UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS RUFINO MARTELOTTE

RELATIONAL GRAPHICAL LINEAR ALGEBRA

RIO DE JANEIRO
2023

LUCAS RUFINO MARTELOTTE

RELATIONAL GRAPHICAL LINEAR ALGEBRA

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. João Antônio Recio da Paixão

RIO DE JANEIRO
2023

LUCAS RUFINO MARTELOTTE

RELATIONAL GRAPHICAL LINEAR ALGEBRA

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em ___ de _______________ de _______

BANCA EXAMINADORA:

João Antônio Recio da Paixão
UFRJ

Hugo Nobrega
UFRJ

Iago Leal de Freitas
PSR

AGRADECIMENTOS

First, we would like to thank CNPq for more than two years of financial support to this project. Special thanks also go to Pawel Sobociński, Iago Leal and Gabriel Ferreira for directly helping with the development of this work.

RESUMO

Desde o final do século XIX, a noção de formalizar a matemática se tornou um tópico cada vez mais importante. Os benefícios da formalização são bem conhecidos, desde o uso de verificadores de provas (proof checkers) até o desenvolvimento de algoritmos. Apesar disso, o formalismo muitas vezes vem ao custo de legibilidade, e geralmente é considerado uma espécie de pós-processamento. Primeiro, a teoria é desenvolvida e entendida de maneira informal, e só então é feito o processo custoso de transferí-la para um ambiente formal. A boa notícia é que, frequentemente, há maneiras de desenvolver uma teoria de maneira mais formal sem sacrificar intuição e legibilidade. Isso pode ser feito, por exemplo, com o uso de provas calculacionais, que é um estilo de prova desenvolvido por Dijkstra (DIJKSTRA; SCHOLTEN, 2012). É uma maneira estruturada de provar proposições, semelhante a uma linguagem de programação. Em outras palavras, assim como linguagens de alto nível, elas mantém a legibilidade e, ao mesmo tempo, são estruturadas o suficiente para permitirem uma fácil tradução para um ambiente formal (por exemplo, um computador). A álgebra linear, como uma das áreas fundamentais da matemática, parece ser um lugar importante para introduzir provas calculacionais. Uma abordagem em particular, chamada álgebra linear relacional gráfica, já faz isso e parece ter muito potencial nesse sentido. É uma linguagem gráfica para álgebra linear e foi desenvolvida inicialmente por Pablo Zanasi em sua tese Interacting Hopf Algebras (ZANASI, 2015). A parte *relacional* representa o uso de relações lineares: em vez de considerar os espaços vetoriais e as transformações lineares como duas entidades separadas, como geralmente é feito na álgebra linear, as relações lineares são capazes de generalizar ambos os conceitos em um só. A parte *gráfica* representa o uso de diagramas de corda, que são diagramas formais bidimensionais, definidos indutivamente. Eles trazem intuição visual para os argumentos sem sacrificar a formalidade e costumam fazer provas ficarem curtas e formais. No entanto, a álgebra linear relacional gráfica ainda não foi desenvolvida o suficiente, no sentido de que a maioria dos teoremas fundamentais de álgebra linear ainda não foram provados nesta linguagem. Este trabalho pretende ser um passo em direção a esse objetivo. Desenvolvemos um conjunto de axiomas mais curto e simétrico. Damos uma definição indutiva de matrizes completamente dentro da linguagem gráfica e um algoritmo para multiplicação de matrizes. Por fim, usando uma decomposição relacional bem conhecida, provamos vários resultados fundamentais de álgebra linear com provas curtas e calculacionais.

Palavras-chave: álgebra linear; linguagem gráfica; relações lineares; provas calculacionais; recursão; linguagens formais.

ABSTRACT

Since the late 19th century, the notion of formalizing mathematics has become an increasingly important topic. The benefits of formalization are well known, ranging from the use of proof checkers to the derivation of algorithms. Despite that, formalism often comes at the cost of readability, and is usually regarded as a kind of post processing. First, one carefully develops and understands the theory in an informal environment, and then goes through the laborious process of transferring it to a formal one. The good news is that, often, there are ways of developing a theory in a semi-formal fashion without sacrificing intuition and readability. This can be done, for instance, with the use of calculational proofs, which is a style of proof developed by Dijkstra (DIJKSTRA; SCHOLTEN, 2012). It is a structured way of proving statements, similar to a programming language. In other words, just like modern high-level languages, it retains human readability, while being structured enough to easily allow translation to a formal environment (i.e. a computer). Linear algebra, as one of the fundamental areas of mathematics, seems to be an important place to introduce calculational proofs. In fact, one particular approach, called relational graphical linear algebra, already does this and seems to have much potential in that regard. It is a graphical language for linear algebra and was initially developed by Pablo Zanasi in his thesis Interacting Hopf Algebras (ZANASI, 2015). The *relational* part stands for the use of linear relations: instead of regarding vector spaces and linear transformations as two separate entities, as it is usually done in linear algebra, linear relations are able to generalize both concepts into a single one. The *graphical* part stands for the use of string diagrams, which are 2-dimensional formal diagrams, which are defined inductively. They bring visual intuition to the arguments without sacrificing formality, and usually make proofs short and formal. However, it has not yet been developed enough, in the sense that most of the fundamental theorems are yet to be proven in this language. This work aims to be a step forward towards that goal. We develop a shorter, more symmetric set of axioms. We give an inductive definition of matrices completely inside the graphical language and an algorithm for matrix multiplication. Lastly, using an appropriate decomposition result, we prove various fundamental linear algebra results with short calculational proofs.

Keywords: linear algebra; graphical language; linear relations; calculational proofs; recursion; formal systems.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 INFORMAL VS. FORMAL MATHEMATICS

Since the late 19th century, with the works of Kurt Gödel and David Hilbert, the notion of formalizing mathematics has been heavily studied and developed. Formalization can be summarized informally in one sentence: it is the act of explaining something so that a computer can understand. By bringing math to the computer level, we can make use of proof-checkers to make sure our theory is sound (BEESON; NARBOUX; WIEDIJK, 2019). Also, it becomes easier to translate constructive proofs into algorithms (BACKHOUSE; FERREIRA, 2011). Despite these advantages, formalized systems are often complicated, and the proofs are hard for humans to read. Consequently, formalization is still regarded as a sort of postprocessing. First, one understands and develops the theory carefully in an informal language, and only then goes through the laborious process of formalizing it inside a computer. In a sense, these are seen as two separate processes. There is the informal world, where humans can understand and develop informal proofs, but computers can't, and there is the formal world, where computers can check the proofs and run algorithms, but the proofs are so complex it becomes unreadable for humans. For some, readability and formalism may even seem like opposite concepts. It would be desirable to have a way of taking advantage of a formalized system while still carrying the readability of an informal one.

1.1.1 Programming languages lie between informal and formal

A similar problem occurred with the development of computer programs. In the middle of the 20th century, programmers had to first describe their code in an informal, readable language that could be transmitted to other programmers, and after that, they had to exhaustively convert their code to machine language. During that time, communication between humans and computers was difficult, and the main solution to that problem was the use of programming languages. In a sense, they are a middle agent between humans and machines. Code can be written in a way other humans can understand, and the process of converting it to machine language is so well determined it can be automatized (by the use of compilers). In fact, since that breakthrough, programming languages became closer and closer to the way humans communicate, without losing the formality necessary to be converted into machine language. Examples of this are high-level languages such as Python, Haskell, and even programming languages developed for children.

An interesting informal question is what would be the equivalent of a "programming language" inside of math. One option would be to develop the theory using a proof assistant like Coq or Lean. For some people, however, that may already be too close to "machine language". Another option is using a style of proof similar, or equal, to a calculational proof.

1.1.1.1 Calculational proofs: making proofs more like programs

Calculational proof is a style of proving introduced by Dijkstra and Scholten in their book Predicate Calculus and Program Semantics (DIJKSTRA; SCHOLTEN, 2012; LIFSCHITZ, 2022; GRIES; SCHNEIDER, 1995; BACK; GRUNDY; WRIGHT, 1997). The key idea is that statements are boolean objects, and a proof is nothing but a series of transformations applied to a statement to arrive at an equivalent statement, or the value *true*. For instance, when trying to prove that A is equivalent to D, one might first transform it into a statement B, which would then be transformed into a statement C, which lastly would be transformed into D. These transformations would all be equivalencies. In the calculational proof format, these steps are shown in sequential order, and in between them lies their justification.

$$
\begin{array}{ll}
 & A \\
\iff & \quad \{ \text{ hint why } A \iff B \ \} \\
 & B \\
\iff & \quad \{ \text{ hint why } B \iff C \ \} \\
 & C \\
\iff & \quad \{ \text{ hint why } C \iff D \ \} \\
 & D
\end{array}
$$

Thanks to the transitivity of equivalencies, these steps put together would imply that $A \iff D$. Notice how that proof format is similar to a series of instructions. If a constructive proof were to be written in that fashion, the individual steps would be visible. The act of translating it into a computer algorithm would be a matter of encoding the individual steps into it, much like what a compiler does. That same proof format could be adapted to all sorts of arguments. Notice that equivalence is the connector between the steps, but any transitive relation would suffice. For instance, it is possible to use $\implies$, $\subseteq$ and $=$.

Comparing math to a programming language is a useful analogy. Some properties that are useful in one context tend to be useful in the other. For instance, recursion is an important concept inside programming and, inside math, recursive data structures are the heart of inductive proofs. Axioms could be compared to dependencies of a computer program. One could go wild with these thoughts. Just keep in mind these are informal analogies and in no way suggest a formal connection (even though there is one!) between programming and mathematics.

The pursuit of formality can lead to interesting insights into different areas of mathematics. In this work, we will focus on a single, fundamental area: linear algebra.

1.2 LINEAR ALGEBRA

Linear algebra is one of the engine rooms of science, underpinning both recent advances in, e.g., data science and machine learning, as well as powering other subjects that saw rapid development during the 20th century, e.g., control theory, computer graphics, network theory, quantum computing, and many others. Seemingly unrelated technologies are inherently linear algebraic: e.g., the Google PageRank concept is an eigenvector, and numerical linear algebra algorithms are at the core of convex optimization. Further afield, linear algebra is the backbone of much of modern mathematical physics; e.g., non-linear differential equations are solved by iterating linear systems. It is difficult to overstate the sheer practicality of linear algebra theory and its influence in shaping modern science.

1.2.1 Why care about language in linear algebra

> “Using matrix notation such a set of simultaneous equations takes the form $Ax = b$ where x is the vector of unknown values, A is the matrix of coefficients and b is the vector of values on the right side of the equation. In this way, a set of equations has been reduced to a single equation. This is a tremendous improvement in concision that does not incur any loss of precision!” (BACKHOUSE; OLIVEIRA, 2006)

Language has always played an important role in linear algebra. One of the most useful language developments inside the theory was the representation of linear systems in a compact way, namely, the use of the point-free notation $Ax = b$ instead of the extensive listing of equalities.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \to Ax = b \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \tag{1.1}$$

In (1.1), the symbols a_{ij} and b_i are not necessary to represent the idea of a linear system. Unique solutions can be expressed merely by the expression $x = A^{-1}b$, without the need of looking “inside” the matrix A. Some aspects, however, may still be improved. One example is the way algorithms are developed mathematically. Linear algebra has lots of algorithms, which require proofs and implementations. Normally, these two things are done separately. One first needs to construct some decomposition mathematically, and after that figure out how to implement it inside a programming language. Calculational proofs could help diminish the gap between these two processes. Proofs done in this style have clear steps, which usually can be translated into computer instructions without difficulty.

1.2.1.1 Example of informal linear algebra

Such an old and fundamental field as linear algebra naturally has already been formulated in several different ways. One of the modern and widely used books is Linear Algebra Done Right by Sheldon Axler (AXLER, 2015), which is a common textbook in universities. Inside the book, linear transformations and vector spaces are defined using sets. The intersection of subspaces, for example, is just the intersection of sets, which will probably spark the image of a Venn Diagram inside most people's heads. Hence, proofs in the book are intuitive in the geometric sense. However, some disadvantages do exist. For a good portion of the book, there are no matrices, and instead the focus is on the set-theoretical definition of linear transformations. This, first of all, leads to point-dependency, since there is the need to pick individual elements out of the vector space's set representation. Also, proofs in the book are wordy, in the sense that there is a lot of explaining inside of them. This leads to long arguments and informality. Consider the following proof present in the book.

Lemma 1 (Every subspace of V is part of a direct sum equal to V). *Let F be a field and V a finite-dimensional vector space in F^k. Let U be a subspace of V. Then there is a subspace W of V such that $V = U \oplus W$.*

Demonstração. Because V is finite dimensional, so is U. Thus there is a basis $u_1, ..., u_m$ of U. Of course, $u_1, ..., u_m$ is a linearly independent list of vectors in V. Hence this list can be extended to a bases $u_1, ..., u_m, w_1, ..., w_n$ of V. Let $W = \text{span}(w_1, ..., w_n)$.

To prove that $V = U \oplus W$, we need only show that

$$V = U + W \text{ and } U \cap W = \{0\}.$$

To prove the first equation above, suppose $v \in V$. Then, because the list $u_1, ..., u_m, w_1, ..., w_n$ spans V, there exist $a_1, ..., a_m, b_1, ..., b_n \in F$ such that

$$v = a_1 u_1 + ... + a_m u_m + b_1 w_1 + ... + b_n w_n$$

for any $v \in V$. Thus, $V = U + W$.

To show that $U \cap W = \{0\}$, suppose $v \in U \cap W$. Then there exists scalars $a_1, ..., a_m, b_1, ..., b_n \in F$ such that

$$v = a_1 u_1 + ... + a_m u_m = b_1 w_1 + ... + b_n w_n.$$

Thus

$$a_1u_1 + ... + a_mu_m - b_1w_1 - ... - b_nw_n = 0.$$

Because $u_1, ..., u_m, w_1, ..., w_n$ is linearly independent, this implies that $a_1 = ... = a_m = b_1 = ... = b_n = 0$. Thus $v = 0$, completing the proof that $U \cap W = \{0\}$. $\square$

While this style of proving contributes to human understanding, it diminishes computer implementability, since there usually are no clear steps in the proof to be translated into an algorithm. In a field with so many computer science applications, this may be a big disadvantage to some people. One may regard this formulation of linear algebra as a human-oriented approach, equivalent to the "readable language" programmers had to use in the computer science analogy.

1.2.1.2 Example of formal linear algebra

There is also the other end of the spectrum. One might call it the computer-oriented approach, equivalent to "machine language" in the computer science analogy. These approaches are usually done inside proof assistants such as Coq and Lean. One example is the paper Point-free Set-free Concrete Linear Algebra by Georges Gonthier (GONTHIER, 2011). There, linear transformations and vector spaces are completely defined in terms of vectors and matrices. Proofs are formal and written inside the Coq proof assistant (which was used in the formalization of the Odd-Number Theorem), so they already correspond to algorithms. Also, as the paper's name suggests, this formulation is point-free. However, it is not as readable as Axler's book, which may be a big disadvantage. As an example, we show in Figure 1 the gaussian elimination algorithm implementation in Coq, as done in the original work.

These two examples (Axler's book and Gonthier's paper) serve different purposes. One is made for teaching students, while the other is made for computer formalization. One is left to wonder what could serve as a midpoint between these two. One recent approach, called Relational Graphical Linear Algebra (usually abbreviated to GLA), seems to be a good candidate for that. It manages to encapsulate the theory in a simple, formal and elegant language where all proofs can be derived calculationally.

1.2.2 A language in between: Relational Graphical Linear Algebra

Graphical Linear Algebra (GLA) is a new approach to linear algebra developed by Fabio Zanasi in his thesis Interacting Hopf Algebras (ZANASI, 2015). It was further explored by other related works (BONCHI; SOBOCIŃSKI; ZANASI, 2017a; PAIXÃO; SOBOCIŃSKI, 2020; FREITAS, 2019), including a nice blog by Pawel Sobocinski which showcases it in a more introductory fashion (SOBOCIŃSKI,). It is a rather radical approach, in the sense that both syntax and semantics are different from what is commonly used. The semantics is relational, while the syntax is graphical.

```
Fixpoint gaussian_elimination {m n} :=
  match m, n return ’M_(m, n) -> ’M_m * ’M_n * nat with
  | _.+1, _.+1 => fun A : ’M_(1 + _, 1 + _) =>
    if [pick ij | A ij.1 ij.2 != 0] is Some (i, j) then
      let a := A i j in let A1 := xrow i 0 (xcol j 0 A) in
      let u := ursubmx A1 in let v := a^-1 *: dlsubmx A1 in
      let: (L, U, r) := gaussian_elimination (drsubmx A1 - v *m u)
      in (xrow i 0 (block_mx 1 0 v L),
          xcol j 0 (block_mx a%:M u 0 U),
          r.+1)
    else (1%:M, 1%:M, 0%N)
  | _, _ => fun _ => (1%:M, 1%:M, 0%N)
  end.
```

Fig. 1 – Gaussian Elimination in Coq (GONTHIER, 2011)

1.2.2.1 Why relational

Semantics in linear algebra is mostly functional: the two objects of study are vector spaces and linear transformations (linear functions) between them. In GLA, however, the semantics are relational. That means the only objects inside the theory are linear relations. The use of relations inside mathematics has been studied for more than 150 years with works of DeMorgan, Pierce and Tarski (MADDUX, 1991; FEFERMAN, 2006; GIVANT, 2006; SCHMIDT, 2011). They provide some advantages in comparison to the functional approach. One can view vector spaces and linear transformations as particular cases of the general concept of linear relations. Besides the beauty of reducing everything to a single object, this view provides us with more streamlined proofs where more symmetries and dualities of linear algebra are apparent. There are many examples of long wordy proofs which can be considerably shortened with this relational view.

Given a field k, a linear relation can be defined as follows.

Definition 2. *Fix a field k and k-vector spaces V, W. A linear relation from V to W is a set $R \subseteq V \times W$ that is a subspace of $V \times W$, considered as a k-vector space. Explicitly this means that:*

1. $(0, 0) \in R$.
2. *If* $k \in \mathsf{k}$ *and* $(v, w) \in R$, $(kv, kw) \in R$.
3. *If* $(v, w), (v', w') \in R$, $(v + v', w + w') \in R$.

The vector spaces V and W may also be zero-dimensional. In this case, they contain only one element which we chose to call $*$. Since linear relations are nothing but sets, it is possible to compare them in terms of set inclusion. For example, any linear relation R must lie between the relation containing only the origin and the relation containing every possible pair of elements, namely $\{(0, 0)\} \subseteq R \subseteq \mathsf{k} \times \mathsf{k}$. Relational composition is defined as follows.

Definition 3 (Relational Composition). *Given two relations $R \subseteq U \times V$ and $S \subseteq V \times W$, the composition $R; S$ is the set of all pairs (u, w) such that there exists a v where $(u, v) \in R$ and $(v, w) \in S$. More precisely, it is the set $\{(u, w) \mid \exists v((u, v) \in R \wedge (v, w) \in S)\}$.*

It is not hard to prove that this composition preserves linear relations. A linear transformation $T : U \to V$ can be tought as the linear relation $\{(x, Tx) \mid x \in U\}$. Also, a subspace U can be thought as the linear relation $\{(*, u) \mid u \in U\}$. With this, one can see that relations are a generalization of these two concepts. Various linear algebra ideas seem to have nice relational representations.

A good example is the exchange lemma.

Lemma 4 (Exchange Lemma). *Let $\{\beta_i\}_{1 \leq i \leq b}$ and $\{\alpha_i\}_{1 \leq i \leq a}$ be two lists of vectors. Let $\{\beta_i\}$ be linearly independent. Then, if span$(\{\beta_i\}) \subseteq$ span$(\{\alpha_i\})$, $b \leq a$.*

This is the usual formulation of the Exchange Lemma, and intuitively it states that the length of any linearly independent list of vectors is less than or equal to the length of any spanning list of vectors. It is possible to look at the list of linearly independent vectors $\{\beta_i\}_{1 \leq i \leq b}$ as an injective matrix B, where each column in the matrix is a vector in the list. Similarly, the list of vectors $\{\alpha_i\}_{1 \leq i \leq a}$ can be thought of as a matrix A. Then, the Lemma can be reformulated in matrix form.

Lemma 5 (Exchange Lemma Matrix Form). *Let $B_{k \times b}, A_{k \times a}$ be matrices. If Img$(B) \subseteq$ Img(A) and B is injective, then $b \leq a$.*

Now consider the Pigeonhole Principle.

Lemma 6 (Pigeonhole). *Let $f : B \to A$ be an injective function. Then, $|B| \leq |A|$, where $|\cdot|$ is the cardinality of a set.*

The Pigeonhole Principle is usually defined using an injective function f. But the hypothesis can be weakened to an injective *total relation* instead.

Lemma 7 (Pigeonhole Principle for Linear Relations). *Let $B_{k \times b}$, $A_{k \times a}$ be matrices. If the relation $B; A^{op}$ is total and injective, then $b \leq a$.*

Here, A^{op} is the "inverse" relation defined by $(u, v) \in A^{op} \iff (v, u) \in A$. Interestingly, the connection between this result and the Exchange Lemma isn't clear in the classical semantics. But by using linear relations this connection becomes clearer. In fact, Lemma 5 and Lemma 7 are equivalent. This comparison is here simply to show how the relational view can shed light on connections that are possibly unclear in the functional view. In the first Lemma, the two properties about A and B seem to be somewhat unrelated. However, in the second Lemma, they are properties about the same linear relation (and these properties are symmetric, as we shall see in future chapters).

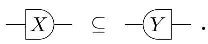
Let's give another example. Consider two linear transformations X, Y and the statement "X is the right inverse of Y". In classical semantics, we would need to write an expression using the identity matrix or write some kind of implication to represent the idea of right inverses.

$$YX = I. \tag{1.2}$$

In relational semantics, however, that idea can be merely expressed as

$$X \subseteq Y^{op}, \tag{1.3}$$

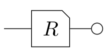
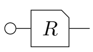
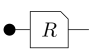
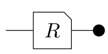
where Y^{op} is the "inverse" relation defined by $(u, v) \in Y^{op} \iff (v, u) \in Y$. Intuitively, we're saying that one relation is contained inside the inverse of the other. We still need to specify the "inverse" operation, though. In this case, it was named *op*. In a graphical language, the notion of a relation X having a inverse Y can be represented without the use of symbols (other than X and Y).


(1.4)

1.2.2.2 Why graphical

In terms of syntax, GLA uses string diagrams to represent linear relations. They are essentially drawings defined in a careful inductive way (SELINGER, 2011; BAEZ; ERBELE, 2015). They allow humans to use their visual intuition while not compromising formalism. Zanasi's axiomatization is also proven to be strong enough to prove any linear algebra result. The proofs using diagrams are graphical and treated as a series of formal rewriting rules, which already makes them a good place to use calculational proofs. Sometimes, these graphical proofs are shorter, more concise and more informative than their traditional symbolic counterparts. In short, many useful concepts in linear algebra can be written graphically in a completely point-free, inductive and formal way. Consider, for instance, the following definition and how the notions of image, domain and nullspace are symmetric statements about a linear relation. You're not supposed to understand this yet, but rather simply observe how the graphical language encapsulates different concepts into a single one (up to duality).

Definition 8. *Given a diagram R, we define*

1. *the nullspace of R:* $N(R) :=$  ,
2. *the multivalued part of R:* $Mul(R) :=$  ,
3. *the image of R:* $Ran(R) :=$  ,
4. *and the domain of R:* $Dom(R) :=$  .

1.2.2.3 Proofs in this language are calculational

Graphical Linear Algebra is a formal language, where axioms are rewriting rules for diagrams. This allows for calculational proofs, in the sense that every theorem is simply a series of rewriting steps. Thanks to that, algorithms and proofs are closely related. In later chapters, when we develop an algorithm for matrix multiplication, one will see that the steps map quite nicely to the usual classical algorithm. Lemma 1 will also be proven in this new approach and, as opposed to the wordy proof given in this chapter, it will have a fully calculational argument.

This approach, however, has some disadvantages. First of all, The Interacting Hopf Algebra's system has more than forty axioms. Matrices are defined iteratively, instead of inductively. Also, as opposed to big works like Axler's, there are few proofs done in GLA. Given the potential this approach displays for linear algebra, this work aims to be a step forward in its growth. Our research question is as follows: how can we further develop graphical linear algebra to fix these problems?

1.3 SUMMARY OF THE WORK

1.3.1 Our contributions

We have three main contributions:

- We propose a new axiomatization for Graphical Linear Algebra, reducing the number of axioms from 33 to merely 8. The axioms are high-level, in the sense that the diagrams quantified in them can have arbitrary size. Instead of duality, they have double duality, meaning that every theorem we prove gives rise to other 3 free theorems, merely by applying 2 main relational symmetries to the statement. Also, we use inequalities to compare linear relations, which allows for shorter proofs in comparison to the approach using only equalities (ZANASI, 2015).

- We develop a recursive definition of matrices completely inside the graphical language. Originally, matrices were defined iteratively, which makes proofs of core algorithms such as matrix multiplication not calculational. Our definition allows for a recursive and calculational proof of the matrix multiplication algorithm.

- We make use of a single relational canonical decomposition to prove various linear algebra results that were not yet proven inside the graphical language. This decomposition is enough to derive in a few lines powerful results which, inside classical books, would take whole paragraphs to prove. Not only that, but the proofs are more connected. Seemingly unrelated theorems are revealed to simply be particular cases of a single relational result.

1.3.2 Overview of the chapters

Each of the chapters will correspond to one of our mentioned contributions. In the second chapter, we'll introduce graphical linear algebra using a new set of axioms. The majority of the chapter is spent proving propositions about the algebraic structure of GLA, and we finish by using this proposition to prove that our axioms are equivalent to the Interacting Hopf Algebra's original ones.

In the third chapter, we present a definition of matrices inside the graphical language. This has already been done (LAFONT, 2003; ZANASI, 2015), but with iterative data structures. In our work, the definition is inductive and corresponds to the two ways one goes about constructing a matrix: by using essentially a list of row vectors or a list of column vectors. Then, we prove various useful theorems about this structure and finish the chapter by proving that matrices are closed by composition. This, thanks to the fact that the proof is constructive and calculational, will naturally lead to an algorithm for matrix multiplication.

In the fourth chapter, we prove various useful characterizations, which describe concepts inside linear algebra in an elegant and unified way. We'll also make use of a relational canonical form to simplify most of the proofs. Finally, we'll conclude the chapter by proving various results which are included in most traditional linear algebra books. One will be able to see that, while some of these proof have extensive, wordy arguments inside classical works, the proofs presented in this chapter will often not go beyond the second line. We'll then conclude by giving an overview of everything that was done and what were the benefits achieved. We'll show some related research, and also discuss some possible future works and what we think are the prospects for graphical linear algebra in the near future.

2 RELATIONAL GRAPHICAL LINEAR ALGEBRA

In this chapter, we introduce the diagrammatic language known as Graphical Linear Algebra (GLA). It is a graphical language for Linear Algebra that uses constructive diagrams to reason about linear relations. Our diagrams are an instance of a particular class of *string diagrams* (SELINGER, 2010). They are rigorous mathematical objects and are defined inductively. We will explain how diagrams are constructed and outline basic principles of how to manipulate, and reason with them.

This language has its roots in category theory, particularly in the notion of strict symmetric monoidal categories. However, as it stands, Graphical Linear Algebra is a self-contained formal language, which means the reader is not expected to be familiar with categorical concepts. The rules inherited from category theory will manifest themselves in graphical deformations of diagrams, making them not only visually intuitive but also understandable even for readers with no experience with category theory. So we will not delve into these details.

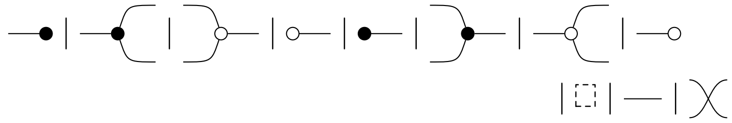
In this chapter, we will showcase our first contribution. The language will be presented using our first contribution, our axiomatization. This axiomatization was inspired by the set of axioms presented in the work Interacting Hopf Algebras (ZANASI, 2015). While the original work had 33 axioms and was lacking proper use of linear algebraic symmetries, ours has merely 8 axioms, two of which encode the two main symmetries of relational linear algebra.

We'll begin the chapter by presenting the graphical language and explaining its syntax and semantics. Then, we'll prove some results that will be necessary for our main theorem: our axiomatization is equivalent to the one presented in the work Interacting Hopf Algebras (ZANASI, 2015).

2.1 SYNTAX OF GLA

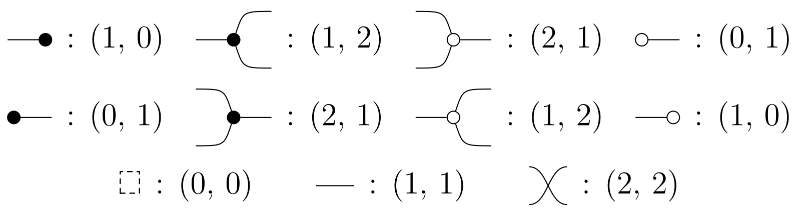
Our starting point is the simple grammar for the language of diagrams. It consists of some base cases and some operations.

$c, d \quad ::=$



$\mid c \,;\, d \mid c \oplus d \quad (2.1)$

The first line of (2.1) consists of eight constants that we refer to as *generators*. Although they are given in diagrammatic form, for now we can consider them as mere symbols. Already here we can see two fundamental symmetries that are present throughout this work: for every generator there is a "mirror image" generator (inverted in the "y-axis"), and for every generator there is a "colour inverse" generator (swapping black with white).

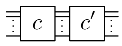
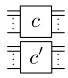


$$\frac{c:(k_1,k_2)\qquad d:(k_2,k_3)}{c\,;d:(k_1,k_3)}\qquad\qquad \frac{c:(k_1,l_1)\qquad d:(k_2,l_2)}{c\oplus d:(k_1{+}k_2,l_1{+}l_2)}$$

Fig. 2 – The sorting rules.

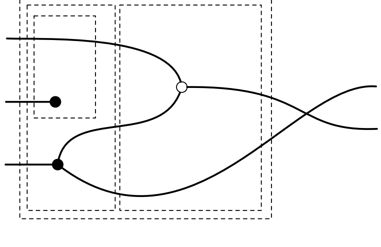
Thus, it suffices to note that there are generators , and that the set of generators is closed with respect to the two symmetries. We shall discuss the formal semantics of the syntax in Section 2.2, but it is useful to already provide some intuition at this point. It is helpful to think of , as gates that, respectively, *discard* and *copy* the value on the left, and of , as *zero* and *add* gates. The mirror-image versions have the same intuitions, but from right to left.

The second line of (2.1) contains some structural terms and two binary operations for composing diagrams. The term is the empty diagram, — is a wire and allows swapping the order of two wires. The two binary operations ; and $\oplus$ allow us to construct diagrams from smaller diagrams. Indeed, the diagrammatic convention is to draw $c\,; c'$ (or $c' \cdot c$) as series composition, connecting the "dangling" wires on the right of c with those on the left of c', and to draw $c \oplus c'$ as c stacked on top of c', (parallel composition), that is:

$c\,; c'$ is drawn  and $c \oplus c'$ is drawn  .

The simple sorting rules of Fig. 2 count the "dangling" wires and thus ensures that the diagrammatic convention for ; makes sense. A *sort* is a pair of natural numbers (m, n): m counts the dangling wires on the left and n those on the right. We consider only those terms of (2.1) that have a sort. It is not difficult to show that if a term has a sort, it is unique.

Example 9. *Consider the term* $\big(\big((— \oplus —\!\bullet) \oplus\ \ \big)\ ;\ \big(\ \ \oplus —\big)\big)\ ;$. *It has sort* $(3, 2)$ *and the diagrammatic representation is*



where the dashed-line boxes play the role of disambiguating associativity of operations.

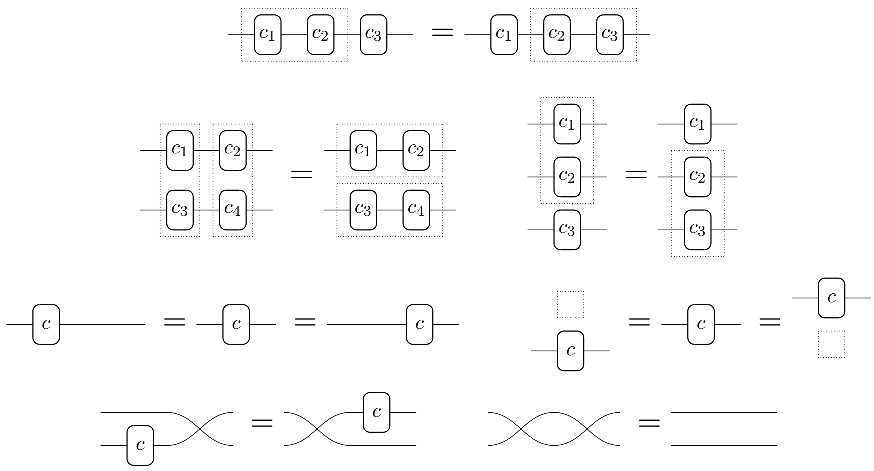
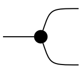
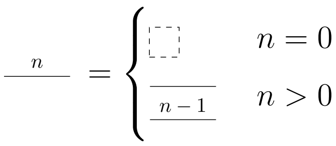


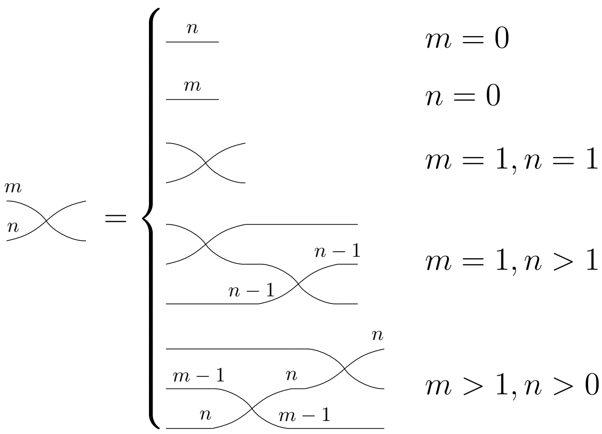
Fig. 3 – Diagrammatic laws. Sort labels are omitted for readability.

We can also give recursive definitions. For example, the following generalization of  will be useful in subsequent sections. The idea is that the generators are being generalized from their version with a single dangling wire, to multiple dangling wires, which we are calling the n-wired version. Similar constructions can be given for the other generators in (2.1).

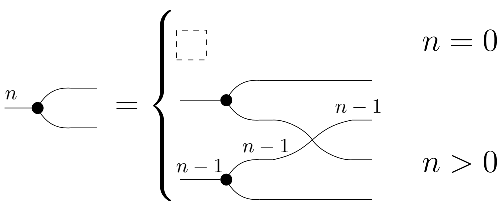
Definition 10 (n-wired Generators)**.**



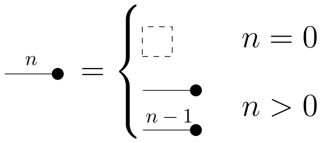
(2.2)



(2.3)



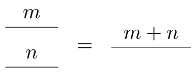
(2.4)



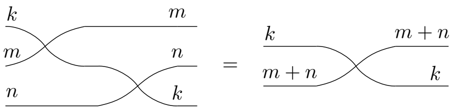
(2.5)

The next proposition follows via induction.

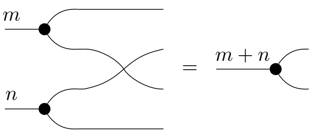
Proposition 11. *The types defined in definition 10 are closed by the following operations.*



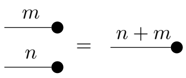
(2.6)



(2.7)



(2.8)



(2.9)

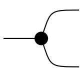
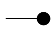
2.2 SEMANTICS OF GLA

We use GLA as a diagrammatic language for linear algebra. Different from traditional developments, the graphical syntax has a relational meaning. The central mathematical concept is that of a *linear relation* (ARENS et al., 1961; LANE, 1961; CODDINGTON, 1973; CROSS, 1998). Linear relations are also sometimes called *additive* relations (LANE, 1961) in the literature.

Definition 12. *Fix a field* k *and* k*-vector spaces* V, W. *A linear relation from* V *to* W *is a set* $R \subseteq V \times W$ *that is a subspace of* $V \times W$, *considered as a* k*-vector space. Explicitly this means that:*

1. $(0, 0) \in R$.
2. *If* $k \in \mathsf{k}$ *and* $(v, w) \in R$, $(kv, kw) \in R$.
3. *If* $(v, w), (v', w') \in R$, $(v + v', w + w') \in R$.

Now, we give the interpretation of the string diagrams of our language GLA as linear relations. Since they are defined as a free grammar, it is enough to define the interpretation for the generators. The rest follows via induction.

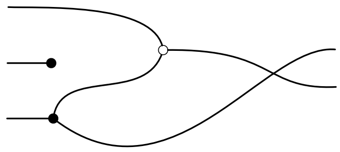
 $\longmapsto \left\{\left(x, \begin{pmatrix} x \\ x \end{pmatrix}\right) \mid x \in \mathsf{k}\right\} \subseteq \mathsf{k} \times \mathsf{k}^2$  $\longmapsto \{(x, \star) \mid x \in \mathsf{k}\} \subseteq \mathsf{k} \times \mathsf{k}^0$

$$\longmapsto \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, x+y \right) \mid x, y \in \mathsf{k} \right\} \subseteq \mathsf{k}^2 \times \mathsf{k} \qquad \longmapsto \{(\star, 0)\} \subseteq \mathsf{k}^0 \times \mathsf{k}$$

The generators , , and map (respectively) to the opposite relations of the above, as hinted by the symmetric diagrammatic notation.

Intuitively, the *black structure* in diagrams refers to copying and discarding – is *copy* and is *discard.* The *white structure* refers to addition in k – is *add* and is *zero*.

Example 13. *Consider the term of Example 9, reproduced below without the unnecessary dashed-line box annotations.*



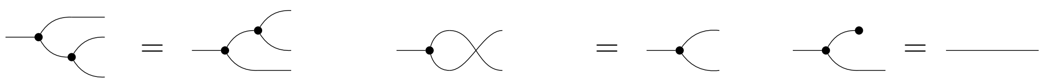
Its semantics is the (functional) relation $\left\{ \left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}, \begin{pmatrix} z \\ x+z \end{pmatrix} \right) \mid x, y, z \in \mathsf{k} \right\}$.

2.2.1 The 8 axioms of GLA

In this section, we identify the (in)equational theory of GLA. We will present all 8 necessary axioms. The last one, however, is about scalars, and it requires a proper definition of them before we can precisely state it. It will be included in this subsection for completeness, but the precise meaning of that axiom will be postponed to Section 2.3, when we construct a graphical definition of numbers.

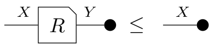
2.2.1.1 The commutative comonoid axiom

Axiom 1 (Commutative comonoid)**.** *The copy structure satisfies the equations of commutative comonoids, that is,* associativity*,* commutativity *and* unitality*, as given below:*

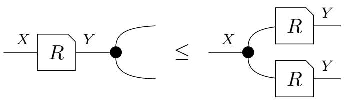


2.2.1.2 The 4 inequality axioms

Instead of a purely equational axiomatization, we find the use of inequalities very convenient in proofs. The development of inequality axioms is one of our contributions (PAIXÃO; SOBOCIŃSKI, 2020). We state four axioms below where R ranges over arbitrary GLA string diagrams. Given that R can have arbitrary sort, the axioms make use of Definition 10.

Axiom 2 (Discard). 

The axiom is sound with respect to the semantics. The left-hand-side denotes $LHS = \{(x, \star) \mid \exists y.\, (x, y) \in R\}$, while the right hand side denotes the relation $RHS = \{(x, \star) \mid \top\}$. Clearly $LHS \subseteq RHS$.

Axiom 3 (Copy). 

The left-hand-side is the relation $LHS = \{(x, \binom{y}{y}) \mid (x, y) \in R\}$, while the right-hand-side is $RHS = \{(x, \binom{y}{y'}) \mid (x, y), (x, y') \in R\}$ and soundness is again easy.

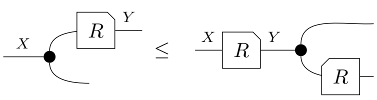
Proposition 14. $\{(x, \binom{y}{y}) \mid (x, y) \in R\} \subseteq \{(x, \binom{y}{y'}) \mid (x, y), (x, y') \in R\}$

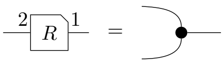
Proof.

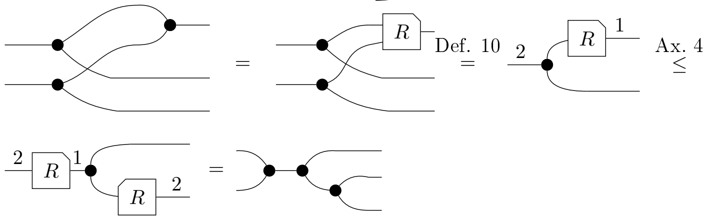
$$
\begin{aligned}
& \{(x, \tbinom{y}{y}) \mid (x, y) \in R\} \\
= \quad & \{ \text{ Substitution } \} \\
& \{(x, \tbinom{y}{y'}) \mid (x, y), (x, y') \in R, y = y'\} \\
\subseteq \quad & \{ \text{ Trivial } \} \\
& \{(x, \tbinom{y}{y'}) \mid (x, y), (x, y') \in R\}
\end{aligned}
$$

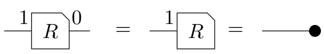
□

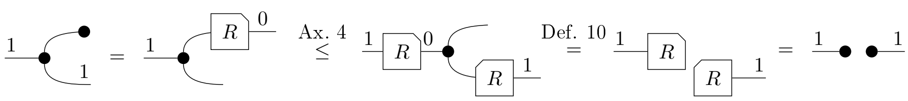
The diagrammatic notation for R allows us to represent the mirror image symmetry intuitively. In the following diagram, the bottom right occurrence of R is the reflected diagram—which we write R^o in linear syntax.

Axiom 4 (Wrong Way). 

For example, let , then Axiom 4 says that



For another example, let , then Axiom 4 says that



The soundness of Axiom 4 is very similar to the soundness of Axiom 3.

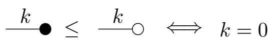
Proposition 15. $\{(x, \binom{y}{x}) \mid (x, y) \in R\} \subseteq \{(x, \binom{y}{x'}) \mid (x, y), (x', y) \in R\}$

Proof.

$$
\begin{aligned}
& \{(x, \tbinom{y}{x})) \mid (x, y) \in R\} \\
= \quad & \{ \text{ Substitution } \} \\
& \{(x, \tbinom{y}{x'}) \mid (x, y), (x', y) \in R, y' = x\} \\
\subseteq \quad & \{ \text{ Trivial } \} \\
& \{(x, \tbinom{y}{x'}) \mid (x, y), (x', y) \in R\}
\end{aligned}
$$

$\square$

The relationship between Axioms 3 and 4 is worth explaining. In tandem, they capture a topological intuition: a relation that is adjacent to a black node can "commute" with the node, resulting in a potentially larger relation. In Axiom 3 the relation approaches from the left. In Axiom 4 the relation comes from the "wrong side", but it can still commute with the node to obtain a larger relation, but one must take care of the lower right wire that "curves around". Lastly, there's one axiom that states there is only one zero-dimensional space.

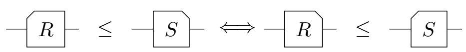
Axiom 5 (Singularity)**.** 

Notice that the opposite inequality is always true, so this axiom actually states that when $k = 0$ we have *equality* between discard and zero. In other words, the entire space is the origin.

2.2.1.3 The 2 symmetry axioms

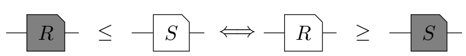
In Section 2.1, we saw that the generators of GLA are closed under two symmetries: the "mirror-image" $(-)^o$ and the "colour-swap". Henceforth we denote the colour swap symmetry by $(-)^\dagger$. Clearly, for any R, we have $R^{oo} = R$ and $R^{\dagger\dagger} = R$. We are now ready to state two more axioms.

Axiom 6 (Converse)**.** *$R^o \leq S \iff R \leq S^o$, or in diagrams:*



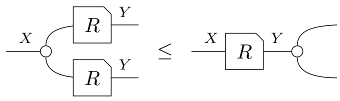
In particular, the mirror-image symmetry is *covariant*: if $R \leq S$ then $R^o \leq S^o$. Soundness is immediate.

Axiom 7 (Colour inverse)**.** *$R^\dagger \leq S \iff R \geq S^\dagger$, or in diagrams:*



This is the most difficult Axiom to understand. The statement might seem simple, but it is saying something quite sophisticated. Color-swapping a diagram is swapping the black and white structures. That means swapping a sum generator for a co-copy generator, swapping a discard generator for a co-zero generator, and so on. This Axiom is stating that this swapping operation almost preserves inequality. It says that the colour-swap symmetry is *contravariant*: if $R \leq S$ then $S^\dagger \leq R^\dagger$. The reader should pause at this moment to think about it and make sure this Axiom is well understood. To help with that, we will proceed carefully through the soundness proof (which will also include some examples).

Here soundness is not as obvious: one way is to show that the inequalities of Axioms 2, 3 and 4 reverse when it is the white structure that is under consideration. For example, "inverting the colors of Axiom 2":

Proposition 16 ("Axiom $3^\dagger$"). 

Proof. Axiom 7 applied to Axiom 3. □

Soundness can be shown:

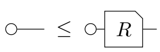
Proposition 17. $\{(x_1 + x_2, \binom{y_1}{y_2}) \mid (x_1, y_1), (x_2, y_2) \in R\} \subseteq \{(x, \binom{y_1}{y_2}) \mid (x, y_1 + y_2) \in R\}$

Proof.

$$
\begin{aligned}
& \{(x_1 + x_2, \tbinom{y_1}{y_2}) \mid (x_1, y_1), (x_2, y_2) \in R\} \\
\subseteq \quad & \{\ R \text{ is a linear relation (Definition 12) } \} \\
& \{(x_1 + x_2, \tbinom{y_1}{y_2}) \mid (x_1 + x_2, y_1 + y_2) \in R\} \\
\subseteq \quad & \{\ x = x_1 + x_2\ \} \\
& \{(x, \tbinom{y_1}{y_2}) \mid (x, y_1 + y_2) \in R\}
\end{aligned}
$$

□

In this example, we're showing that the color-swap operation is in fact contravariant in Axiom 3. Intuitively, the idea is the following: when you slide a single relation through a copy, you get a bigger relation; when you slide a single relation through a sum, you get a smaller relation. Another example is "inverting the colors of Axiom 2".

Proposition 18 ("Axiom $2^\dagger$"). 

Proof. Axiom 7 applied to Axiom 2. □

Soundness can be shown:

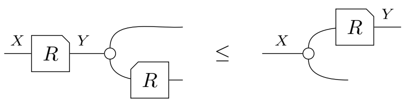
Proposition 19. $\{(*, 0)\} \subseteq \{(*, x) \mid (0, x) \in R\}$

Proof.

$$
\begin{aligned}
& \{(*, 0)\} \\
\subseteq \quad & \{\ R \text{ is a linear relation (Definition 12) } \} \\
& \{(*, 0) \mid (0, 0) \in R\} \\
\subseteq \quad & \{\ x = 0\ \} \\
& \{(*, x) \mid (0, x) \in R\}
\end{aligned}
$$

□

In this case, intuitively speaking, we're saying that when you "produce"a relation from a discard, the resulting relation is smaller, but when you "produce"a relation from a zero, the resulting relation is bigger.

Proposition 20 ("Axiom 4†"). 

Proof. Axiom 7 applied to Axiom 4. □

Soundness can be shown:

Proposition 21. $\{(x_1, \begin{pmatrix} y_1 - y_2 \\ x_2 \end{pmatrix}) \mid (x_1, y_1), (x_2, y_2) \in R\} \subseteq \{(x_1, \begin{pmatrix} y \\ x_2 \end{pmatrix}) \mid (x_1 - x_2, y) \in R\}$

Proof.

$$
\begin{aligned}
& \{(x_1, \begin{pmatrix} y_1 - y_2 \\ x_2 \end{pmatrix}) \mid (x_1, y_1), (x_2, y_2) \in R\} \\
\subseteq \quad & \{\ R \text{ is a linear relation (Definition 12) } \} \\
& \{(x_1, \begin{pmatrix} y_1 - y_2 \\ x_2 \end{pmatrix}) \mid (x_1 - x_2, y_1 - y_2) \in R\} \\
\subseteq \quad & \{\ y = y_1 - y_2\ \} \\
& \{(x_1, \begin{pmatrix} y \\ x_2 \end{pmatrix}) \mid (x_1 - x_2, y) \in R\}
\end{aligned}
$$

□

This one is similar to Proposition 17. Again, we're proving that when a single relation slides through a sum, the resulting relation is smaller, but when a single relation slides through a copy, the resulting relation is bigger.

It is easy to see that the color-swap and converse operations commute.

Proposition 22. $(R^o)^\dagger = (R^\dagger)^o$

By the above proposition, we can define the transpose operation.

Definition 23. $R^t := (R^o)^\dagger = (R^\dagger)^o$.

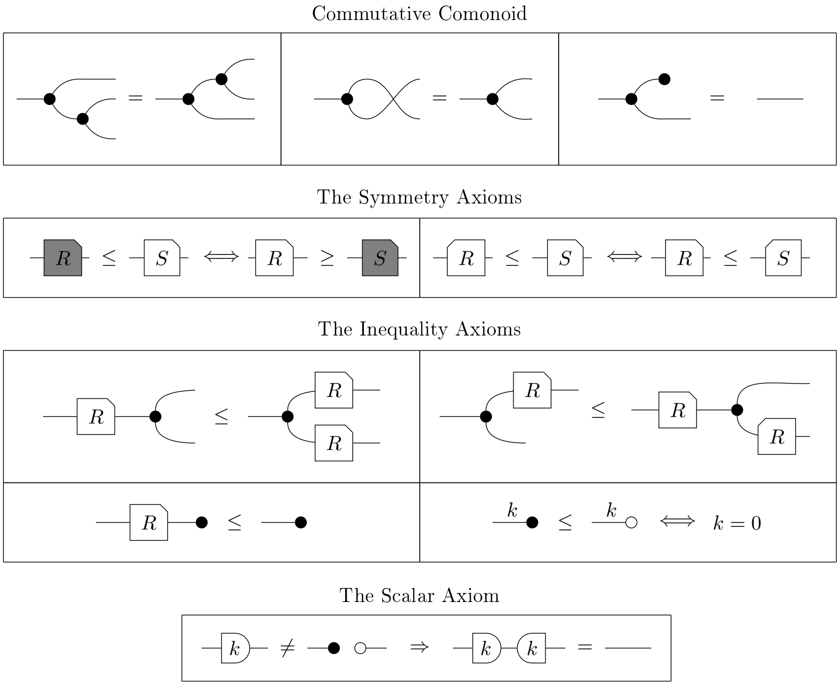
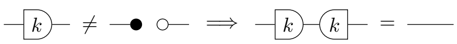


Fig. 4 – All the axioms of GLA.

2.2.1.4 The scalar axiom

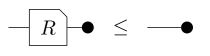
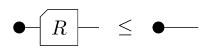
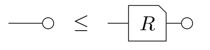
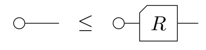
As we stated at the beginning of this subsection, the last axiom is about scalars. In this work, the concept of scalars is not axiomatic but instead constructed inductively. This will be done in the middle of Section 2.3, so until then, we cannot precisely state what the scalar axiom means. However, the intuition may be clear for anyone with a basic understanding of linear algebra. The axioms state that a particular class of diagrams, called scalars, are invertible when they aren't zero.

Axiom 8. 

Having identified all the axioms of this work's Graphical Linear Algebra, we can compile all of them in Figure 4.

2.2.2 Prove one, get three theorems for free!

We will explore the symmetries of Axioms 5-6 throughout the work. When we prove a result, we will usually assume we can use any of the other three theorems (its converse theorem, its colour inverse theorem, and its transpose theorem) afterwards just by referencing the original result. For example when we use, in a proof, the colour inverse of Axiom 3 (Proposition 16) we will simply denote Ax. 3. As an example, here are the four versions of Axiom 2.

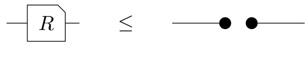
Axiom 2	Axiom $2^\dagger$
	
Axiom 2^o	Axiom $2^{o\dagger}$
	

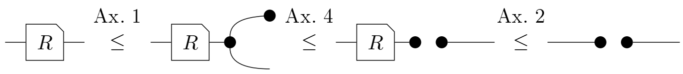
2.3 EQUIVALENCY TO INTERACTING HOPF ALGEBRA'S AXIOMS

Graphical Linear Algebra wasn't born with the set of axioms presented in this work. It was originally formalized with a set of axioms represented in Fig. 6. It was independently developed by John C. Baez (BAEZ; ERBELE, 2015) and Fabio Zanasi in his thesis known as Interacting Hopf Algebras (BONCHI; SOBOCIŃSKI; ZANASI, 2017b; ZANASI, 2015). One of the contributions of this work is establishing a new, more compact set of axioms that retains the point-freeness, but also manages to expose the dualities and symmetries of linear algebra in a more explicit, relational way. Our inspiration came from the realization that some useful properties of the graphical language can be represented in a high-level style. For instance, while Interacting Hopf Algebra's axioms are defined using the set of generators (ZANASI, 2015), our approach uses their generalized n-wired versions. That, alone, serves the purpose of shortening the amount of proofs required to achieve meaningful linear algebra results. Also, we've noticed that the original set of axioms could be considerably shortened by making use of relational inequality and the color and mirror symmetries defined in Axioms 6 and 7. With these changes, we were able to shrink the number of axioms from 33 to merely 8.

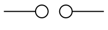
In Zanasi's thesis, a completeness result is proven, which states that his axiomatization can derive any linear algebra result. For the rest of this section, we'll spend some time proving that our axiomatization can simulate his, therefore, concluding that our axioms are also strong enough to derive any linear algebra result. In this section we identify some of the algebraic structure that is a consequence of the six axioms so far and finish the chapter by proving that our approach can simulate Zanasi's.

First, we need to prove that, given a diagram sorting (m, n), there is a largest element (contains every other relation with the same number of dangling wires) and a smallest element (is contained in every other relation with the same number of dangling wires), which we are calling respectivelly top and bottom elements.

Proposition 24 (Top Element). 

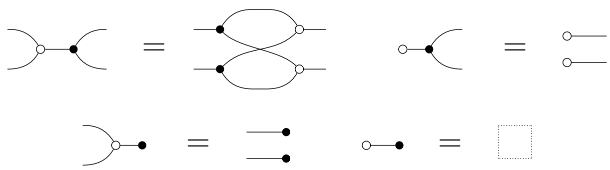
Proof. 

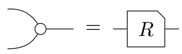
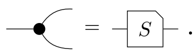
In the second step, we are using the second example described right after Axiom 4. □

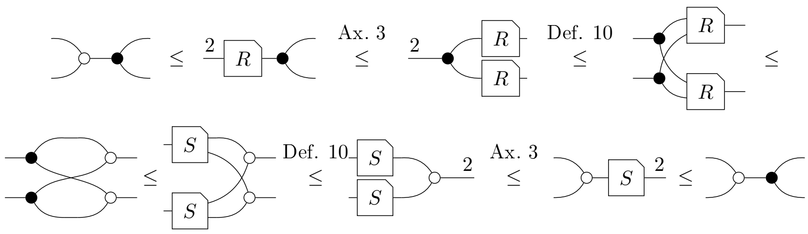
Dually, from Axiom 7 we obtain that  is the bottom element.

Now, we'll prove a proposition regarding the way the white and black structures interact. The following proof was inspired by (BAEZ, 2010), which is one of the main inspirations for this chapter.

Proposition 25 (Bialgebra)**.**



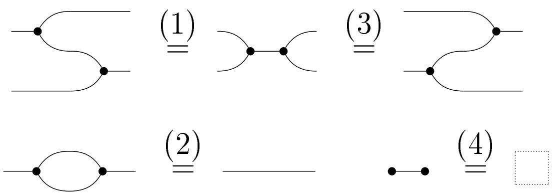
Proof. Let  and . We have

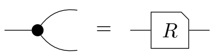


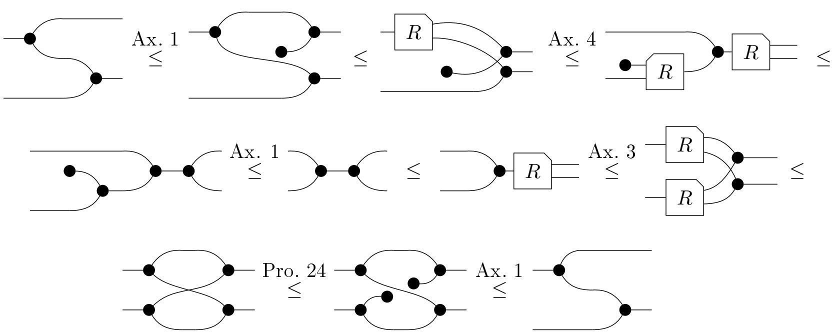
We use Axiom 3 in the first inequality and its color inverse in the second inequality. Note that while using Axiom 3 in the proof, we must use Definition 10. The other derivations are similar. $\square$

After seeing how the black structure interacts with the white structure, we'll now see how the white structure (or the black structure) interacts with itself.

Proposition 26 (Frobenius)**.** *The following equations hold.*

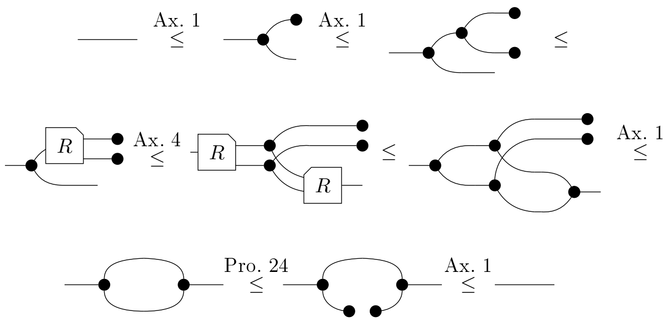


Proof. Let . Then, we have the proof of Equation (1):



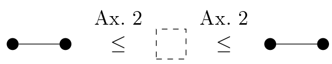
In the second step of the above proof, we are using the first example described right after Axiom 4.

Proof of Equation (2):



Equations (3) can be proven similarly.

Finally, proof of equation (4). This one may seem counterintuitive since we're using Axiom 2 on a diagram with zero wires on one of the sides.

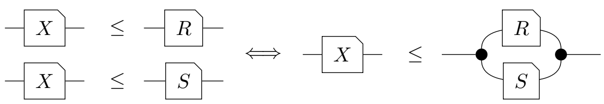


□

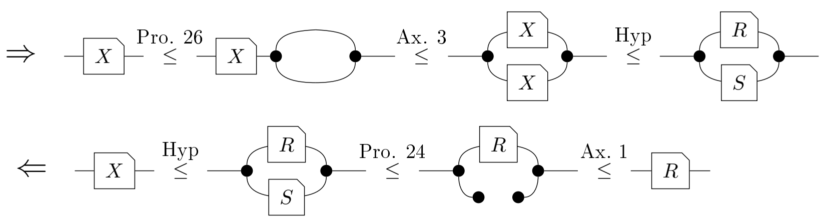
Notice that there is no indication of the number of wires in this proof. Since our axioms assume diagrams with any number of wires, most of the time it is unnecessary to specify it. So from now on we'll omit the number of wires most of the time.

With the previous proposition, we can show now that the set of (m, n) string diagrams has also a meet and join operation, which is essentially an analogous notion to the intersection and union of sets.

Proposition 27 (Universal property of intersection)**.**



Proof.

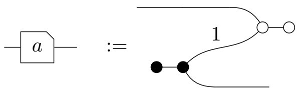


The other inequality follows similarly. □

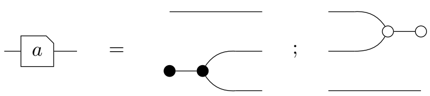
By duality of Axiom 7, we obtain the join. In the semantics, meet is intersection $R \cap S$, while join is the smallest subspace containing R and S: i.e. the subspace closure of the union $R \cup S$, usually denoted $R + S$.

2.3.1 The Antipode: how to write negative numbers

One of the most surprising facts about this presentation of linear algebra is the plethora of concepts derivable from the basic components of copying and adding (black and white) structures. This includes the notion of *antipode* a, i.e. -1. Informally if the left wire of the antipode carries a value, then the right wire carries its opposite. The following propositions and definitions were inspired by (ERBELE,).

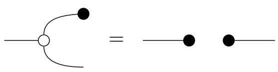
Definition 28 (Antipode). 

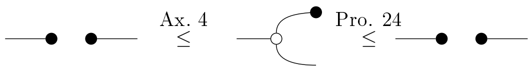
The semantics of this diagram is quite interesting. Informally speaking, since there is a zero attached to the right of the sum, the two inputs must be additive inverses of each other. The copy is merely picking that additive inverse and spitting it as the output.



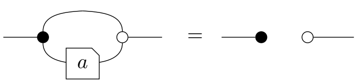
$$(x, -x) = \left(x, \begin{pmatrix} x \\ y \\ y \end{pmatrix} \right) ; \left(\begin{pmatrix} x \\ -x \\ y \end{pmatrix}, y \right)$$

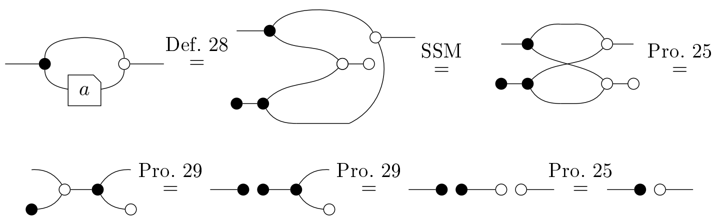
Before proving properties of the antipode, we prove a useful proposition.

Proposition 29. 

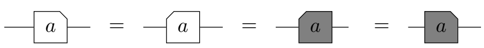
Proof. 

In the first step, we are using essentially the same idea from the second example of Axiom 4. □

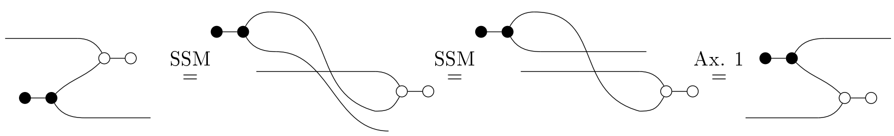
Proposition 30. *(Hopf)* 

Proof. 

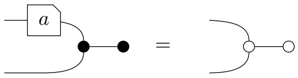
□

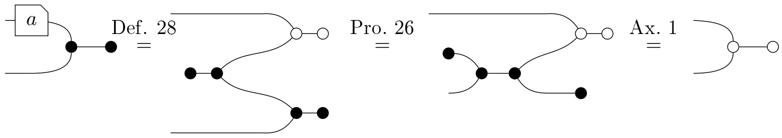
Proposition 31. 

($a = a^o = a^\dagger = a^t$)

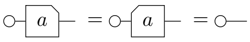
Proof. 

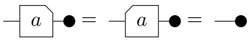
In the last step we are using the commutative property twice. All other equations are proven similarly. □

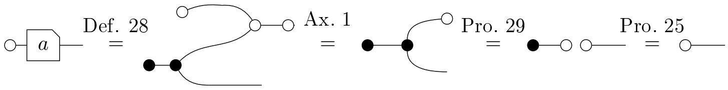
Proposition 32. 

Proof. 

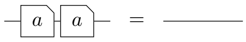
□

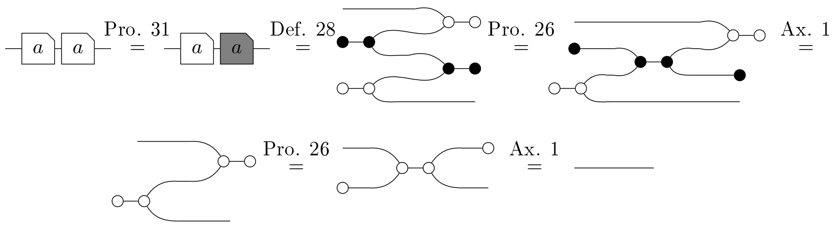
Proposition 33. 

and 

Proof. 

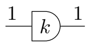
By Proposition 31 the other three symmetric properties have symmetrical proofs. □

Proposition 34. 

Proof. 

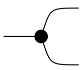
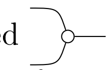
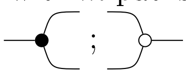
□

2.3.2 Building the scalars, inductively

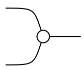
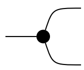
Linear algebra is often understood as parametric over a fixed field k, say the real numbers or complex numbers. Given that string diagrams are finite, we have no hope of expressing arbitrary real numbers for cardinality reasons. In fact, there is a principled way of introducing the elements of any field into the diagrammatic language. In summary, an additional generator 

is added for each $k \in \mathsf{k}$, and the axioms in Figure 3, recalled from (BONCHI; SOBOCIŃSKI; ZANASI, 2017b), ensure that the field operations are compatible with the diagrammatic algebra.

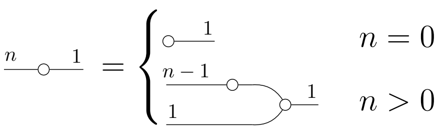
To develop the diagrammatic theory of rational numbers, we first focus on the *natural numbers* as particular GLA diagrams of sort (1, 1).

A natural number m in GLA is a (1, 1) diagram which, roughly speaking, begins with iterated  on the left, and ends with iterated  on the right. The idea is that m corresponds to a diagram with m paths from the left boundary to the right: 1 is identified with the identity, 2 with 

and so on.

We first define recursively-defined types of iterated  and .

Definition 35.



(2.10)

Diagrams that correspond to natural numbers can now be defined as follows.

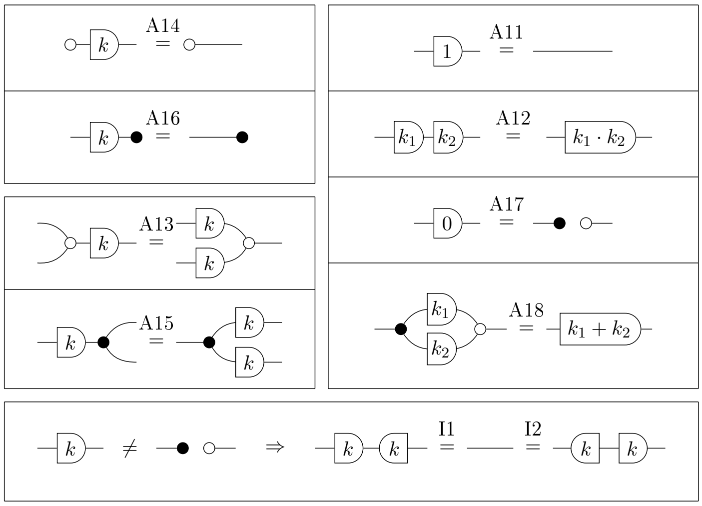
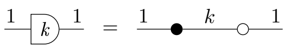
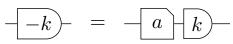


Fig. 5 – Parametrising over a field k.

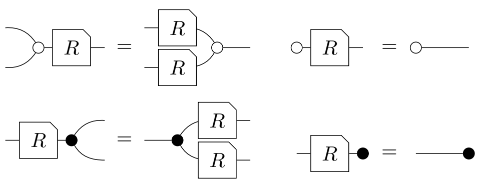
Definition 36. 

Using this, one can easily construct a new definition for the whole type of integers by using the antipode.

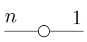
 (2.11)

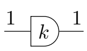
To elaborate on the properties of diagrammatic integers, we need to introduce some concepts. The next proposition states that a class of diagrams called Maps (Definition 61) enjoys a distributivity property with respect to adding and copying.

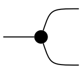
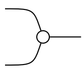
Proposition 37. *If R is a Map, the following equations hold:*

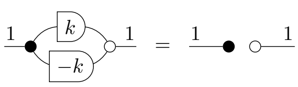


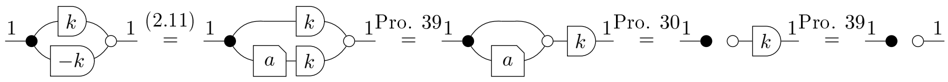
The following propositions can be easily proved by induction.

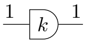
Proposition 38.  *is a Map.*

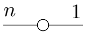
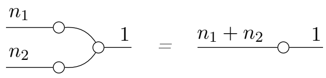
Proposition 39.  *is a Map.*

With this in place we can now make use of Proposition 37 in proofs concerning natural numbers. The fact that numbers are maps makes them particularly easy to handle, since we can just "slide" them through  and .

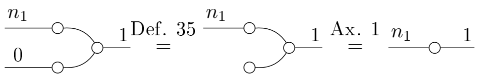
Proposition 40. 

Proof.  □

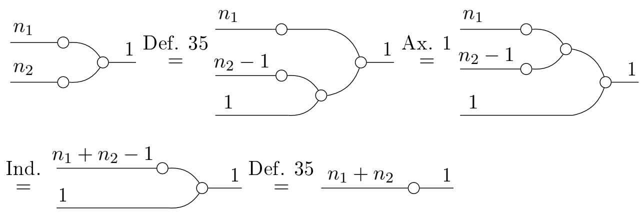
The type  is closed by addition and multiplication, which have a natural interpretation in the graphical language.

Proposition 41 ( closed by Sum). 

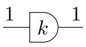
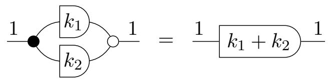
Proof. First, we do the base case where $n_2 = 0$.

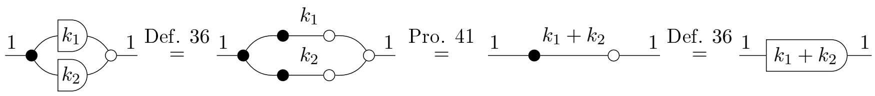


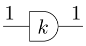
Now we do the inductive step, taking $n_2 > 0$.



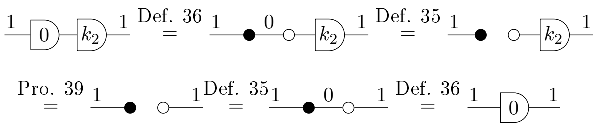
□

Proposition 42 ( closed by addition). 

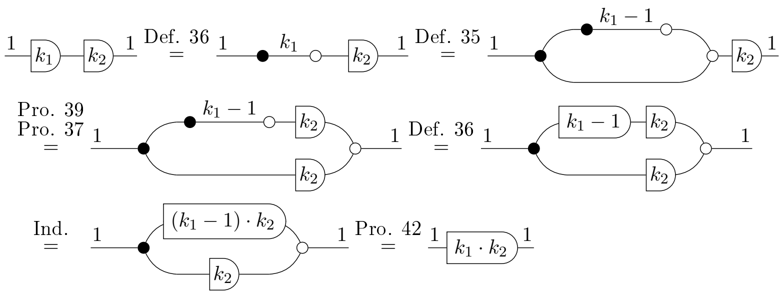
Proof.  □

Proposition 43 ( closed by composition). 

Proof. This time, we are doing induction on k_1. So, first, in the base case, we take $k_1 = 0$.



In the inductive step, we take $k_1 > 0$.



□

By using Propositions 30 and 34, we could also show that integers (as in equation (2.11)) are also closed by addition and composition.

The complete set of equations is known as the theory of Interacting Hopf Algebras (BONCHI; SOBOCIŃSKI; ZANASI, 2017b). In fact, given the previous results of this paper, we have the main result of this section. The following Theorem establishes that our axiomatization can simulate the Interacting Hopf Algebra's axiomatization.

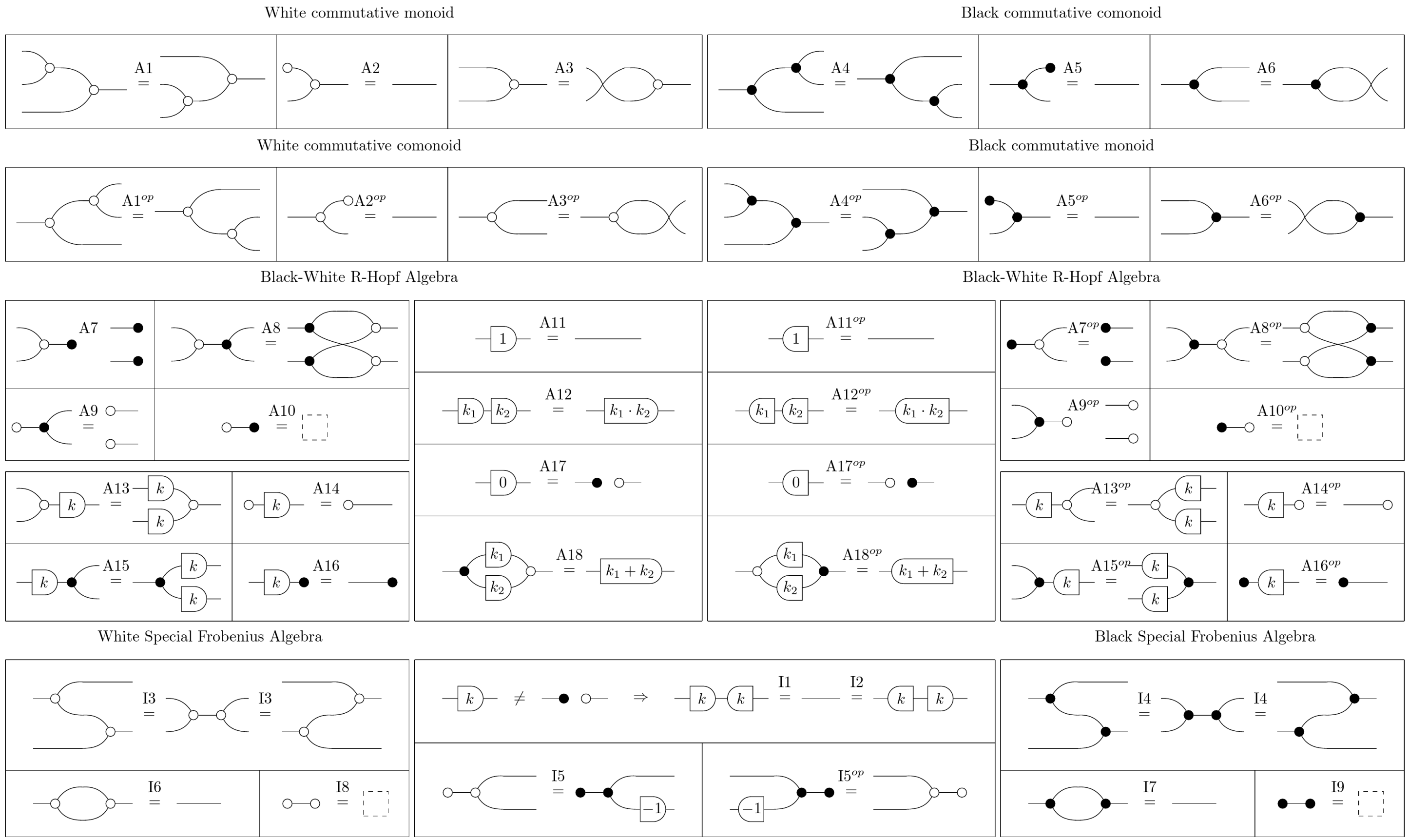


Fig. 6 – Axiom of Interacting Hopf Algebras (Image adapted from (ZANASI, 2015)).

Theorem 44. *Axioms 1-8 suffice to derive all of the algebraic structure of Interacting Hopf (IH) algebras(BONCHI; SOBOCIŃSKI; ZANASI, 2017b; ZANASI, 2015), reproduced here in Figure 6. In fact, they are also sound in that theory.*

Proof.

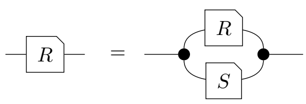
- (Black Monoid) Axiom 1 $\implies A4, A5$, and $A6$
- (White Monoid) Axiom 1 and Axiom 7 $\implies A1, A2$, and $A3$
- (Bialgebra) Proposition 25 $\implies A7, A8, A9$, and $A10$.
- (Numbers) Definition 36 $\implies A11$ and $A17$.
- (Numbers closed by composition) Proposition 43 $\implies A12$.
- (Numbers are Maps) Proposition 37 and Proposition 39 $\implies A13, A14, A15$ and $A16$.
- (Numbers closed by addition) Proposition 42 $\implies A18$.
- (Invertible numbers) Axiom 8 $\implies I1$.
- (Invertible numbers) Axiom 8^t $\implies I2$.
- (Black Frobenius) Proposition 26 $\implies I4, I7$ and $I9$.
- (White Frobenius) Proposition 26 and Axiom 7 $\implies I3, I6$ and $I8$.
- (Antipode properties) Proposition 32 $\implies I5$

All the converse axioms are proven by simply applying Axiom 6 to their normal versions.

$\square$

The above result proves our initial goal in this chapter. We've successfully demonstrated that our axiomatization can simulate the Interacting Hopf Algebras' axiomatization. It was proven in the original work that the latter is strong enough to derive any linear algebra result (BONCHI; SOBOCIŃSKI; ZANASI, 2017b, Theorem 6.4). This then implies that our axiomatization is also strong enough. This is a key result in the research, since now we know that these 8 axioms, despite being compact and symmetric, are not lacking in power in comparison to previous axiomatizations.

As a sidenote, it is possible to simulate the inequality axioms in the purely equational theory of Interating Hopf Algebras. The statement $R \subseteq S$ can be stated as



,

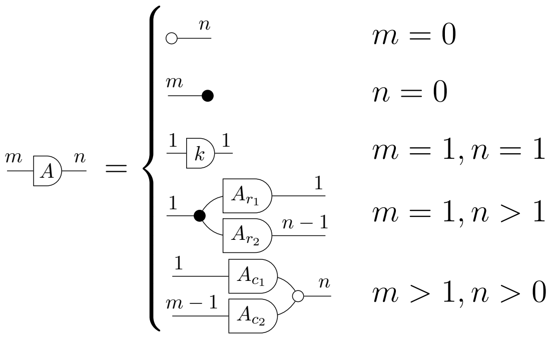
which intuitively states that $R = R \cap S$. It is also possible to simulate the inequality using the white structure (sum), instead of the black structure (copy). That choice is one way to create the asymmetry present in the inequality axioms.

Now, we managed to guarantee that any linear algebra proof can be achieved with this approach. However, we still don't know how to produce these proofs. We still need to lay down a good foundation in which these proofs can be developed. In particular, we still need a way of constructing matrices, and if possible, inductively. Informally speaking, matrices provide a nice inductive structure in which proofs can be built. We'll also see in later chapters that any linear transformation can be represented by matrices and co-matrices (which are matrices "reversed"by the † operator). So, in the next chapter, we'll carefully construct inside the graphical language an inductive definition of matrices and prove some useful properties about them.

3 MATRICES IN GLA

In this chapter, we continue the recursive/inductive style of definitions and proofs to study the type of matrices in GLA. One of our contributions is providing a completely diagrammatical definition of matrices, together with a matrix multiplication algorithm. A matrix, diagrammatically, is intuitively a "generalised" scalar of Section 2.3.2: like scalars, these are diagrams built up of black structure followed by white structure. However, we do not consider merely (1, 1) diagrams and allow an arbitrary number of wires.

Definition 45 (Mat)**.**



(3.1)

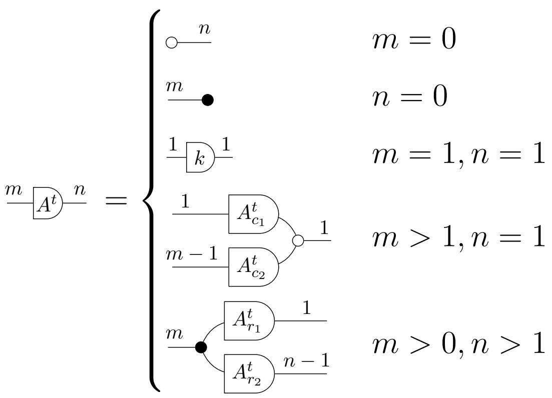
The definition above is inspired by the recursive definition of "classical" matrices:

Definition 46 (Recursive Definition of Classical Matrices)**.**

$$A(n,m) = \begin{cases} k \in \mathsf{k} & m = 1, n = 1 \\ \begin{bmatrix} A_{r_1}(1,1) \\ A_{r_2}(n-1,1) \end{bmatrix} & m = 1, n > 1 \\ \begin{bmatrix} A_{c_1}(n,1) & A_{c_2}(n,m-1) \end{bmatrix} & m > 1, n > 0 \end{cases}$$

This definition is similar to defining a matrix as a list of column vectors. A 1×1 matrix is a scalar, a $m \times 1$ matrix is a $m - 1 \times 1$ matrix with a scalar appended at the top, and a $m \times n$ matrix is a $m \times n - 1$ matrix with a $m \times 1$ matrix appended at the left. Analogously, we can also represent a matrix as a list of row vectors.

Definition 47 (Mat^t)**.**



(3.2)

Soon, we will prove that Definition 45 and Definition 47 are equivalent, in the sense that every matrix A constructed column by column (a diagram of type Mat) can be written as a matrix A^t constructed row by row (a diagram of type Mat^t).

The reader might get confused by the notation A^t for row matrices. This is not the transposed operation. Instead, it hints at the way the matrix was constructed. On Definition 45, matrices were constructed column by column, inductively. However, in Proposition 47, matrices are constructed row by row, inductively.

As in the case of scalars (Proposition 39), the next propositions follow via straightforward induction arguments.

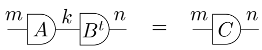
Proposition 48. *Mat is a Map.*

Proposition 49. *Mat^t is a Map.*

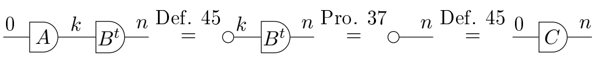
Again, similar to natural numbers, with these propositions we can use Proposition 37 for regular and transposed matrices.

3.1 MATRIX MULTIPLICATION

Our goal is to give an inductive proof that the type Mat is closed by the algebraic operations of sum and multiplication, which correspond to the usual operations on matrices. We start by showing that Mat (a matrix as a list of column vectors) is closed by composition with Mat^t (a matrix as a list of row vectors).

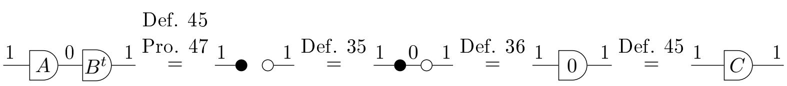
Proposition 50 (Mat ; Mat^t = Mat)**.** 

Proof. Here, since there are multiple parameters, we will need to do more than one induction. First, we will consider the trivial cases where $m = 0$ and $n = 0$, respectively. The proof for $m = 0$ is the following:

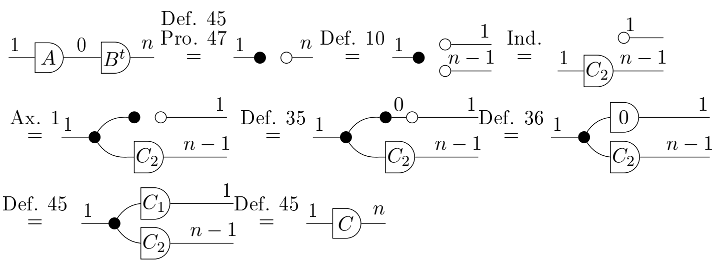


The case where $n = 0$ is completely symmetric, so there is no need to draw its proof.

Now we shall prove the case where $k = 0$. In this one, we will require an induction on m and n. First, we do the base case where $m = n = 1$ and $k = 0$.



Now we do the induction on n:



Matrix multiplication algorithm

procedure $\textsc{Mult}(\ B(n,k),\ A(k,m))$

    if $m > 1$ **then** $\triangleright$ Split A "vertically"

        $\begin{bmatrix} A_1 & A_2 \end{bmatrix} \leftarrow A$

        $C \leftarrow \begin{bmatrix} \textsc{Mult}(B, A_1) & \textsc{Mult}(B, A_2) \end{bmatrix}$

    else if $n > 1$ **then** $\triangleright$ Split B "horizontally"

        $\begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \leftarrow B$

        $C \leftarrow \begin{bmatrix} \textsc{Mult}(B_1, A) \\ \textsc{Mult}(B_2, A) \end{bmatrix}$

    else if $k > 1$ **then** $\triangleright$ Split A "horizontally" and B "vertically"

        $\begin{bmatrix} B_1 & B_2 \end{bmatrix} \leftarrow B$

        $\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \leftarrow A$

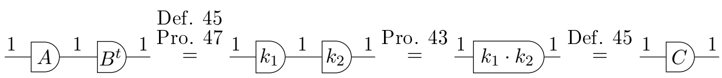
        $C \leftarrow \textsc{Mult}(B_1, A_1) + \textsc{Mult}(B_2, A_2)$

    else $\triangleright$ "Regular" multiplication of numbers$(m = 1, k = 1, n = 1)$

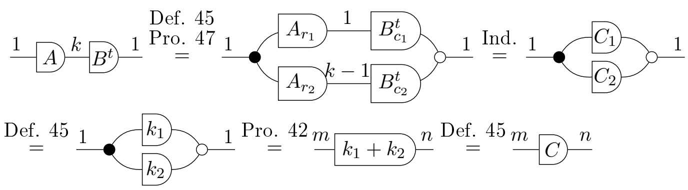
        $C \leftarrow B \cdot A$

    return C

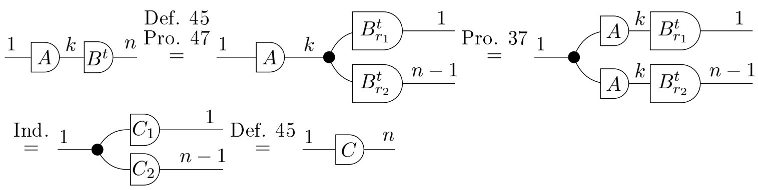
The induction on m is symmetric. After considering all the cases where a parameter equals 0, we will do one more base case where $m = n = k = 1$ and then do an induction on each one of the parameters. So, first, the base case:



Now, we do the induction on k.



Now, the induction on n.

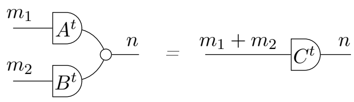


Lastly, the induction on m is symmetric to the previous one. And this completes the proof.

$\square$

This proof actually mimics a classical recursive Matrix Multiplication Algorithm (shown in Algorithm 1). This can be seen as a demonstration that working with the diagrammatic language can yield compositional algorithms for linear algebra.

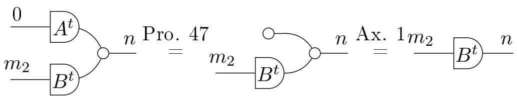
Proposition 51 (Mat^t closed by sum).



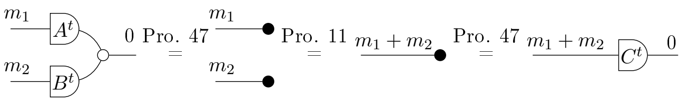
In the classical language, this operation can be viewed as stacking the two matrices side by side:

$$\begin{bmatrix} A^t & B^t \end{bmatrix} = C^t \tag{3.3}$$

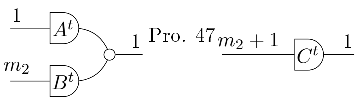
Proof. Here, we do induction on m_1 and n, while leaving m_2 free. First, we do the trivial cases where $m = 0$ and $n = 0$, respectively.



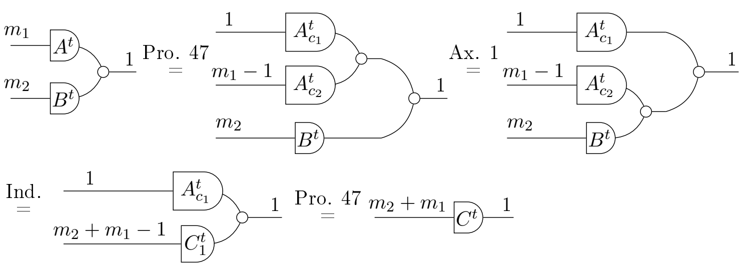
Now, the case where $n = 0$.



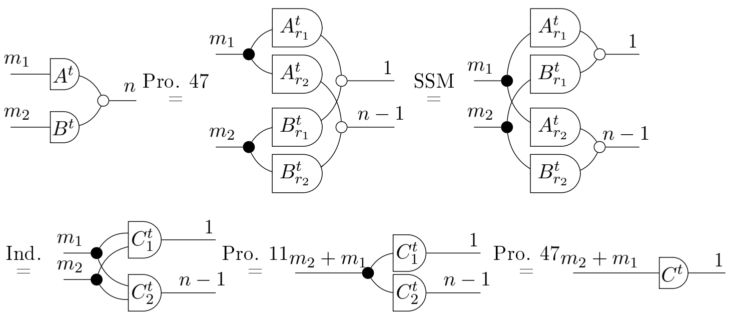
After doing the trivial cases, we prove the base case for the induction. Specifically, the case where $m_1 = n = 1$.



Now we do the induction on m_1.

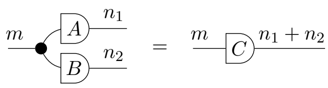


Finally, the induction on n.



□

Notice that, by proving that the type Mat^t is closed by sum we also get for free its transposed version, namely that the type Mat is closed by copy.

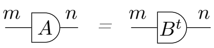


(3.4)

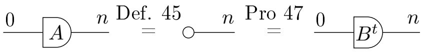
As one would expect, in the classical language it is written as:

$$\begin{bmatrix} A \\ B \end{bmatrix} = C \tag{3.5}$$

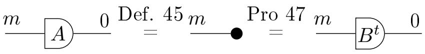
[] We will be referencing Proposition 51 for both results (sum and copy). The same will hold true for any Proposition that has a relevant transposed version.

Proposition 52 (Mat = Mat^t)**.** 

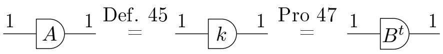
Proof. First, as always, we do the cases where $m = 0$ and $n = 0$, respectively.



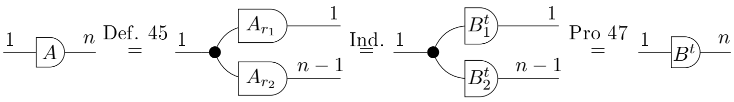
Now, the case where $n = 0$.



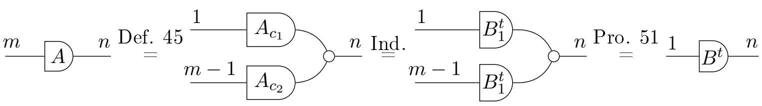
Now, the base case for the induction, where $n = m = 1$.



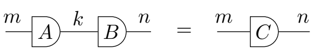
Induction on n.

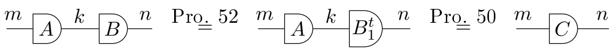


Lastly, induction on m.



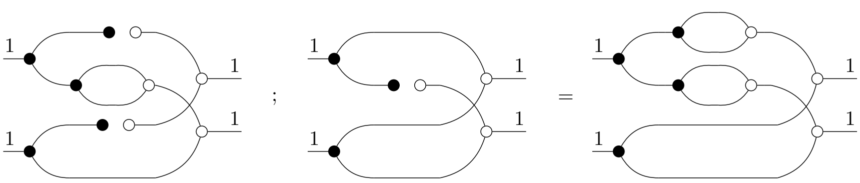
□

Corollary 53 (Mat is closed by composition)**.** 

Proof. 

□

One can view this proof as an algorithm for matrix multiplication, where A and B are the inputs and C is the output. The following image is an example of this operation.

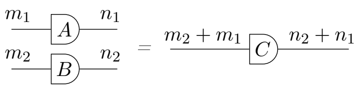


The equivalent way of writing it in the classical language is:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$$

Notice the relation between the operations ; and $\cdot$. Classically, matrices are composed from right to left, but in the graphical language, this is done in the opposite direction. So the first equation is written as $A; B = C$, but the second is $B \cdot A = C$.

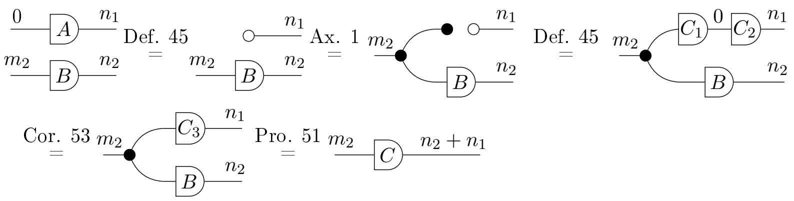
After seeing the example above, one may notice that there is a nice way of comparing classical matrices to their graphical counterparts. The quantity A_{ij} can be determined simply by counting the number of paths from the j-th input wire to the i-th output wire. Similarly, a scalar k in the graphical language is a diagram with 1 input wire and 1 output wire, where the number of paths from left to right is exactly k.

Proposition 54 (Mat closed by parallel composition). 

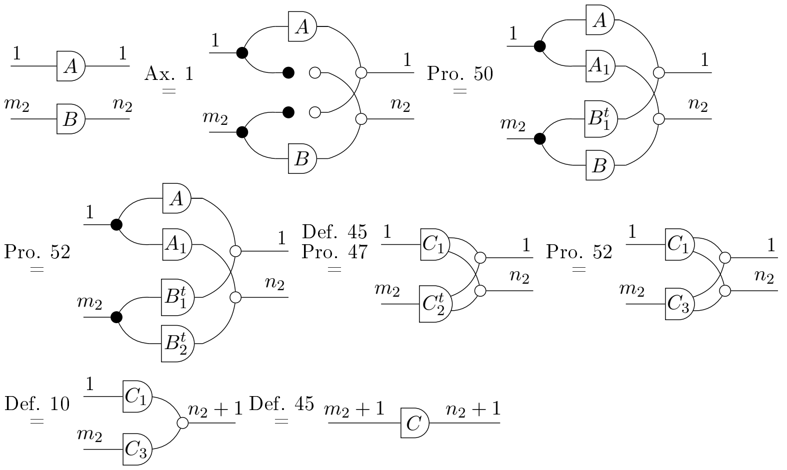
Parallel composition is the same as direct sum in the classical language. The equivalent statement would be:

$$\begin{bmatrix} A_{n_1 \times m_1} & 0 \\ 0 & B_{n_2 \times m_2} \end{bmatrix} = C_{n_1+n_2 \times m_1+m_2} \tag{3.6}$$

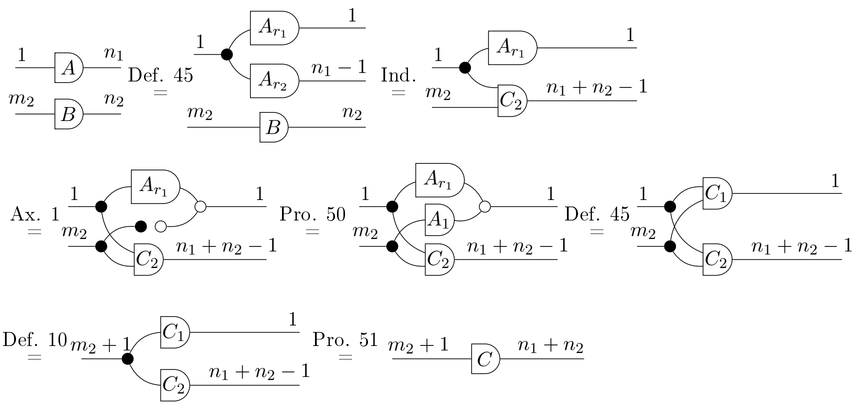
Proof. Here, we will do induction on the parameters m_1 and n_1. First, starting with the cases where $m_1 = 0$ or $n_1 = 0$, respectively.



The case where $n_1 = 0$ is symmetric. Now, the base case for the induction, where $m_1 = n_1 = 1$.



Now, induction on n_1.



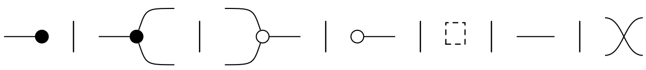
Lastly, induction on m_1 is symmetric. This completes the proof. □

Notice that, in the last step, we use Pro. 51 to reference its transposed version shown in (3.4).

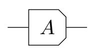
3.1.1 Diagrams from "left to right" are matrices

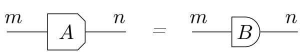
We can now showcase the power of the toolbox of results we have developed. Consider the following grammar, which intuitively uses those elements of (2.1) where the causal flow is left-to-right, i.e. the inputs are on the left and the outputs are on the right.

Definition 55 (HA).

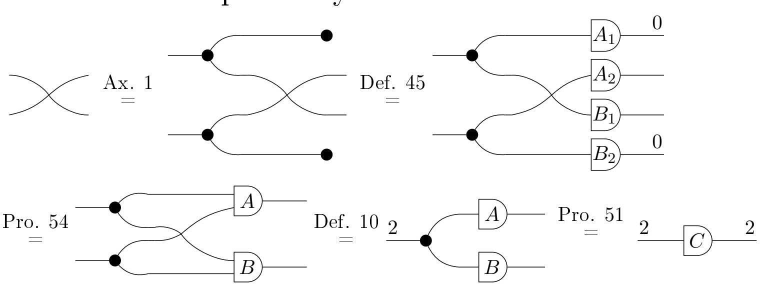
$c, d \quad ::=$  $\mid \; c\,;d \;\mid\; c \oplus d \quad (3.7)$

The toolbox of results proved above can now be used to give a purely "syntactic" proof of a known result, namely that every element of HA is equal to a matrix.

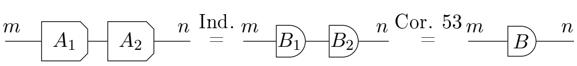
In order to avoid confusion, we will use the symbol  to denote a diagram A of type HA. One can use structural induction to prove the following.

Theorem 56 (HA = Mat). 

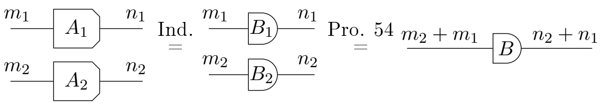
Proof. The base cases will be on the generators. We will be showing only one case, since the other ones are quite easy to do.



Now, in the inductive step, first we consider the composition case.



Finally, the parallel composition case.



$\square$

4 APPLICATIONS OF GLA

This is the most exciting part of the work. In this chapter, we showcase our third contribution. Here, we'll use our axioms to develop a series of proofs and useful characterizations. We first provide a characterization of the notions of injectiveness, surjectiveness and two other related properties. After that, we provide a characterization of solvable linear systems. These two results will turn out to be generalizations of some fundamental facts such as the Invertible Matrix Theorem. Then, we will present a relational decomposition, which will aid us in Section 4.4, where we will prove various linear algebra results calculationally, including the Exchange Lemma. One will also be able to see that, despite some of these theorems having classically extensive and wordy proofs, the arguments provided in this chapter are short and formal. In fact, most of the proofs will have no more than 2 lines.

4.1 INJECTIVENESS, SURJECTIVENESS, KERNEL AND IMAGE IN GLA

Given a diagram R, we first define some derived diagrams from R directly in GLA. Some of these definitions already have names inside the classical, functional linear algebra. Most of the results on this section are from (BONCHI; PAVLOVIC; SOBOCIŃSKI, 2017).

Definition 57 (Derived diagrams)**.**

1. *the nullspace of R:* $N(R) := -\boxed{R}-\circ$,
2. *the multivalued part of R:* $Mul(R) := \circ-\boxed{R}-$,
3. *the image of R:* $Ran(R) := \bullet-\boxed{R}-$,
4. *and the domain of R:* $Dom(R) := -\boxed{R}-\bullet$.

Definition 58 (Dictionary)**.** *Let R be a diagram. We call R*

$$\begin{array}{rlrl} \textit{injective (INJ) if} & -\boxed{R}-\circ \;\leq\; -\circ\,; & \textit{total (TOT) if} & -\bullet \;\leq\; -\boxed{R}-\bullet\,; \\ \textit{single-valued (SV) if} & \circ-\boxed{R}- \;\leq\; \circ- \,; & \textit{surjective (SUR) if} & \bullet- \;\leq\; \bullet-\boxed{R}- \,. \end{array}$$

Notice that injectiveness, surjectiveness and the two new properties, total and single-valued, are expressed as almost identical inequalities (which check if the relation is "consumed" by a zero or a discard). Moreover, since the converses of these inequalities hold (are axioms) in GLA, the four inequalities are actually equalities. With the definition above we clearly obtain some nice symmetries.

Proposition 59 (Dictionary symmetry)**.** *R is total* $\iff$ R^o *is surjective* $\iff$ $R^\dagger$ *is injective* $\iff$ $(R^\dagger)^o$ *is single-valued.*

Proposition 60. *TOT, SV, INJ, SUR are properties closed by series and parallel composition (i.e., if A and B are TOT, then $A;B$ is TOT and $A \oplus B$ is TOT).*

The latter can be easily checked by structural induction. With the four properties in Definition 58, we're finally able to give definitions of maps and invertible maps.

Definition 61. *A diagram is a Map if it is TOT and SV. Also, a diagram is a Co-map if it is INJ and SUR. A diagram is* invertible *if it is a Map and a Co-map (See Figure 7).*

Thanks to the relational symmetry, we also obtain the notion of a co-map, which, roughly speaking, is a map "going from right to left" as opposed to going from left to right. By the definition above and by Proposition 33, we have that the antipode is an invertible map. Definition 58 yields a nice hierarchical structure that relates maps, relations and co-maps (Fig. 7). Notice, however, the distinction between a map and a matrix. In the same sense that linear transformations differ from matrices in the classical sense, this definition of map is not the same as the definition of a matrix given in the previous chapter. On the other hand, they are equivalent, in the sense that every map is equal to some matrix and vice-versa, but this is a proposition that shall be proven later in this chapter.

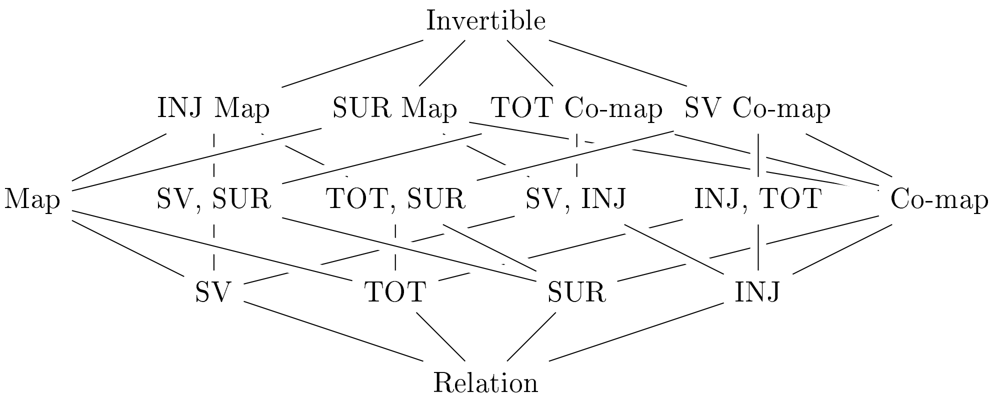
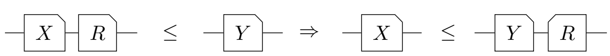
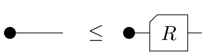


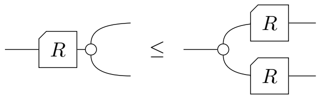
Fig. 7 – Special Relations

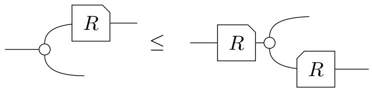
The next proposition shows that the notions in Definition 58 that were described through the converse inequalities of Axiom 2, can also be characterized by universal properties (item 1 from Proposition 62), the converse inequalities of Axioms 3 (item 3 from Proposition 62) and 4 (item 4 from Proposition 62), or by comparing the relation composed with its converse and the identity (item 5 from Proposition 62).

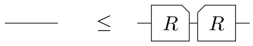
Proposition 62 (Total). *All the following statements are equivalent.*

1. 

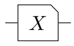
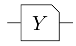
2. 

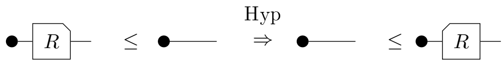
3. 

4. 

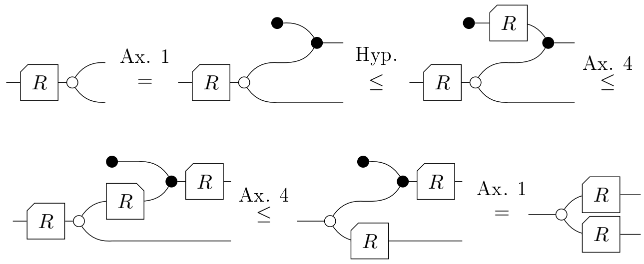
5. 

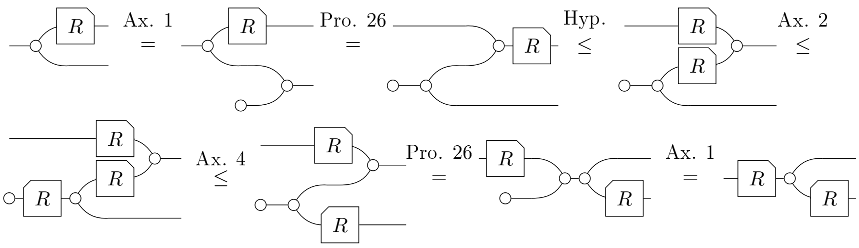
Proof. (1 ⇒ 2)

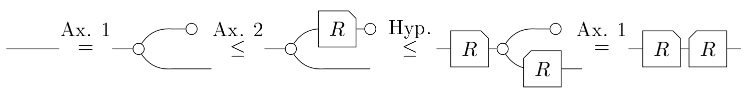
We can instantiate the hypothesis by taking  =  = , obtaining following implication:

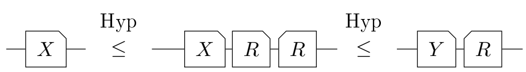


The first term is Axiom 2, and the second term is what we were trying to prove.

(2 ⇒ 3) 

(3 ⇒ 4) 

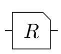
(4 ⇒ 5) 

(5 ⇒ 1) 

□

With the symmetries described in Proposition 59, we can obtain the same characterizations for injective, single-valued and surjective. This yields a dictionary shown in Figure 8. One can see that some theorems about linear transformations are already being captured in these equivalencies. For instance, if we combine all four columns of the table we get a characterization for invertible maps.

Corollary 63. *The following are equivalent.*

1.  *is invertible (TOT, SV, INJ, SUR)*

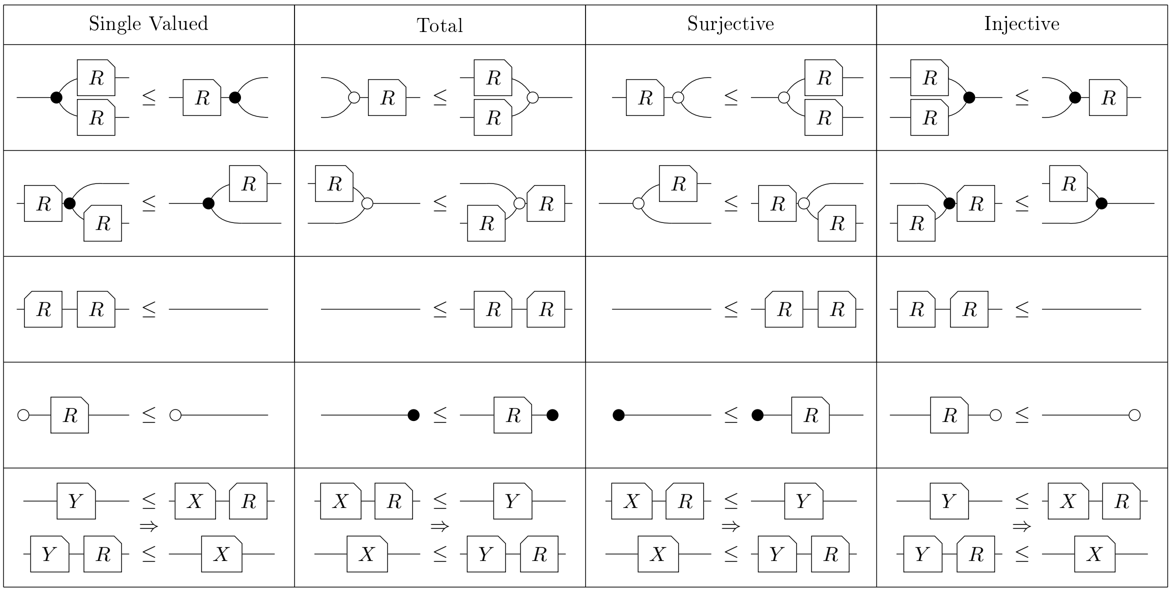
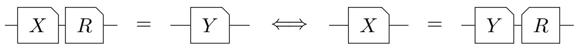
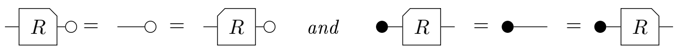


Fig. 8 – The Dictionary

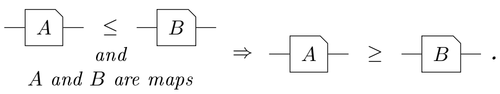
2. 

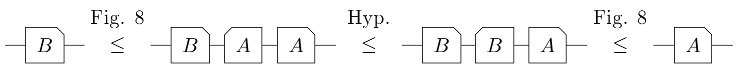
3. 

4. 

The other two were omitted since these 4 are the most interesting ones. The second one states the existence of an inverse, and the third one provides a universal property that characterizes invertible matrices. It's interesting to see that these results all come from the same general characterization.

One immediate result of Fig. 8 is that map inequality is equality, in other words, if two maps are related, they must be equal.

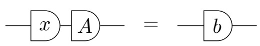
Proposition 64. 

Proof. 

□

4.2 LINEAR SYSTEMS IN GLA

The act of solving linear systems is a fundamental part of linear algebra. In the classical syntax, the compact point-free notation $Ax = b$ is preferable instead of the extensive listing of equalities shown in (1.1). In the graphical language, this representation can be written as the following.

 (4.1)

This way of representing linear systems is point-dependent, in the sense that the solution x needs to be explicitly written in the expression. This is not ideal, since x is not providing any new information to the reader, it is simply there to make the number of wires on both sides of the equation equal. It would be nice to have a way of expressing a linear system without the need for that variable x. And, in fact, there is. In GLA, the existence of a solution of the system $Ax = b$ is equivalent to saying that $b; A^{op}$ is a total relation. Before proving that, however, we'll need the following constructive result, which will be proven in the next section.

Proposition 65. $-\boxed{B}-\boxed{A}-$ *is TOT* $\iff \exists S \quad -\boxed{S}- \;\leq\; -\boxed{B}-\boxed{A}-$

The need for this proposition, in a sense, is intuitive. If we want to establish a connection between the relation $B; A^{op}$ and the existence of a solution S to the system, we need a way of constructing that solution. Assuming this result, one is able to describe "solvability" inside GLA merely by stating that $B; A^{op}$ is a total relation.

As a sidenote, the ($\Rightarrow$) direction of this Proposition can be seen as a weakened version of the axiom of choice (since we're only dealing with linear relations). A total relation has at least one element (x, y) for every x. However, there may be multiple elements. The way to construct a map S using R is, essentially, by "choosing" a single element (x, y) for each input x.

Proposition 66 (Choice). $-\boxed{R}-$ *is TOT* $\Rightarrow \exists S \quad \begin{array}{c} -\boxed{S}- \;\leq\; -\boxed{R}- \\ \textit{and} \\ -\boxed{S}- \;\textit{is a map} \end{array}$

After Proposition 65, we're ready to prove a complete characterization.

Lemma 67. *The following are equivalent.*

1. $-\boxed{B}-\boxed{A}-$ *is TOT* ,
2. $\exists S \quad -\boxed{S}- \;\leq\; -\boxed{B}-\boxed{A}-$,
3. $\exists S \quad -\boxed{S}-\boxed{A}- \;=\; -\boxed{B}-$,
4. $\bullet-\boxed{B}- \;\leq\; \bullet-\boxed{A}-$.

Proof.

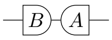
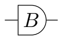
(1 $\Rightarrow$ 2) Proposition 65,

(2 $\Rightarrow$ 3) $-\boxed{S}-\boxed{A}- \overset{\text{Hyp.}}{\leq} -\boxed{B}-\boxed{A}-\boxed{A}- \overset{\text{Fig. 8}}{\leq} -\boxed{B}- \overset{\text{Prop. 64}}{\Rightarrow} -\boxed{S}-\boxed{A}- = -\boxed{B}-$,

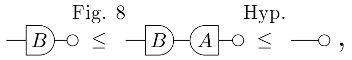
(3 $\Rightarrow$ 4) $\bullet-\boxed{B}- \overset{\text{Fig. 8}}{\leq} \bullet-\boxed{S}-\boxed{B}- \overset{\text{Hyp.}}{\leq} \bullet-\boxed{S}-\boxed{S}-\boxed{A}- \overset{\text{Fig. 8}}{\leq} \bullet-\boxed{A}-$,

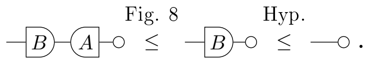
(4 $\Rightarrow$ 1) $-\boxed{B}-\boxed{A}-\bullet \overset{\text{Hyp.}}{\leq} -\boxed{B}-\boxed{B}-\bullet \overset{\text{Fig. 8}}{\leq} -\bullet$. $\square$

Also, in the case where the relation is injective (or single-valued, by symmetry), there is an additional property in the characterization.

Lemma 68.  *is INJ* $\iff$  *is INJ* .

Proof.

(⇒) 

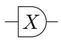
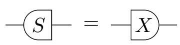
(⇐) 

□

All of these equivalencies yield a new dictionary, shown in Fig. 9, which we're calling "linear system dictionary". Different from Fig. 8, this one considers relations in the form $B; A^{op}$, where A and B are matrices. Notice that TOT and SUR don't share the same equivalencies with SV and INJ. That happens because the symmetry of inverting the colors moves us from $B; A^{op}$ form to $B^{op}; A$ form. In fact, there is a dictionary analogous to this one where the relation is in the latter form. However, since the equivalencies are analogous, we will not talk about this one.

One interesting thing about the dictionary shown in Fig. 9 is that it generalizes various useful results in linear algebra. First of all, take an invertible matrix X, that is, SUR, TOT, INJ, SV. if we let $A = I, B = X$ we get parts of the Invertible Matrix Theorem.

Corollary 69 (Invertible Matrix Theorem)**.** *The following are equivalent.*

1.  *is invertible (TOT, SV, INJ, SUR)* ,
2. $\exists S$  ,
3. $\exists S$ 

It is possible to get similar equivalencies by assuming X is only injective or only surjective. One can also instantiate $A = X$ and $B = X$. Each instance leads to a different identity. This is a good example of the expressiveness of these results. The dictionary in Fig. 9, together with Fig. 8, do not seem to be merely nice little side facts, but rather fundamental characterizations of linear algebra.

4.3 GENERALIZED RANK-REVEALING DECOMPOSITION

In linear algebra, the induction inside the proofs is usually encapsulated in the gaussian elimination. As pointed out by Georges Gonthier (GONTHIER, 2011), "All of the usual results on linear spaces and bases easily follow from these definitions, (...) all with proofs under twelve lines (most are under two) and no induction. The reason for this is that all the induction we need is neatly encapsulated inside the Gaussian elimination procedure".

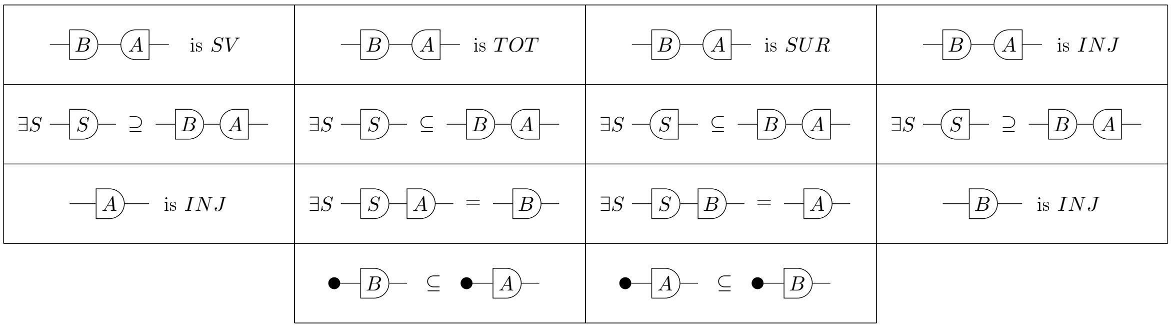
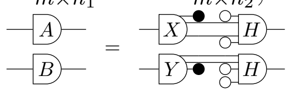
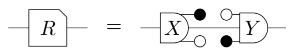


Fig. 9 – The Linear System Dictionary

Here, the algorithm we'll use will be called Generalized Rank Revealing decomposition (GRR), and it will play a similar role. Rank-revealing decompositions have already been studied in various related works (LOAN, 1976; PAIGE; SAUNDERS, 1981; STEWART, 2016). Despite its name, we'll not be talking (explicitly) about rank in this work. It is a decomposition of pairs of matrices and can produce bases for all relevant subspaces about them such as the intersection and the sum of their image. In this section, we present the graphical form of the Generalized Rank Revealing decomposition and prove some of its properties.

Proposition 70 (Explicit Form)**.** *For all matrices $A_{m\times n_1}$ and $B_{m\times n_2}$, there exists invertible matrices $X_{n_1\times n_1}$, $Y_{n_2\times n_2}$ and $H_{m\times m}$ such that*  .

As one can see, this is a decomposition that relates a pair of matrices. It is not explicitly relational, but it can be adapted via structural induction to work for any linear relation. The next Proposition shows an equivalent and relational form of the GRR Decomposition.

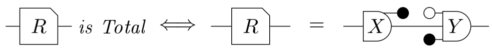
Proposition 71 (Implicit Form)**.** *For all relations $R_{m\times n}$, there exists invertible matrices $X_{n\times n}$ and $Y_{m\times m}$ such that*  .

In this version, the H doesn't appear, and that is why this version is being called "implicit". H, roughly speaking, is the matrix that "connects" A and B and holds all the relevant information about the subspaces concerning them. It is possible to reconstruct H given the relational decomposition, however, we shall not delve deeper into this matter, as the current goal is to merely use GRR as a tool to shorten proofs. The proof of Propositions 70 and 71 are out of the scope of this work and will be omitted. However, the reader should keep in mind that, in terms of proofs, these two Propositions will be the main tools of this chapter.

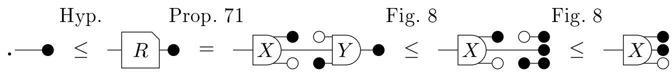
4.3.1 Counting the number of zeros and discards

Informally speaking, the dictionaries provide us with characterizations about the abstract notion of linear relations. The GRR decomposition, on the other hand, can express linear relations in terms of matrices, which have useful inductive structures. By combining the GRR Decomposition with the dictionaries, we can prove some interesting Propositions. But first, it will be useful to explicitly write the following Proposition about the structure of GLA.

Proposition 72. *$A \oplus B \leq C \oplus D \iff A \leq B$ and $C \leq D$.*

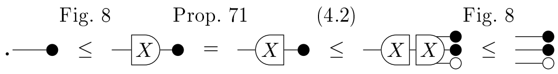
Theorem 73 (Total means no co-zeros)**.** 

Proof. ($\Rightarrow$)

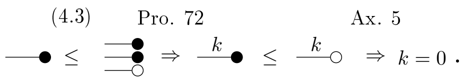
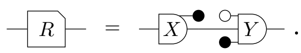


(4.2)

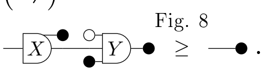
That implies the following.



(4.3)

Lastly, we have that  Therefore, the number of zeros "to the left" in the decomposition of R is 0. Thus 

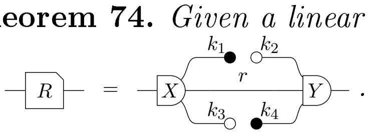
($\Rightarrow$)



$\square$

This result establishes a connection between the number of zeros and discards in the GRR decomposition and the properties of TOT, SV, SUR, INJ. The other three theorems can be obtained by using the symmetries in Proposition 59, forming the characterization below.

Theorem 74. *Given a linear relation R and its GRR decomposition*



. Then, the following hold.

1. *R is TOT $\iff k_3 = 0$,*

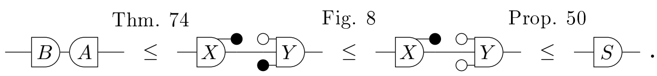
2. *R is SV $\iff k_4 = 0$,*

3. *R is INJ $\iff k_1 = 0$,*

4. *R is SUR $\iff k_2 = 0$.*

If we combine the total and single-valued versions of it, we arrive at a constructive version of maps. Also, with this theorem we are finally able to prove Proposition 65.

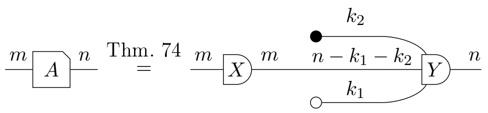
Proposition 75. 

Proof.  $\square$

4.4 LINEAR ALGEBRA RESULTS

In this section, we'll use all the tools developed throughout the chapter to prove various linear algebra results. First, we prove that relations that are total and injective at the same time can't have more input wires than output wires. This immediately implies that injective maps can't have the domain bigger than the codomain.

Proposition 76. $\xrightarrow{m}\boxed{A}\xrightarrow{n}$ *is INJ and TOT* $\Rightarrow m \leq n$.

Proof.  $\Rightarrow \; m = n - k \Rightarrow m \leq n$. ∎

By symmetry, we also obtain the version for surjective and single-valued relations.

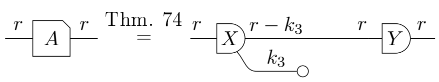
Proposition 77. $\xrightarrow{m}\boxed{A}\xrightarrow{n}$ *is SUR and SV* $\Rightarrow m \geq n$.

By joining these two, we can arrive at a theorem for invertible maps.

Proposition 78. $\xrightarrow{m}(A)\xrightarrow{n}$ *is invertible* $\Rightarrow m = n$.

This proposition guarantees that, when all four properties (INJ, SUR, TOT, SV) hold, then the relation must be square, that is, with the same number of input and output wires. There is a similar result in linear algebra, which states that if at least three properties hold and the relation is square, then the fourth property must hold. We'll prove it only for the TOT case, but by symmetry, the same can be achieved for any of the remaining three.

Proposition 79. $\xrightarrow{r}\boxed{R}\xrightarrow{r}$ *is SUR, INJ and SV* $\Longrightarrow$ $\xrightarrow{r}\boxed{R}\xrightarrow{r}$ *is TOT.*

Proof.  $\Rightarrow \; r - k_3 = r \Rightarrow k_3 = 0 \overset{\text{Thm. 74}}{\iff} \xrightarrow{r}\boxed{A}\xrightarrow{r}$ is *TOT* . ∎

The following result states that every map can be represented by some matrix.

Proposition 80. *A is a map* $\iff$ *there is a matrix M such that* $-\boxed{A}- \; = \; -(M)-$.

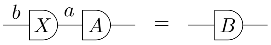
Proof. $-\boxed{A}- \overset{\text{Thm. 74}}{=} -(X)-\bullet \quad \circ-(Y)- \overset{\text{Prop. 50}}{=} -(M)-$. ∎

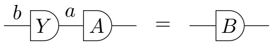
Analogously, one can prove similar results for co-maps and invertible maps.

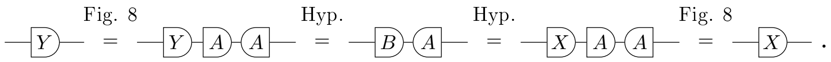
Proposition 81. *A is a co-map* $\iff$ *there is a matrix M such that* $-\boxed{A}- \; = \; -(M)-$.

Proposition 82. *A is an invertible map* $\iff$ *there is an invertible matrix M such that* $-\boxed{A}- \; = \; -(M)-$.

It is also possible to prove uniqueness in the graphical language. In particular, the following result will be useful.

Proposition 83.  *and A is INJ $\Longrightarrow$ X is unique.*

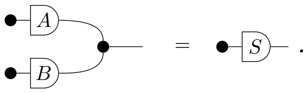
Proof. Suppose there exists Y such that  . Then



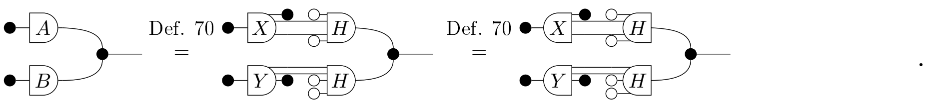
□

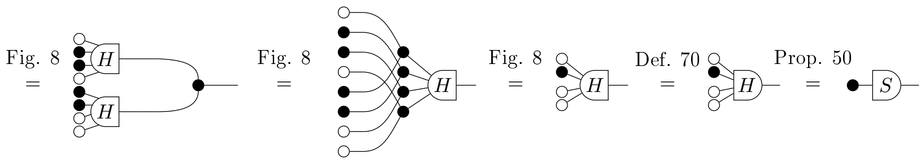
4.4.1 Calculating subspaces with GRR

The GRR decomposition can generate bases for all the relevant subspaces about a pair of matrices. Also, the bases for each of these subspaces are subsets of a single basis that spans the whole space. As an example, below is the construction of a basis for the intersection of the image of two matrices A and B.

Lemma 84. 

Proof.





□

The construction of the other subspaces is analogous, and they generate a table of different bases related to the relevant subspaces of A and B. This table is shown in Figure 10, where A and B are short terms for Img(A) and Img(B). Notice that all of the bases are contained in the basis for the whole space, in contrast to other algorithms such as Zassenhaus' (ZASSENHAUS, 1966), which calculate unrelated bases for sum and intersection.

In Fig. 10, we can extract intuition about the 4 input wires of the matrix H. Roughly speaking, the first wire contains information about the part that is "only inside of B", namely, $B - (A \cap B)$. The third wire contains information about the part that is "only inside of A" ($A - (A \cap B)$). The second wire contains information about their intersection. Finally, the last wire contains information about the rest of the space. Together, all wires represent the whole space.

4.4.2 Proving classical results in GLA

In this subsection, we derive some results from the book Linear Algebra Done Right by Sheldon Axler (AXLER, 2015). The propositions will have their numeration inside the book in order to facilitate future reference. This is being done as a proof of concept, showing that graphical linear algebra has the potential of not only simplifying the theory

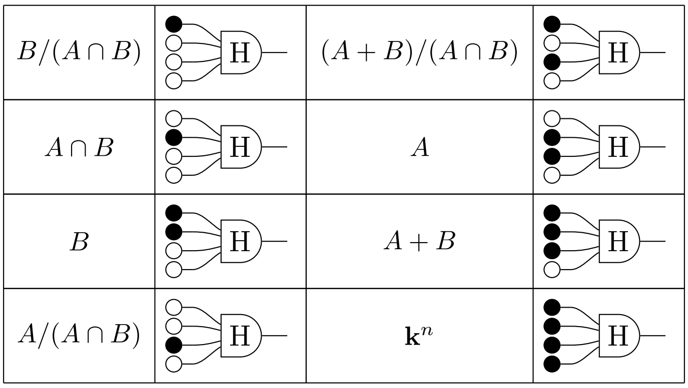
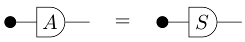
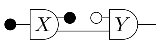
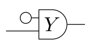


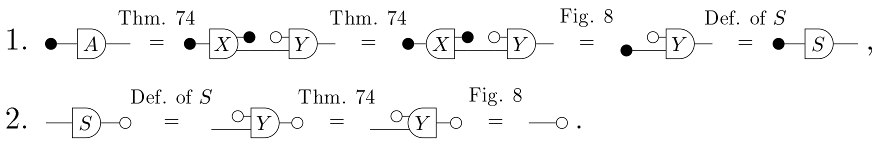
Fig. 10 – Subspaces of a pair of matrices A,B, where $\mathbf{k}^n$ is the whole space.
Here, / and + are the usual vector space operations.

but also formalizing it and making new connections. One will be able to see that, despite some of these theorems having extensive and wordy classical proofs, the arguments inside this approach are short and calculational.

First, we'll represent finite-dimensional vector spaces by the image of a matrix, which can be thought of as the span of a list of vectors. After the following proposition, which asserts that every finite-dimensional vector space has a basis, we'll instead represent them by a basis.

Proposition 85 (Every finite-dimensional vector space has a basis (2.32))**.** *For all A, there exists S such that*  *and S is INJ.*

Proof. Let  be the GRR decomposition of A. Then, define S as . We have



□

Here, a basis is being represented by an injective matrix, which can be thought of as a linearly independent list of vectors. For the rest of this section, the reader should consider the following.

1. Linearly independent list of vectors will be represented by injective matrices.

2. Vector spaces will be represented by the image of a matrix.

3. A list of vectors will be a basis of V if it is linearly independent and its span is equal to V.

Proposition 86 (Exchange Lemma (2.23)). $\bullet\overset{b}{-}\boxed{B}\!- \;\leq\; \bullet\overset{a}{-}\boxed{A}\!-$ *and B is INJ* $\implies b \leq a$.

Proof.

1. $\bullet\overset{b}{-}\boxed{B}\!- \;\leq\; \bullet\overset{a}{-}\boxed{A}\!- \overset{\text{Fig. 9}}{\Rightarrow} \overset{b}{-}\boxed{B}\!-\!\boxed{A}\overset{a}{-}$ is TOT ,
2. B is $INJ \overset{\text{Fig. 9}}{\Rightarrow} \overset{b}{-}\boxed{B}\!-\!\boxed{A}\overset{a}{-}$ is INJ ,
3. $\overset{b}{-}\boxed{B}\!-\!\boxed{A}\overset{a}{-}$ is INJ and $TOT \overset{\text{Prop. 76}}{\Rightarrow} b \leq a$.

□

Here, a linearly independent list of vectors is being represented by an injective matrix, and a spanning list of vectors is merely a matrix whose image contains the image of the linearly independent list. The exchange lemma states that the length of any spanning list must be greater or equal to the length of any linearly independent list, in other words, $b \leq a$. Normally, the proof would include a description of an exchange process of the vectors. Here, this doesn't happen, since the last implication makes use of the GRR decomposition. All the algorithmic part of the proof is postponed to the last step and encapsulated in a single auxiliary Lemma.

Proposition 87 (Linearly independent list of the right lenght is a basis (2.39)). $\bullet\overset{r}{-}\boxed{B}\!- \;\leq\; \bullet\overset{r}{-}\boxed{A}\!-$, *B is INJ and A is INJ* $\implies \bullet\overset{r}{-}\boxed{B}\!- \;=\; \bullet\overset{r}{-}\boxed{A}\!-$.

Proof.

1. $\bullet\overset{r}{-}\boxed{B}\!- \;\leq\; \bullet\overset{r}{-}\boxed{A}\!- \overset{\text{Fig. 9}}{\Rightarrow} \overset{r}{-}\boxed{B}\!-\!\boxed{A}\overset{r}{-}$ is TOT ,
2. B is $INJ \overset{\text{Fig. 9}}{\Rightarrow} \overset{r}{-}\boxed{B}\!-\!\boxed{A}\overset{r}{-}$ is INJ ,
3. A is $INJ \overset{\text{Fig. 9}}{\Rightarrow} \overset{r}{-}\boxed{B}\!-\!\boxed{A}\overset{r}{-}$ is SV ,
4. $\overset{r}{-}\boxed{B}\!-\!\boxed{A}\overset{r}{-}$ is $INJ, SV, TOT \overset{\text{Prop. 79}}{\Rightarrow} \overset{r}{-}\boxed{B}\!-\!\boxed{A}\overset{r}{-}$ is $SUR \overset{\text{Fig. 9}}{\Rightarrow} \bullet\overset{r}{-}\boxed{B}\!- \;\leq\; \bullet\overset{r}{-}\boxed{A}\!-$.

□

This theorem classically states that every linearly independent list of vectors in a finite-dimensional subspace V with length $r := \dim(V)$, where dim means dimension, is a basis of V. We're representing V by a basis A. Then, to prove the theorem, we showed that a linearly independent list B with length r and whose image is in V must span V, therefore it must be a basis.

Proposition 88 (Spanning list of the right lenght is a basis (2.42)). $\bullet\overset{r}{-}\boxed{B}\!- \;=\; \bullet\overset{r}{-}\boxed{A}\!-$ *and A is INJ* $\implies$ *B is INJ.*

Proof.

1. $\bullet\overset{r}{-}\boxed{B}- \;=\; \bullet\overset{r}{-}\boxed{A}- \;\Rightarrow\; \bullet\overset{r}{-}\boxed{B}- \;\leq\; \bullet\overset{r}{-}\boxed{A}- \;\overset{\text{Fig. 9}}{\Rightarrow}\; \overset{r}{-}\boxed{B}\boxed{A}\overset{r}{-}$ is TOT ,
2. $\bullet\overset{r}{-}\boxed{B}- \;=\; \bullet\overset{r}{-}\boxed{A}- \;\Rightarrow\; \bullet\overset{r}{-}\boxed{B}- \;\leq\; \bullet\overset{r}{-}\boxed{A}- \;\overset{\text{Fig. 9}}{\Rightarrow}\; \overset{r}{-}\boxed{B}\boxed{A}\overset{r}{-}$ is SUR ,
3. A is INJ $\overset{\text{Fig. 9}}{\Rightarrow}$ $\overset{r}{-}\boxed{B}\boxed{A}\overset{r}{-}$ is SV ,
4. $\overset{r}{-}\boxed{B}\boxed{A}\overset{r}{-}$ is TOT, SUR, SV $\overset{\text{Prop. 79}}{\Rightarrow}$ $\overset{r}{-}\boxed{B}\boxed{A}\overset{r}{-}$ is INJ $\overset{\text{Fig. 9}}{\Rightarrow}$ B is INJ .

$\square$

Again, let V be the subspace represented by the basis A. This result is similar. Instead of showing that a linearly independent list B with length r must span V, we're showing that a spanning list B with length r is linearly independent, and therefore a basis.

Proposition 89 (Bases have same size (2.35))**.** $\bullet\overset{b}{-}\boxed{B}- \;=\; \bullet\overset{a}{-}\boxed{A}-$ *, B is INJ and A is INJ* $\Longrightarrow b = a$.

Proof.

1. $\bullet\!-\boxed{B}- = \bullet\!-\boxed{A}- \Rightarrow \bullet\!-\boxed{B}- \leq \bullet\!-\boxed{A}- \overset{\text{Fig. 9}}{\Rightarrow} -\boxed{B}\boxed{A}-$ is TOT ,
2. $\bullet\!-\boxed{B}- = \bullet\!-\boxed{A}- \Rightarrow \bullet\!-\boxed{B}- \leq \bullet\!-\boxed{A}- \overset{\text{Fig. 9}}{\Rightarrow} -\boxed{B}\boxed{A}-$ is SUR ,
3. $-\boxed{B}-$ is INJ $\overset{\text{Fig. 9}}{\Rightarrow}$ $-\boxed{B}\boxed{A}-$ is INJ ,
4. $-\boxed{A}-$ is INJ $\overset{\text{Fig. 9}}{\Rightarrow}$ $-\boxed{B}\boxed{A}-$ is SV ,
5. $-\boxed{B}\boxed{A}-$ is TOT, SUR, INJ, SV $\overset{\text{Prop. 78}}{\Rightarrow}$ $b = a$.

$\square$

Here, we showed that two equal bases must have the same size.

Proposition 90 (Unique solution to linear system)**.** $\bullet\overset{b}{-}\boxed{B}- \;\leq\; \bullet\overset{a}{-}\boxed{A}-$ *and A is INJ* $\Longrightarrow$ $\exists! X \; \overset{b}{-}\boxed{X}\overset{a}{-}\boxed{A}- \;=\; -\boxed{B}-$.

Proof. $\bullet\!-\boxed{B}- \;\leq\; \bullet\!-\boxed{A}- \overset{\text{Fig. 9}}{\Rightarrow} \exists X \;\; -\boxed{X}\boxed{A}- = -\boxed{B}- \overset{\text{Prop. 83}}{\Rightarrow} -\boxed{X}-$ is unique. . $\square$

This proposition states that if $\mathrm{Img}(B) \subseteq \mathrm{Img}(A)$ and A is injective, then the system $Ax = B$ must admit an unique solution.

5 CONCLUSION

Our goal in this work was to understand how GLA could be developed to address the three following issues: the lack of classical results proven inside the graphical language, the extensive set of Axioms as originally formulated in the work Interacting Hopf Algebras (ZANASI, 2015), and lastly the definition of matrices, which up until then was iterative, instead of recursive. We addressed each of these issues in the three main chapters of this work.

In the second chapter, we first constructed a careful set of 8 Axioms that encapsulate everything we need, in the sense that these Axioms are equivalent to the Axioms present in the work Interacting Hopf Algebras, as shown in Theorem 44. With this, the completeness result proven in that work extends to our axioms too. This set of Axioms was more compact, had double duality instead of duality and made use of inequalities to shorten proofs.

In the third chapter, we developed the theory further by introducing a fully graphical inductive definition of matrices. We then carefully proved each of the basic matrix properties until we reached a matrix multiplication algorithm. This definition was recursive, as opposed to the already existent iterative definitions of matrices in the graphical language.

Lastly, in the fourth chapter, we used the GRR decomposition and all the tools we've developed to prove various results and equivalencies. Here, we are addressing the main issue concerning GLA: most of the classical results of linear algebra weren't yet proven inside the graphical language. We showcase a single relational decomposition which served as a tool to simplify proofs. Some results proven in this chapter have wordy and complicated classical proofs, which in turn have one-line proofs in GLA using the decomposition. Not only that but, in this chapter, several different results, such as the Exchange Lemma and the Inverse Matrix Theorem, were generalized into a single characterization.

The advantages of this approach are apparent. The proven results are diverse, but they all have short, calculational and somewhat similar proofs. We believe that the formality and graphical intuition of GLA showcase its potential as a tool to develop and teach linear algebra, and the proofs we've constructed in this work are the evidence for that.

5.1 FUTURE WORKS

GLA is still in its growth period. We did nothing but explore the tip of the iceberg. Using the foundations built in this work, we would like to develop the theory even further. We're interested in bringing more chapters of a typical linear algebra textbook into the graphical language. In other words, notions of orthogonality, eigenvalues, eigenvectors, and so on. To do that, it would be necessary to understand how different fields can be represented inside GLA. In this work, we only developed diagrammatic integers. It turns out that the field $\mathbb{Q}$ can be represented by an integer a composed with the converse of a non-zero integer b (in other words, a rational number a/b). We didn't delve much into this for simplicity. It is also possible to use $\mathbb{Z}_p$ simply by stating one additional axiom: $-\!\boxed{p}\!- \;=\; -\!\boxed{0}\!-$. We are particularly interested in how one can express real numbers inside the graphical language. They can't be all constructed inductively since there is a cardinality problem. However, there are at least two options. The boring option is to define a single generator for each real number. Another option would be to represent them in terms of continued fractions. In that sense, real numbers have nice representations in the graphical language. We also wonder if, under this definition, proofs about reals such as the irrationality of $\sqrt{2}$ can be proven in GLA.

We introduced a compact set of axioms for GLA. However, it seems to be possible to shorten the set of axioms even more. More specifically, Axioms 4 and 3 seem redundant. They seem to simply be "rotations"of the same general axiom. Informally speaking, if we disregard direction (namely, the left to right flow of the diagrams) they indeed become the same axiom. It would be interesting to see GLA being developed in a directionless environment (CARETTE, 2021) and understand what this new structure is representing.

We've presented a single relational decomposition in this work, the Generalized Rank Revealing Decomposition. However, we didn't provide proof for its construction, and instead merely assumed it to hold. Unfortunately, these proofs were out of the scope of this work, but we plan on presenting them in another work. This also includes the other dictionaries which were omitted. For instance, the dictionary mentioned in 4.2, which is similar to the linear system dictionary.

It would be interesting to explore different decompositions in the graphical language. Two examples are a relational version of the SVD and a relational version of LU. The latter is, currently, already in development by us. We can say that triangular matrices and triangular decompositions behave well in the graphical language, and we do plan on writing these proofs down in a near future.

Other types of matrices could also be explored, such as the Reduced Row Echelon Form. One of our interests is proving inside the graphical language the uniqueness of the matrix that appears as the result of a gaussian elimination. That matrix is, in fact, in Reduced Row Echelon Form. Most classical proofs of this fact are wordy and confusing, and we hope GLA can help clarify these arguments.

Lastly, the biggest future work we're planning to do is a graphical linear algebra textbook. This textbook would develop the theory from the ground up using only GLA. It would be suitable for university use and would provide a different approach to how linear algebra is taught today.

BIBLIGRAPHY

ARENS, R. et al. Operational calculus of linear relations. **Pacific Journal of Mathematics**, Pacific Journal of Mathematics, v. 11, n. 1, p. 9–23, 1961.

AXLER, S. **Linear algebra done right**. [S.l.]: Springer, 2015.

BACK, R.; GRUNDY, J.; WRIGHT, J. V. Structured calculational proof. **Formal Aspects of Computing**, Springer, v. 9, n. 5, p. 469–483, 1997.

BACKHOUSE, R.; FERREIRA, J. F. On euclid's algorithm and elementary number theory. **Science of Computer Programming**, Elsevier, v. 76, n. 3, p. 160–180, 2011.

BACKHOUSE, R.; OLIVEIRA, J. N. **Mathematics of Program Construction: 5th International Conference, MPC 2000 Ponte de Lima, Portugal, July 3-5, 2000 Proceedings**. [S.l.]: Springer, 2006.

BAEZ, J. **Bimonoids from Biproducts**. 2010. Disponível em: https://golem.ph.utexas.edu/category/2010/09/bimonoids_from_biproducts.html.

BAEZ, J.; ERBELE, J. Categories in Control. **Theory and Application of Categories**, v. 30, p. 836–881, 2015.

BEESON, M.; NARBOUX, J.; WIEDIJK, F. Proof-checking euclid. **Annals of Mathematics and Artificial Intelligence**, Springer, v. 85, n. 2, p. 213–257, 2019.

BONCHI, F.; PAVLOVIC, D.; SOBOCIŃSKI, P. Functorial semantics for relational theories. **arXiv preprint arXiv:1711.08699**, 2017.

BONCHI, F.; SOBOCIŃSKI, P.; ZANASI, F. The calculus of signal flow diagrams i: linear relations on streams. **Information and Computation**, Elsevier, v. 252, p. 2–29, 2017.

BONCHI, F.; SOBOCIŃSKI, P.; ZANASI, F. Interacting Hopf algebras. **Journal of Pure and Applied Algebra**, Elsevier, v. 221, n. 1, p. 144–184, 2017.

CARETTE, T. **Wielding the ZX-calculus, Flexsymmetry, Mixed States, and Scalable Notations**. Tese (Doutorado) — Université de Lorraine, 2021.

CODDINGTON, E. A. **Extension theory of formally normal and symmetric subspaces**. [S.l.]: American Mathematical Soc., 1973. v. 134.

CROSS, R. **Multivalued linear operators**. [S.l.]: CRC Press, 1998. v. 213.

DIJKSTRA, E. W.; SCHOLTEN, C. S. **Predicate calculus and program semantics**. [S.l.]: Springer Science & Business Media, 2012.

ERBELE, J. **Redundancy and Zebra Snakes**. Disponível em: https://graphicallinearalgebra.net/2017/10/23/episode-r1-redundancy-and-zebra-snakes/.

FEFERMAN, S. Tarski's influence on computer science. **Logical Methods in Computer Science**, Episciences. org, v. 2, 2006.

FREITAS, V. M. d. **A vida em setas: uma introdução à teoria de categorias aplicada**. Tese (Doutorado) — Universidade de São Paulo, 2019.

GIVANT, S. The calculus of relations as a foundation for mathematics. **Journal of Automated Reasoning**, Springer, v. 37, n. 4, p. 277–322, 2006.

GONTHIER, G. Point-free, set-free concrete linear algebra. In: SPRINGER. **International Conference on Interactive Theorem Proving**. [S.l.], 2011. p. 103–118.

GRIES, D.; SCHNEIDER, F. B. Teaching math more effectively, through calculational proofs. **The American mathematical monthly**, Taylor & Francis, v. 102, n. 8, p. 691–697, 1995.

LAFONT, Y. Towards an algebraic theory of boolean circuits. **Journal of Pure and Applied Algebra**, Elsevier, v. 184, n. 2-3, p. 257–310, 2003.

LANE, S. M. An algebra of additive relations. **Proceedings of the National Academy of Sciences of the United States of America**, National Academy of Sciences, v. 47, n. 7, p. 1043, 1961.

LIFSCHITZ, V. Calculational proofs. In: **Edsger Wybe Dijkstra: His Life, Work, and Legacy**. [S.l.: s.n.], 2022. p. 231–246.

LOAN, C. F. V. Generalizing the singular value decomposition. **SIAM Journal on numerical Analysis**, SIAM, v. 13, n. 1, p. 76–83, 1976.

MADDUX, R. D. The origin of relation algebras in the development and axiomatization of the calculus of relations. **Studia logica**, Springer, v. 50, p. 421–455, 1991.

PAIGE, C. C.; SAUNDERS, M. A. Towards a generalized singular value decomposition. **SIAM Journal on Numerical Analysis**, SIAM, v. 18, n. 3, p. 398–405, 1981.

PAIXÃO, J.; SOBOCIŃSKI, P. Calculational Proofs in Relational Graphical Linear Algebra. In: SPRINGER. **Brazilian Symposium on Formal Methods**. [S.l.], 2020. p. 83–100.

SCHMIDT, G. **Relational mathematics**. [S.l.]: Cambridge University Press, 2011. v. 132.

SELINGER, P. A survey of graphical languages for monoidal categories. In: **New structures for physics**. [S.l.]: Springer, 2010. p. 289–355.

SELINGER, P. A survey of graphical languages for monoidal categories. **New structures for physics**, Springer, p. 289–355, 2011.

SOBOCIŃSKI, P. Graphical linear algebra. **mathematical blog.[Online]. Available: https://graphicallinearalgebra.net**.

STEWART, M. Rank decisions in matrix quotient decompositions. **SIAM Journal on Matrix Analysis and Applications**, SIAM, v. 37, n. 4, p. 1729–1746, 2016.

ZANASI, F. **Interacting Hopf Algebras: the theory of linear systems**. 177 p. Tese (Doutorado) — Ecole Normale Supérieure de Lyon, 2015.

ZASSENHAUS, H. The sum intersection method. **Manuscript, Ohio State Univ., Columbus**, 1966.