow l$ , and identity  $n \rightarrow n$  the diagonal  $\{(x, x) \mid x \in \mathbb{K}^n\}$ ,
- monoidal product given by

$$R_1 \oplus R_2 = \left\{ \left( \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right) \mid (x_1, y_1) \in R_1 \wedge (x_2, y_2) \in R_2 \right\}$$

for  $R_1 : n_1 \rightarrow m_1$  and  $R_2 : n_2 \rightarrow m_2$ ,

- symmetry  $n+m \rightarrow m+n$ , the relation  $\left\{ \left( \begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} y \\ x \end{pmatrix} \right) \mid (x, y) \in \mathbb{K}^n \times \mathbb{K}^m \right\}$ .

## 2.2 Ordered Props and Symmetric Monoidal Inequality Theories

Our semantic prop— $\text{Rel}_{\mathbb{K}}$ —carries additional structure that we wish to lift to the syntax: as subsets of  $\mathbb{K}^n \times \mathbb{K}^m$ , relations  $n \rightarrow m$  can be ordered by inclusion. The corresponding structure is that of an *ordered prop*, a prop enriched over the category of posets, whose composition and monoidal product are monotone maps.

If props can be presented by SMTs, ordered props can be presented by *symmetric monoidal inequality theories* (SMIT). Formally, the data of a SMIT is



- $\text{IH}_{\geq}^+$  provides an axiomatisation of polyhedral relations [4, Corollary 25]; it can be found in the first four blocks of Fig. 1.

*Example 1 (Duality).* Two diagrams play a special role in this paper: the half turns  $\bullet\!\!\!\curvearrowright$  and  $\!\!\!\curvearrowleft\!\!\!\bullet$ , called cup and cap, respectively. Using these and  $\times$ , we can build cups and caps for any number  $n$  of wires:  $\bullet\!\!\!\curvearrowright^n$  and  $\!\!\!\curvearrowleft^n\!\!\!\bullet$ .

They allow us to associate a dual  $d^{op} : n \rightarrow m$  to any diagram  $d : m \rightarrow n$  by turning its left ports into right ports and vice-versa:

$$n \text{---} \boxed{d^{op}} \text{---} m = \begin{array}{c} n \\ \text{---} \end{array} \boxed{d} \begin{array}{c} \bullet\!\!\!\bullet \\ \text{---} \end{array} m \quad (2)$$

Correspondingly,  $\llbracket d^{op} \rrbracket$  is the *opposite* relation, i.e.  $\llbracket d^{op} \rrbracket = \{(y, x) \mid (x, y) \in \llbracket d \rrbracket\}$ . We will use of a suggestive mirror notation to denote the dual of a given generator:  $\text{---} \boxed{r} \text{---} := (\text{---} \boxed{r} \text{---})^{op}$ ,  $\text{---} \vdash := (\text{---} \vdash)^{op}$  and  $\text{---} \leq \text{---} := (\text{---} \leq \text{---})^{op}$ .

### 3 Symmetric Monoidal Semi-Lattice Theories

There are several routes to describe piecewise-linear subsets of  $\mathbb{K}^n$ . In this paper we choose to equip our syntax with a primitive operation of join, in order to describe piecewise-linear sets as (finite) unions of polyhedra. In the same way that we moved from simple props to ordered props in Section 2.2, we now move to the setting of semi-lattice-enriched props.

A  $\cup$ -prop is a prop enriched over the monoidal category of semi-lattices – partially-ordered sets with least upper bounds for any finite subset – and join-preserving maps, with the Cartesian product as monoidal product. In other words a  $\cup$ -prop is a prop whose homsets are semi-lattices, with composition and monoidal product themselves join-preserving. The paradigmatic example is  $\text{Rel}_{\mathbb{K}}$  which is a  $\cup$ -prop with the union of relations as join.

As we would like to incorporate binary joins into our syntax, we need a new description of the free  $\cup$ -prop  $\text{P}_{\cup}\Sigma$  over a given signature  $\Sigma$ .

- The arrows  $n \rightarrow m$  of  $\text{P}_{\cup}\Sigma$  are finite sets of arrows  $n \rightarrow m$  of  $\text{P}\Sigma$ . We use capital letters  $C, D \dots$  to range over them. We will also abuse notation slightly, using  $c, d \dots$  to refer to singletons  $\{c\}, \{d\} \dots$  and writing  $d_1 \cup \dots \cup d_k$  for the set  $\{d_1, \dots, d_k\}$ . The set can be empty, yielding the bottom of the semi-lattice.
- The composition of  $C : n \rightarrow m$  and  $D : m \rightarrow l$  is given by  $C ; D = \{c ; d \mid c \in C, d \in D\}$  where  $c ; d$  denotes composition in  $\text{P}\Sigma$ . The identity over  $n$  is the singleton  $\{id_n\}$ .
- The monoidal product of  $D_1 : n_1 \rightarrow m_1$  and  $D_2 : n_2 \rightarrow m_2$  is given by  $D_1 \oplus D_2 = \{d_1 \oplus d_2 \mid d_1 \in D_1, d_2 \in D_2\}$  where  $d_1 \oplus d_2$  is the monoidal product in  $\text{P}_{\cup}\Sigma$ .
- For the enrichment, each homset  $\text{P}_{\cup}\Sigma(n, m)$  is a semi-lattice with union as join. By definition, composition and monoidal product distribute over union and define join-preserving maps  $(-); (-) : \text{P}_{\cup}\Sigma(n, m) \times \text{P}_{\cup}\Sigma(m, l) \rightarrow \text{P}_{\cup}\Sigma(n, l)$  and  $(-) \oplus (-) : \text{P}_{\cup}\Sigma(n_1, m_1) \times \text{P}_{\cup}\Sigma(n_2, m_2) \rightarrow \text{P}_{\cup}\Sigma(n_1 + n_2, m_1 + m_2)$

We now define a corresponding notion of theory for  $\cup$ -props. A *symmetric monoidal (semi-)lattice theory* (SMLT) is the data of a signature  $\Sigma$  and a set  $E$  of equations: formally the latter is a set of pairs  $(C, D)$  of arrows  $C, D : n \rightarrow m$  from  $\mathbf{P}_{\cup}\Sigma$ . We will write the elements of  $E$  as equations of the form  $\bigcup_{c \in C} c = \bigcup_{d \in D} d$ . We now explain how to define a  $\cup$ -prop  $\mathbf{P}_{\cup}\Sigma_{/E}$  from the data of an SMLT  $(\Sigma, E)$ . As for SMTs, we can build the smallest congruence  $\stackrel{E}{=}$  wrt to  $;$  and  $\oplus$ , which equates the pairs in  $E$ . Then define  $\mathbf{P}_{\cup}\Sigma_{/E}$  to be the quotient of  $\mathbf{P}_{\cup}\Sigma$  by  $\stackrel{E}{=}$ . That this is a well-defined  $\cup$ -prop follows again from the distributivity of the composition and monoidal product over unions.

Note that the semi-lattice structure allows us to define an order over the homsets of any  $\cup$ -prop, making it into an ordered prop: we write  $C \subseteq D$  as a shorthand for  $C \cup D = D$ . We will also use  $C \stackrel{E}{\subseteq} D$  for  $C \cup D \stackrel{E}{=} D$  in  $\mathbf{P}_{\cup}\Sigma_{/E}$ . (We prefer this notation to avoid the confusion with the order  $\geq$  on  $\mathbb{K}$  itself.)

*Remark 1 (Reasoning in  $\cup$ -props).* The reader familiar with string diagrams and equational reasoning might be surprised by certain features of derivations that combine diagrammatic and traditional syntax (joins, in this case). We would like to clarify one particular point. When we want to use an equality of the form  $d = d_1 \cup d_2$  inside a term of the form  $c_1 \cup c_2 \cup c$ , we need to identify a linear context  $C[-]$  (i.e. the hole occurs exactly once in  $C$ ) common to  $c_1$  and  $c_2$  such that  $c_1 = C[d_1]$  and  $c_2 = C[d_2]$ . Then we are allowed to use the fact that  $C[d] = C[d_1] \cup C[d_2]$  to conclude that  $c_1 \cup c_2 \cup c = C[d] \cup c$ . An example of this form of reasoning can be found in the proof of Lemma 2, which we reproduce here: we apply the equality  $\bullet \stackrel{(total)}{=} \circ \text{--} \boxed{\leq} \text{--} \cup \circ \text{--} \boxed{\geq} \text{--}$  in

$$\begin{array}{c} \text{---} \bullet \text{---} \boxed{D} \text{---} \boxed{H} \text{---} \boxed{\geq} \text{---} \circ \end{array} \cup \begin{array}{c} \text{---} \bullet \text{---} \boxed{D} \text{---} \boxed{H} \text{---} \boxed{\leq} \text{---} \circ \end{array} = \begin{array}{c} \text{---} \bullet \text{---} \boxed{D} \text{---} \boxed{H} \text{---} (\text{---} \boxed{\geq} \text{---} \circ \cup \text{---} \boxed{\leq} \text{---} \circ) \end{array} \stackrel{(total)}{=} \begin{array}{c} \text{---} \bullet \text{---} \boxed{D} \text{---} \boxed{H} \text{---} \bullet \end{array}$$

Note that, to clarify the common context to the reader, we will often use the intermediate notation  $C[d_1 \cup d_2]$ , as we did in the first step above.

## 4 The Theory of Piecewise-Linear Relations

### 4.1 Syntax and Semantics

For piecewise-linear relations we retain the same signature  $\Sigma_{\geq}^+$  and consider  $\mathbf{P}_{\cup}(\Sigma_{\geq}^+)$ , the free  $\cup$ -prop over it. As we saw, its morphisms are nonempty finite sets of diagrams of  $\mathbf{P}\Sigma_{\geq}^+$ . This is our syntax.

On the semantic side, we now need to extend the functor  $\llbracket \cdot \rrbracket$  to have  $\mathbf{P}\Sigma_{\geq}^+$  as domain, retaining  $\mathbf{Rel}_{\mathbb{K}}$  as codomain. Concretely, since we already know how to assign a relation to each diagram of  $\mathbf{P}\Sigma_{\geq}^+$ , we only need to specify how to interpret finite sets of such diagrams: unsurprisingly, we set

$$\llbracket \{d_1, \dots, d_n\} \rrbracket := \llbracket d_1 \rrbracket \cup \dots \cup \llbracket d_n \rrbracket$$



This is join-preserving by construction, and remains monoidal and functorial.

By definition, we call piecewise-linear (pl) any relation in the image of this functor, i.e., any relation that is a finite union of polyhedral relations. As far as we know, this is the first time that this notion appears in print. However, it does capture our intuitive notion of piecewise-linearity as submanifolds of  $\mathbb{K}^n$  that can be subdivided into linear subspaces.

## 4.2 Equational Theory

$\text{IH}_{PL}$ , the SMLT of pl relations, is presented in Fig. 1. The first block is the theory of matrices/linear maps; the second block,  $\text{IH}$ , axiomatises all linear relations; the third block axiomatises the behaviour of the order  $-\geq-$ ; the fourth, deals with the affine fragment of the theory, axiomatising the behaviour of the constant  $\vdash-$ . Taken together, those four blocks constitute  $\text{IH}_{\geq}^+$ , an axiomatisation of polyhedral relations—we refer the reader to [4] for more details on this fragment.

The key addition of  $\text{IH}_{PL}$  is the last block, containing the axiom of *totality*, which states that any real number belongs to the non-negative *or* to the non-positive fragment of  $\mathbb{K}$ . Remarkably, this simple axiom is the only one we need to add to  $\text{IH}_{\geq}^+$  to obtain a complete theory for pl relations. Its soundness is simply a consequence of the definition of an ordered field: the order is assumed to be total in the sense that, for any  $x, y \in \mathbb{K}$  we have  $x \leq y$  or  $y \leq x$ . Take  $y = 0$  to recover the last axiom of  $\text{IH}_{PL}$ .

*Remark 2.* As a consequence of the Frobenius laws ( $\bullet$ -fr) and of (co)unitality ( $\bullet$ -un)-( $\bullet$ -coun), the diagrams  $\overset{n}{\bullet} \frown \bullet$  and  $\smile \bullet \bullet$  satisfy

$$\overset{n}{\bullet} \frown \bullet \stackrel{\text{IH}_{PL}}{=} \overset{n}{\bullet} \smile \bullet \quad (3)$$

for any  $n$ , the defining equations of a compact closed category. Intuitively, these allow us to forget the direction of wires. In addition, compactness implies the following proposition.

**Proposition 1.**  $C \stackrel{\text{IH}_{PL}}{\subseteq} D \text{ iff } C^{op} \stackrel{\text{IH}_{PL}}{\subseteq} D^{op}$ .

Another important property of compact closed category which we will exploit to simplify the completeness proof is stated in the following proposition. It is an immediate consequence of (3).

**Proposition 2.** *Given  $C, D : m \rightarrow n$ ,  $C \stackrel{\text{IH}_{PL}}{\subseteq} D \text{ iff } \frac{m}{n} \text{C} \stackrel{\text{IH}_{PL}}{\subseteq} \frac{m}{n} \text{D}$*

## 4.3 Completeness Theorem

As we stated above, the axioms in Fig. 1 form a complete theory for pl relations. We will prove that claim in this section. Without loss of generality, using Proposition 2, we restrict to  $n \rightarrow 0$  diagrams.

We start by defining appropriate normal forms for polyhedral and pl relations, and then show that every diagram can be reduced to normal form.

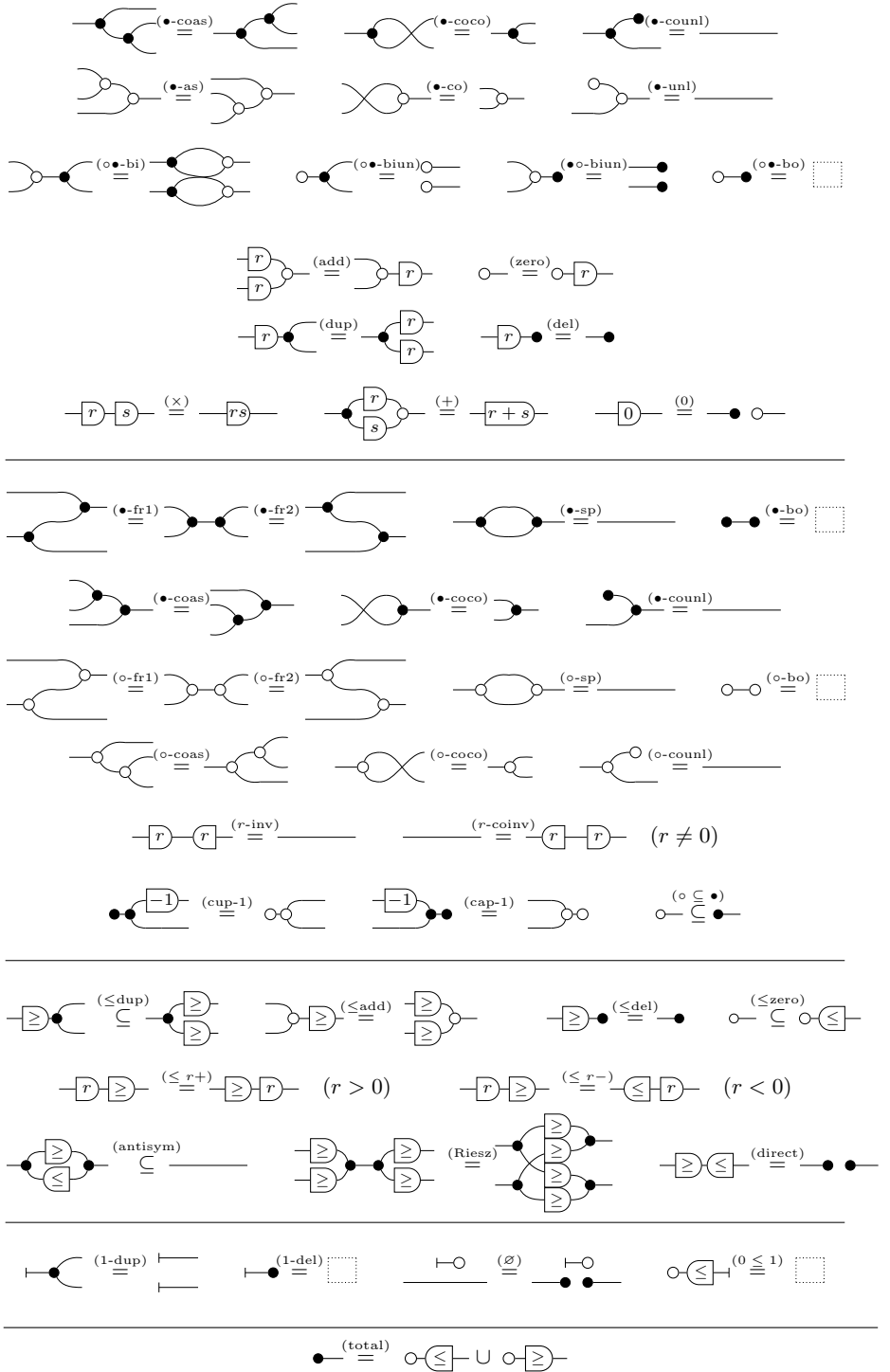


Fig. 1. Axioms of GPLA.

**Definition 2.** We call hyperplane a nonzero affine map  $H : n \rightarrow 1$  which we write  $-\boxed{H}-$ . A given hyperplane  $H$  defines two half-spaces  $-\boxed{H}[\geq]\circ$  and  $-\boxed{H}[\leq]\circ$ , as well as an affine subspace  $-\boxed{H}\circ$ . Since inequality is not strict, the half-spaces include the affine subspace.

In [4, Theorem 14], polyhedral relations have a normal form given by a set of inequations of the form  $A_i x + b_i \geq 0$ . In other words, the normal form is given by an intersection of half-spaces. For our purposes we define a related but slightly different normal form.

**Definition 3.** A  $\mathbf{P}\Sigma_{\geq}^+$ -diagram  $d : n \rightarrow 0$  is in polyhedral normal form if there are hyperplanes  $H_i$  and diagrams  $-(d_i) \in \{\circ, -\boxed{\geq}\circ, -\boxed{\leq}\circ\}$  such that:

$$-(d) = -\bullet \begin{array}{c} \boxed{H_0}-(d_0) \\ \vdots \\ \boxed{H_k}-(d_k) \end{array}$$

Where the  $d_i$  are minimal in the following sense: fixing the set of hyperplanes  $H_i$ , we consider all choices of  $d_i$  that give  $d$  when composed as above. We then require the  $d_i$  in the normal form to be minimal (wrt the order of  $\mathbf{IH}_{\geq}^+$ ) among those. We call the set of the  $d_i$  a valuation for  $d$  relative to the hyperplanes  $H_i$ .

**Definition 4.** We say that a morphism  $D$  of  $\mathbf{P}_{\cup}\Sigma_{\geq}^+$  is in pl normal form if it is written as a non-empty union of diagrams  $d_i$  each in the language of  $\mathbf{P}\Sigma_{\geq}^+$  (i.e. without unions), the  $d_i$  are in the normal form defined in Definition 3, and all the normal forms use the same set of hyperplanes.

**Lemma 1.** Every  $d : n \rightarrow 0$  in  $\mathbf{P}\Sigma_{\geq}^+$  has a polyhedral normal form.

*Proof.* The normal form from [4, Theorem 14] already has the right shape. We only need to find a minimal valuation. Observe that the intersection of two valuations for  $d$  is again a valuation for  $d$ : let  $v$  and  $v'$  be two valuations

for  $d$  relative to the hyperplanes  $H_i$ . If we write  $-\boxed{A}- := -\bullet \begin{array}{c} \boxed{H_0} \\ \vdots \\ \boxed{H_k} \end{array}$  then

$$-\boxed{A}- \begin{array}{c} \circ v \\ \circ v' \end{array} = -\bullet \begin{array}{c} \boxed{A}-v \\ \boxed{A}-v' \end{array} = -\bullet \begin{array}{c} \circ d \\ \circ d \end{array} = -\boxed{d}-$$

Therefore  $v \cap v'$  is again a valuation for  $d$ . Since there are finitely many valuations, we construct the minimal one by intersecting them all.  $\square$

**Lemma 2.** If a morphism  $D$  of  $\mathbf{P}_{\cup}\Sigma_{\geq}^+$  is in pl normal form and  $H$  is a hyperplane, there exists  $C$  in pl normal form such that  $D \stackrel{\text{HPL}}{=} C$  and  $\text{Hyperplanes}(C) = \text{Hyperplanes}(D) \cup \{H\}$ .

*Proof.* We write the normal form of  $D$  as  $D = \bigcup_i d_i$ . Define  $C$  to be the following morphism:

$$-\boxed{C}- = \bigcup_i \begin{array}{c} \circ d_i \\ \bullet \boxed{H}[\geq]\circ \end{array} \cup \bigcup_i \begin{array}{c} \circ d_i \\ \bullet \boxed{H}[\leq]\circ \end{array}$$

We transform  $C$  into  $C'$  by reducing all the terms in the union to polyhedral normal form. This makes  $C'$  be in pl normal form. Since we add the same hyperplane  $H$  to all  $d_i$ ,  $\text{Hyperplanes}(C') = \text{Hyperplanes}(D) \cup \{H\}$ .

Moreover:

$$-\textcircled{C'} = -\textcircled{C} = -\bullet \left( \begin{array}{c} (\bigcup_i -\textcircled{d_i}) \\ \textcircled{H} - (-\textcircled{\geq} \circ \cup -\textcircled{\leq} \circ) \end{array} \right) \stackrel{(\text{total})}{=} -\bullet \left( \begin{array}{c} \textcircled{D} \\ \textcircled{H} \bullet \end{array} \right) = -\textcircled{D}$$

□

**Theorem 1.** *Every morphism of  $\mathbf{P}_{\cup \Sigma}_{\geq}^+$  has a pl normal form.*

*Proof.* Let  $D$  be a  $n \rightarrow 0$  morphism of  $\mathbf{P}_{\cup \Sigma}_{\geq}^+$ . First using distributivity of the union over sequential and parallel composition, we move all the uses of the union to the top-level.

Thus  $D$  is written  $\bigcup_i d_i$  where each  $d_i$  doesn't use the union, i.e. is in the language of  $\mathbf{P}_{\Sigma}_{\geq}^+$ . We then rewrite each  $d_i$  into polyhedral normal form using Lemma 1.

Each  $d_i$  is thus also individually in *pl normal form*, so we can use Lemma 2 to add to each  $d_i$  all the hyperplanes of the other  $d_j$ . For each  $i$  we get a new diagram  $d'_i \stackrel{\text{HPL}}{=} d_i$  in pl normal form, and all the  $d'_i$  use the same set of hyperplanes. So  $\bigcup_i d'_i$  is a pl normal form for  $D$ . □

Before we can prove completeness, we need a final notion: the interior of a polyhedral relation, which is the set of its points that don't touch any of its faces.

**Definition 5.** *Let  $d$  be morphism in polyhedral normal form. We define  $\text{Int}(d)$  to be the set of points  $x \in \llbracket d \rrbracket$  for which  $H_i(x) \neq 0$  when  $-\textcircled{d_i} \neq -\circ$ . In other words,  $H_i(x)$  is nonzero for all hyperplanes where it can be nonzero without  $x$  leaving  $\llbracket d \rrbracket$ .*

Note that we define  $\text{Int}$  only on polyhedral normal form diagrams.  $\text{Int}$  appears to be representation-independent at least when  $\mathbb{K} = \mathbb{R}$ , but we won't try to prove it in the general case as we don't need this here.

*Remark 3.* This is not the usual topological notion of interior. In particular, this notion is independent from the dimension of the surrounding space: a polyhedron of dimension  $0 < k < n$  within  $\mathbb{R}^n$  has an empty topological interior but a nonempty  $\text{Int}$ , as we'll see in the next theorem.  $\text{Int}(d)$  instead coincides with the interior of  $d$  with the topology of the smallest containing affine space.

**Lemma 3.** *Let  $d$  be a diagram in polyhedral normal form. If  $\llbracket d \rrbracket$  is nonempty, then  $\text{Int}(d)$  is nonempty.*

*Proof.* First, write  $d$  in polyhedral normal form:

$$-\textcircled{d} = -\bullet \left( \begin{array}{c} \textcircled{H_0} \textcircled{d_0} \\ \dots \\ \textcircled{H_k} \textcircled{d_k} \end{array} \right)$$

Up to negating some of the  $H_i$ , we can assume that none of the  $\neg(d_i)$  are  $\neg(\leq)\circ$ . If  $\forall i. \neg(d_i) = \neg\circ$ , then by definition  $\text{Int}(d) = \llbracket d \rrbracket$  which is nonempty so we're done. Assume then that  $\neg(d_i) = \neg(\geq)\circ$  for at least some  $i$ . For each such  $i$ , by minimality of the  $d_i$  in the normal form there must be a  $x_i \in \llbracket d \rrbracket$  such that  $H_i(x_i) > 0$ . We pick such an  $x_i$  for each  $i$ , and define  $x := \frac{1}{p} \sum_i x_i$  to be their average. By convexity,  $x \in \llbracket d \rrbracket$ .  $H_i$  is an affine map, hence is concave, thus if we had picked an  $x_i$  then  $H_i(x) \geq \frac{1}{p} \sum_j H_i(x_j) \geq \frac{1}{p} H_i(x_i) > 0$ . Then for each  $i$  either  $\neg(d_i) = \neg\circ$  or  $H_i(x) > 0$ , hence  $x \in \text{Int}(d)$ .  $\square$

**Theorem 2 (Completeness).**  $\llbracket D \rrbracket \subseteq \llbracket C \rrbracket \implies D \stackrel{\text{IH}_{PL}}{\subseteq} C$

*Proof.* Using Proposition 2 we can without loss of generality assume that  $D$  and  $C$  have  $n$  inputs and 0 outputs. Using Theorem 1, we reduce  $D$  and  $C$  into pl normal form. Using Lemma 2, we add each others' hyperplanes to  $D$  and  $C$  so that they both use the exact same set. So  $D = \bigcup_i d_i$  and  $C = \bigcup_i c_i$ , where the  $d_i$  and  $c_i$  are in polyhedral normal form and use a same set of hyperplanes  $\{H_i\}_i$ . Pick one of the  $d_i$  in  $D$ .

If  $d_i$  is the empty polyhedron, we have  $\llbracket d_i \rrbracket = \emptyset \subseteq \llbracket c_0 \rrbracket$ , so by completeness of  $\text{IH}_{\geq}^+$  we get  $d_i \stackrel{\text{IH}_{\geq}^+}{\subseteq} c_0$ . Thus  $d_i \stackrel{\text{IH}_{PL}}{\subseteq} c_0 \stackrel{\text{IH}_{PL}}{\subseteq} C$ .

Otherwise  $d_i$  is nonempty, and using Lemma 3 we pick  $x \in \text{Int}(d_i)$ . Then:

$$x \in \text{Int}(d_i) \subseteq \llbracket d_i \rrbracket \subseteq \llbracket D \rrbracket \subseteq \llbracket C \rrbracket = \left\llbracket \bigcup_j c_j \right\rrbracket = \bigcup_j \llbracket c_j \rrbracket$$

Thus there is a  $j$  such that  $x \in \llbracket c_j \rrbracket$ . Now pick a  $k$ . If  $\neg(d_{ik}) = \neg\circ$ , then  $\neg(d_{ik}) \stackrel{\text{IH}_{PL}}{\subseteq} \neg(c_{jk})$  regardless of  $\neg(c_{jk})$ . If  $\neg(d_{ik}) = \neg(\geq)\circ$ , then by definition of  $\text{Int}(d_i)$ , we have  $H_k(x) > 0$ . Since moreover  $x \in \llbracket c_j \rrbracket$ ,  $\neg(c_{jk})$  must be  $\neg(\geq)\circ$ . If  $\neg(d_{ik}) = \neg(\leq)\circ$ , similarly  $\neg(c_{jk})$  must be  $\neg(\leq)\circ$ . In all three cases,  $\neg(d_{ik}) \stackrel{\text{IH}_{PL}}{\subseteq} \neg(c_{jk})$ . This is the case for every  $k$ , so:

$$\neg(d_i) = \neg \left( \begin{array}{c} \boxed{H_0} \circ (d_{i0}) \\ \vdots \\ \boxed{H_m} \circ (d_{im}) \end{array} \right) \subseteq \neg \left( \begin{array}{c} \boxed{H_0} \circ (c_{j0}) \\ \vdots \\ \boxed{H_m} \circ (c_{jm}) \end{array} \right) = \neg(c_j) \subseteq \neg(C)$$

Finally, since we have  $d_i \stackrel{\text{IH}_{PL}}{\subseteq} C$  for all  $i$ , we derive  $D = \bigcup_i d_i \stackrel{\text{IH}_{PL}}{\subseteq} C$ .  $\square$

## 5 Generating Piecewise-Linear Relations

Piecewise-linear subsets of vector spaces give us a rather wide semantic space to explore. One might suspect that there exist useful structured relations that live strictly between the linear and piecewise-linear worlds.

Formally, we're interested in finding sub-props of  $\text{Rel}_{\mathbb{K}}$  that contain not only linear or polyhedral relations, but some selected non-convex relations that would be useful for particular applications. It turns out that for many sensible choices, the resulting image will coincide with pl relations—a somewhat surprising fact. Note that we are interested in generating sub-props of  $\text{Rel}_{\mathbb{K}}$  here, not  $\cup$ -props, since the  $\cup$ -prop generated by the image of  $\text{P}_{\cup}\Sigma_{\geq}^+$  under  $\llbracket \cdot \rrbracket$  already contains all pl relations.

We will go through a few natural choices, each time defining them as a term of  $\text{P}_{\cup}\Sigma_{\geq}^+$ , a shortcut which makes reasoning about them much easier than with their set-theoretic semantics. Of course, their semantics in  $\text{Rel}_{\mathbb{K}}$  can be recovered via  $\llbracket \cdot \rrbracket$ .

### 5.1 The $n$ -Fold Union Generators

We first show that the main difference between polyhedral and pl relations—the unions—can be bridged. Indeed, it is not obvious that we can build arbitrary unions of diagrams without having access to the syntax of a SMLT. For this we introduce a family of diagrams we call the  $n$ -fold union generators, defined for a given  $n$  as:

$$n \text{ } \bigcup \text{ } n := n \text{ } \text{---} \text{ } n \cup n \text{ } \text{---} \text{ } n$$

These generators suffice to reproduce the behaviour of the syntactic union:

**Theorem 3.** *The image of the free prop generated by  $\Sigma_{\geq}^+$  and the  $n$ -fold union generators for all  $n$  is the prop of pl relations.*

*Proof.* If  $\text{---} \textcircled{C}$  and  $\text{---} \textcircled{D}$  are non-empty  $n \rightarrow 0$  diagrams,

$$n \text{ } \bigcup \text{ } \begin{matrix} \textcircled{C} \\ \textcircled{D} \end{matrix} = n \text{ } \text{---} \begin{matrix} \bullet \textcircled{C} \\ \textcircled{D} \end{matrix} \cup n \text{ } \text{---} \begin{matrix} \textcircled{C} \\ \bullet \textcircled{D} \end{matrix} = \text{---} \textcircled{C} \cup \text{---} \textcircled{D}$$

Since every pl relation can be written as a finite union of diagrams in  $\text{P}\Sigma_{\geq}^+$ , and we can easily avoid diagrams denoting the empty relation, this generates all of pl relations.  $\square$

This means that we didn't formally need to introduce the notion of a SMLT after all: we could have defined an equivalent SMIT by adding these generators. However, this is for most purposes a much less convenient syntax, and the corresponding equational theory would be more difficult to calculate with. This is also the case for the examples that follow.

### 5.2 The Simplest Non-Convex Diagram

The following is one of the simplest diagrams that captures a non-convex relation:

$$\text{---} \textcircled{+} \text{---} := \text{---} \circ \bullet \text{---} \cup \text{---} \bullet \text{---} \circ \text{---}$$

It is named after its semantics: the union of the  $x$  and  $y$  axes in the plane, corresponding to the simple equation  $x = 0 \vee y = 0$ . Despite its simplicity, it suffices to generate all of pl relations.

**Theorem 4.** *The image of the free prop generated by  $\Sigma_{\geq}^+$  and  $-(+)-$  is the prop of pl relations.*

*Proof.* Define  $\text{dup} : 1 \rightarrow 2$ :

This diagram has the interesting property of duplicating black and white units:

$$\circ \text{---} (\text{dup}) \text{---} = \circ \text{---} \quad \bullet \text{---} (\text{dup}) \text{---} = \bullet \text{---}$$

We can chain it to build  $-(\text{dup})^{n+1} := -(\text{dup}) \text{---} (\text{dup})^n$  for any  $n$ .

Then, let  ${}^n(\text{+})^n := {}^n \text{---} (\text{dup}^{op})^1 (\text{+})^1 (\text{dup})^n = {}^n \circ \bullet \cup {}^n \bullet \circ {}^n$

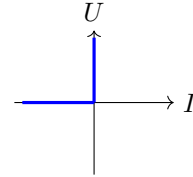
This allows us to build:

$$\begin{aligned} & \text{Diagram 1: } {}^n \text{---} (\text{+})^n \text{---} {}^n \\ &= \text{Diagram 2: } {}^n \text{---} (\text{dup}^{op})^1 (\text{+})^1 (\text{dup})^n \text{---} {}^n \\ &= \text{Diagram 3: } {}^n \text{---} \bullet \cup {}^n \bullet \text{---} {}^n \\ &= \text{Diagram 4: } {}^n \text{---} (\cup) \text{---} {}^n \end{aligned}$$

□

### 5.3 The Semantics of a Diode

Most basic electrical circuit components can be modelled with an affine semantics. The first exception is the (ideal) diode: the idealised current-voltage semantics across a diode is that the current can be negative and the voltage difference positive but not both at the same time.



On a graph, the allowed (current, voltage difference) pairs are depicted above. Not only is this not affine, it is not even convex. The corresponding diagram,  $-(\leq) \circ \text{---} \cup \text{---} \circ \text{---} (\leq)$ , is outside of both affine and polyhedral algebra.

We will see how to model electrical circuits with diodes in more detail in the next section. We will focus here on the following fact: adding a generator with this semantics is once again enough to recover all pl relations. In fact we can even build the  $\geq$  relation from the diode, so we can start from affine algebra (without requiring the generality of polyhedral algebra).

For convenience, we define a new generator whose semantics is the mirror image of the diode's graph:

$$-(\text{L})- := -(\geq) \circ \text{---} \cup \text{---} \circ \text{---} (\leq)-$$

**Theorem 5.** Recall that  $\Sigma^+$  is  $\Sigma_{\geq}^+$  without  $-\boxed{\geq}-$ . The image of the free prop generated by  $\Sigma^+$  and  $L$  is the prop of pl relations.

*Proof.* First, we can construct the  $\geq$  generator from  $L$ :

$$-\boxed{L}- = -\boxed{\geq}- \circ -\boxed{\leq}- = -\boxed{\geq}- \cup -\boxed{\leq}- = -\boxed{\geq}-$$

So we generate all polyhedral relations. Then we can also recover the  $+$  generator from the previous section, which is enough to generate all of pl relations:

$$\begin{aligned} -\boxed{L}- \circ -\boxed{L}- \circ -\boxed{-1}- &= -\boxed{\geq}- \circ -\boxed{\geq}- \cup -\boxed{\geq}- \circ -\boxed{\leq}- \\ &\cup -\boxed{\leq}- \circ -\boxed{\geq}- \cup -\boxed{\leq}- \circ -\boxed{\leq}- \\ &= -\boxed{\geq}- \circ -\boxed{\geq}- \cup -\boxed{\leq}- \circ -\boxed{\leq}- \cup -\boxed{\geq}- \circ -\boxed{\leq}- \cup -\boxed{\leq}- \circ -\boxed{\geq}- \\ &= -\boxed{\geq}- \circ -\boxed{\geq}- \cup -\boxed{\leq}- \circ -\boxed{\leq}- = -\boxed{+}- \end{aligned}$$

$$\begin{aligned} -\boxed{+}- \circ -\boxed{-1}- \circ -\boxed{+}- &= -\boxed{\geq}- \circ -\boxed{\leq}- \cup -\boxed{\geq}- \circ -\boxed{\geq}- \\ &\cup -\boxed{\leq}- \circ -\boxed{\geq}- \cup -\boxed{\leq}- \circ -\boxed{\leq}- \\ &= -\boxed{\geq}- \circ -\boxed{\geq}- \cup -\boxed{\leq}- \circ -\boxed{\leq}- \cup -\boxed{\geq}- \circ -\boxed{\leq}- \cup -\boxed{\leq}- \circ -\boxed{\geq}- \\ &= -\boxed{\geq}- \circ -\boxed{\geq}- \cup -\boxed{\leq}- \circ -\boxed{\leq}- = -\boxed{+}- \end{aligned}$$

□

#### 5.4 Alternative generators: $\max$ , $ReLU$ and $\text{abs}$

Three of the most basic piecewise-linear functions one might come across are  $\text{abs}$ ,  $\max$  and  $ReLU$ . We define them diagrammatically as follows:

$$\begin{aligned} \text{max} &:= -\boxed{\leq}- \cup -\boxed{\leq}- \\ \text{abs} &:= -\boxed{-1}- \circ \text{max} \\ \text{ReLU} &:= \text{max} \end{aligned}$$

While the reader will certainly be familiar with the first two,  $ReLU$  has acquired significant fame as one of the basic building blocks of neural networks. In fact, all neural networks whose activation function is  $ReLU$  can be represented in GPLA. This opens up the exciting possibility of applying equational reasoning to neural networks, a possibility that we leave for future work.

Once again, adding either of them to the syntax for affine algebra suffices to construct any pl relation.



**Theorem 6.** *The image of the free prop generated by  $\Sigma^+$  and any of  $\max$ ,  $\text{abs}$  or  $\text{ReLU}$  is the prop of pl relations.*

*Proof.* First, we notice that the three functions are inter-definable.  $\text{abs}$  and  $\text{ReLU}$  were already defined in terms of  $\max$ , and we can complete the cycle:

$$\max(x, y) = x + \max(0, y - x) = x + \text{ReLU}(y - x)$$

$$\text{ReLU}(x) = \max(0, x) = (x + \text{abs}(x))/2$$

So we only need to show the result for one of them. Let's pick  $\max$ . We recover  $\mathbf{L}$ , which we know suffices by Theorem 5. First  $\text{max} \circ = \text{ReLU} \cup \text{abs}$ .

Thus  $\text{max} \circ = \text{ReLU} \cup \text{abs} = \mathbf{L} \quad \square$

*Remark 4.* It is standard that  $\max$  together with linear maps generates all continuous pl functions. Our result can be seen as a generalization of this fact to the relational setting.

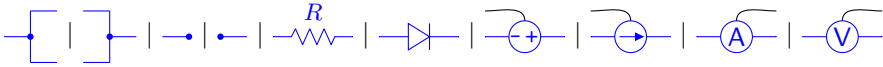
## 5.5 Conclusion

These examples justify the generality of pl relations: they constitute the minimal extension of polyhedral algebra (and in some cases affine algebra) that can express any of the very useful relations above. This is interesting because pl relations form a nearly universal domain: they can approximate any smooth manifold over a bounded domain.

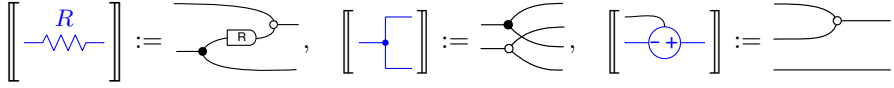
Despite our compelling examples, there could still be interesting props between polyhedral and pl relations. In particular, determining the prop generated by  $\Sigma_{\geq}^+$  together with  $\circ \cup \vdash$  is currently an open problem.

## 6 Case Study: Electronic Circuits

To illustrate how one would use this theory in a concrete case, we turn to the study of electronic circuits. We build on the work done in [3]. The syntax mimics the usual circuits drawn by electrical engineers, by generating a free two-colored prop from basic elements and wires. The blue wires are electrical wires, and the black wires carry information; for details see [3].

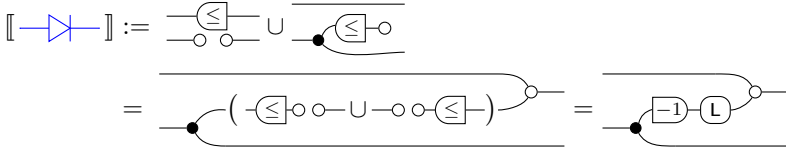


The corresponding physical model imposes constraints between two quantities: current and voltage. To express this, we map an electrical wire into two GPLA wires, the top one for voltage and the bottom one for current. We then give to each generator a semantics in GPLA that expresses the relevant physical equations. For example:



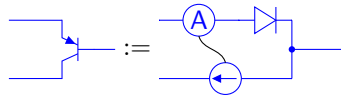
The core of this approach is the fact that composition of constraints in GPLA gives the behaviour of the corresponding composite electrical circuit. We can thus define the semantics of a whole circuit compositionally, and get the physically expected result.

So far this follows exactly [3]. Our contribution is the ability to express the behaviour of diodes:



*Remark 5.* We cannot include capacitors and inductors, because they require semantics in  $\mathbb{IH}_{\mathbb{R}(x)}^+$ , and  $\mathbb{R}(x)$  cannot be ordered in a way that would be consistent with the physics. Finding diagrammatic semantics that can accommodate both capacitors and diodes is an important open problem.

This extension allows us to model electronic circuits! As hinted in the previous section, diodes by themselves can be used to build many things. For example, we can model a simple idealized transistor as follows: [10, Fig. 59.1]



That said, it is impractical to prove the equality of two non-trivial electronic circuits explicitly as the number of alternatives grows exponentially in the number of diodes. Like in standard mathematical practice, making this practical will require finding appropriate techniques and approximations, which we leave for future work.

*Acknowledgements.* The authors would like to thank the various Twitter and Zulip users who contributed to the genesis and development of the theory contained in this paper, notably Jules Hedges, Cole Comfort and Reid Barton. Reid Barton in particular contributed significantly to the proof of completeness.

The first author is funded by the EPSRC under grant OUCS/GB/1034913. The second author acknowledges support from EPSRC grant EP/V002376/1.

## References

1. Baez, J.C., Coya, B., Rebro, F.: Props in network theory. *Theory and Applications of Categories* **33**(25), 727–783 (2018)

2. Baez, J.C., Fong, B.: A compositional framework for passive linear networks. *Theory and Applications of Categories* **33**(38), 1158–1222 (2018)
3. Boisseau, G., Sobociński, P.: String Diagrammatic Electrical Circuit Theory. arXiv:2106.07763 [cs] (2021), <http://arxiv.org/abs/2106.07763>
4. Bonchi, F., Di Giorgio, A., Sobocinski, P.: Diagrammatic Polyhedral Algebra. arXiv:2105.10946 [cs, math] (2021), <http://arxiv.org/abs/2105.10946>
5. Bonchi, F., Holland, J., Piedeleu, R., Sobociński, P., Zanasi, F.: Diagrammatic algebra: from linear to concurrent systems. In: *Proceedings of the 46th Annual ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)* (2019)
6. Bonchi, F., Piedeleu, R., Sobociński, P., Zanasi, F.: Graphical Affine Algebra. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. pp. 1–12. IEEE, Vancouver, BC, Canada (2019). <https://doi.org/10.1109/LICS.2019.8785877>
7. Bonchi, F., Sobociński, P., Zanasi, F.: Deconstructing lawvere with distributive laws. *Journal of logical and algebraic methods in programming* **95**, 128–146 (2018)
8. Cvitanovic, P., Cvitanović, P.: *Group theory*. Princeton University Press (2008)
9. Stephan, P.R., Brayton, R.K.: Physically realizable gate models. In: *Proceedings of 1993 IEEE International Conference on Computer Design ICCD'93*. pp. 442–445. IEEE (1993)
10. Theraja, B., Theraja, A.: *A textbook of electrical technology : in S.I. system of units*. Publication division of Nirja Construction and Development Co., New Delhi (1994)
11. Willems, J.C.: The behavioral approach to open and interconnected systems. *IEEE Control Systems Magazine* **27**(6), 46–99 (2007)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

