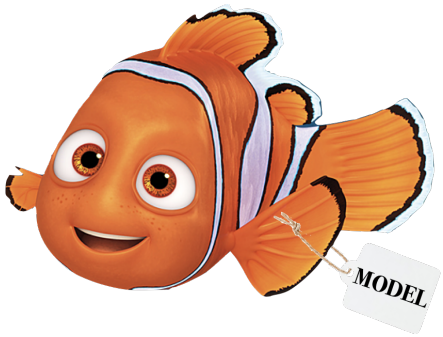
INSTITUTO SUPERIOR TÉCNICO

2ND CYCLE INTEGRATED PROJECT IN APPLIED MATHEMATICS AND COMPUTATION

ALGEBRAIC MACHINE LEARNING

Finding Models



Author
Miguel Bernardo
Guerreiro Pereira

Supervisor
Dr. Gonzalo de Polavieja
Co-Supervisor
Prof. Pedro A. Santos

July 29, 2022

TÉCNICO LISBOA

Champalimaud Foundation

Abstract

Traditional Machine Learning frameworks, such as supervised learning, have as main objective the minimization of some kind of error function. In some approaches, this objective, in combination with other factors, can lead to the model overfitting the data so that the error is smaller, and so it may lead to a weak ability for generalization (in a broader sense, learning). Algebraic Machine Learning proposes a new approach to the process of learning using only discrete structures and algebraic manipulations that try to orchestrate the construction of a model that fits the data. This work briefly explains the context of this kind of learning with a very simple example, the 2×2 squares with black or white cells. This toy-example leads to the main problem addressed in this work, which was the generation of simple examples that fit the model provided by algebraic machine learning. This problem appears in several scientific areas and is known as the Minimal Hitting Set problem. Since there was a vast number of different areas where this problem arises, each one with different heuristic solutions, only one approach was analysed. The chosen approach was STACCATO which was one of the most promising in our scenario. For that purpose, this algorithm was implemented and tested with three different subset models.

Contents

Introduction **3**
Concepts and Ideas . . . 3

How to Learn **9**
Illustrative Example . . . 11
Initial Model . . . 11
1^{st} Training Image . . . 11
2^{nd} Training Image . . . 12
3^{rd} Training Image . . . 13
4^{th} Training Image . . . 14
5^{th} Training Image . . . 15
6^{th} Training Image . . . 16
7^{th} Training Image . . . 17

Extracting Rules **18**
The Minimal Hitting Set Problem . . . 21
The Set Cover Problem . . . 22
STACCATO . . . 23
Hitman . . . 26

Discussion and Final Thoughts **28**

Introduction

The concept of learning has occupied many scientists over the past century onward. Both its definition and the way it materializes are features that have been explored in many ways and in several fields such has philosophy, psychology and biology. Given the fact that, for centuries, the only system recognized to be able to learn was our own brain, it was our process of learning that gathered the most attention in this field. However, in the more recent decades, that same concept has had to been given more broad definitions with the surge of artificial intelligence systems. With inspirations from various origins such as biological scenarios or statistical perspectives, scientists have been able to create structures, such as the *perceptron*, that exhibit forms of learning. The development of most of these structures and their respective computational implementation are the current tasks in the field known as Machine Learning. This area has had very successful applications to a very wide range of problems. However, one of the most vastly used techniques nowadays, Deep Neural Networks, presents a challenge to the explainability of the predictions being made [8]. When dealing with critical applications such as medical diagnosis or autonomous vehicle driving it is not enough to have a very accurate model since experts and decision-makers require more information than just the final decision outputted by the model. With the use of $DNNs$, we rely on architectures with hundreds of layers and millions of parameters [8], making them black-box models from which we cannot retrieve any direct understanding of the mechanism that is producing the results, other than the one we have about the network structure.

A new (formal) approach to the concept of learning (in a more computational framework) has been proposed in [1] and is called Algebraic Machine Learning. In contrast with $DNNs$, this approach might not be as efficient but enjoys a great level of interpretability. As a contrast to the previously mentioned well-known techniques, AML does not seek any error minimization. The main idea behind this paradigm is to be able to symbolically represent and add "knowledge" along the learning process without assuming any property underlying the learned data with the exception that it can, at least, be represented with the use of constants, which will be defined further ahead.

In a very summarized way, the main goal is to embed a problem in an algebraic structure, a *semilattice*, and make use of this representation, through the use of some algebraic operations, to reach a model that satisfies that same problem.

Concepts and Ideas

The concept of model used here comes from its notion in Model Theory. This subject focuses on the study of mathematical structures through the analysis of the first-order sentences that are true in those structures and the sets that can be defined using first-order formulas [6].

In model theory, each structure has an underlying first-order language which is the environment where that structure can be expressed in. That language can be defined as follows:

Definition A language $\mathcal{L}$ is defined by a triple $(\mathcal{C}, \mathcal{F}, \mathcal{R})$ where:

- $\mathcal{C}$ is a set of constant symbols (often represented by latin letters)
- $\mathcal{F}$ is a set of function symbols
- $\mathcal{R}$ is a set of relation symbols

Also, for each function symbol $f \in \mathcal{F}$ there is a corresponding positive integer n_f which indicates the arity of that function symbol. Similarly, for each function symbol $r \in \mathcal{R}$ there is a positive integer n_r that indicates the arity of that relation symbol.

Now, we can define the notion of structure using the concepts of a language. Thus, a structure is defined as:

Definition A structure $\mathcal{M}$ over the language $\mathcal{L}$, is a triple (M, F, R) where:

- M is a non-empty set, usually called universe or domain, containing an element c^M for each $c \in \mathcal{C}$
- F is a set containing an element f^M for each $f \in \mathcal{F}$
- R is a set containing an element r^M for each $r \in \mathcal{R}$

The elements c^M, f^M, r^M are called the interpretations of the symbols c, f and r respectively for every $c \in \mathcal{C}$, $f \in \mathcal{F}$ and $r \in \mathcal{R}$.

Now that we can identify structures it's easy to introduce the most used concept from here onwards which is the concept of **model**. However, first we need to define what is a theory and then relate that to the models of that theory.

Definition A theory $\mathcal{T}$ over the language $\mathcal{L}$ is a set of sentences (formulas where each variable is in the scope of some quantifier) written with the symbols from $\mathcal{L}$.

Definition If $\mathcal{L}$ is a language and $\mathcal{T}$ is a theory over $\mathcal{L}$, then a structure $\mathcal{J}$ is a model of $\mathcal{T}$ if $\mathcal{J} \vDash p$ for every sentence $p \in \mathcal{T}$.

This means that a structure becomes a model for a theory if it "obeys" the rules of that theory. And the rules are given by first-order logical formulas where each variable is within the scope of some quantifier.

In the context of this work, it follows that the structures we'll be interested in are (*atomized*) *semilattices*. Using the previous notation, the *semilattice* structure can be represented as the triple (X_c,F,R) where X_c is a set of constants, the interpretation of the function symbol is $F = \{\odot\}$ and the interpretation of the relation symbol is $R = \{<\}$.

In order to make sense of these symbols, it's useful to define what is a *semillatice*. For that purpose we describe the *theory of semilattices* using first-order formulas involving the interpretation of the functional symbol, $\odot$.

Definition A *semilattice* is a set , X, which is equipped with an operation $\odot$ that is:

$$\cdot \ \textit{commutative}\text{: } \forall x, y \in X : x \odot y = y \odot x$$

$$\cdot \ \textit{associative}\text{: } \forall x, y, z \in X : x \odot (y \odot z) = (x \odot y) \odot z$$

$$\cdot \ \textit{idempotent}\text{: } \forall x \in X : x \odot x = x$$

In the following, this operation will be called "merge" or "idempotent summation", as the authors call it in [1]. Also, an important feature of this structure is the inclusion relation induced by the merge operation.

The inclusion relation

The fact that this operator, $\odot$, is idempotent results in the establishment of an inclusion relation "$<$" between the elements of the *semilattice* (which will be called constants). This inclusion is defined by: $a < b$ if and only if $a \odot b = b$, where a and b are elements of the *semilattice*. This means

that a "is in" b if and only if, when we merge a and b, we get b.

Example Consider the *semilattice* spanned by the set of constants $X_c = \{c_1, c_2, c_3\}$ (every element is a merge of constants in X_c). Using the notation $\rightarrow$ to represent the inclusion relation $<$, we can construct the following graph:

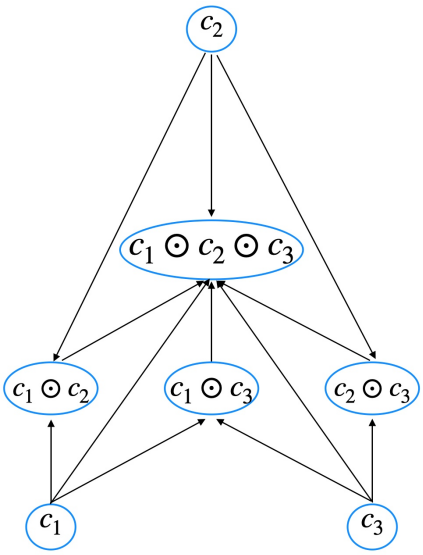


Figure 1: Semilattice Partial Order

On one hand, this graph illustrates how this inclusion relation allows us to view this *semilattice* as a partially ordered set (a set on which there is an order relation between some pairs of elements but not between others). Also, by inspection of this diagram, one can notice that each element can be uniquely represented by a set of constants. Taking this representation one step further, one could also represent each constant by a set of elements (from another kind). By calling these new elements, *atoms*, we can now define an enhanced kind of *semilattice* called *atomized semilattice* that, in addition, extends the partially ordered set that is already in place.

Definition If $M = (X, \{\odot\}, \{<\})$ is a model for the *theory of semilattices*, we say that M is *atomized* if $X = X_c \cup X_a$ where X_c is a set of constants, as before, for which the sentences that describe $\odot$ still hold true and X_a is a set of atoms that extend the partial order already defined for the elements in X_c. This *atomized semilattice* must also satisfy the following properties [2]:

1. For every atom $\phi \in X_a$ there is a constant $c \in X_c$ such that $\phi < c$
2. For every atom $\phi \in X_a$ and constant $c \in X_c$ we have $c \nleqslant \phi$
3. For all constants $c, d \in X_c$ we have $c \leqslant d$ if and only if there is no atom $\phi \in X_a$ such that $\phi < c$ and $\phi \nleqslant d$
4. For every atom $\phi \in X_a$ we have that $\phi < c \odot d$ is true if and only if $\phi < c$ or $\phi < d$
5. For every constant $c \in X_c$, if $\phi < c$ is equivalent to $\psi < c$ then $\phi = \psi$
6. For each constant $c \in X_c$ there is an atom $\phi \in X_a$ such that $\phi < c$

Therefore, considering the previous case, but now allowing for the representation of each constant by a set of atoms, let's consider for example that $c_1 = \{\phi_1\}$, $c_2 = \{\phi_{2a}, \phi_{2b}\}$ and $c_3 = \{\phi_3\}$. Then the extended partial order can be pictured as:

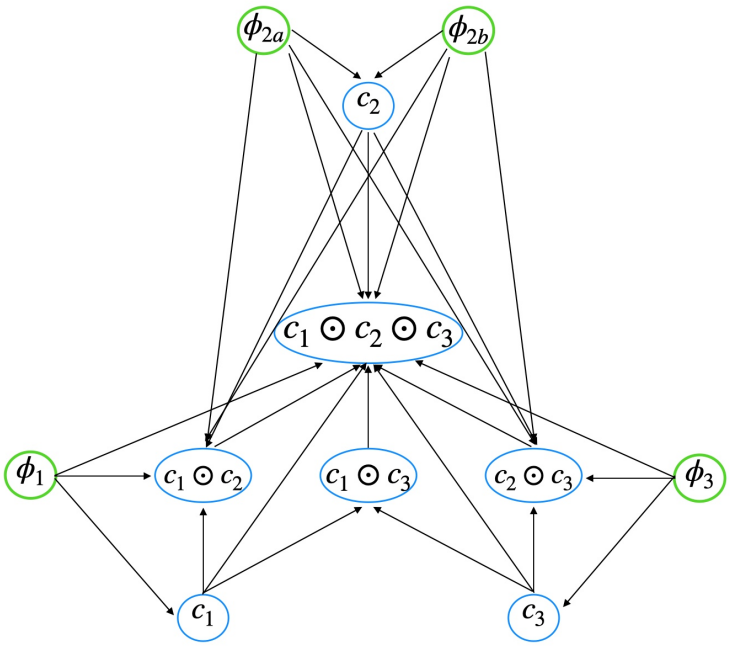


Figure 2: (Almost) Atomized Semilattice Partial Order

Observing this diagram one can notice that atoms ϕ_{2a}, ϕ_{2b} are essentially the same element since they have the exact same relations with the other elements. Actually, in the light of property 5, we reach the conclusion that $\phi_{2a} = \phi_{2b}$, since that $\phi_{2a} < c_2 \Leftrightarrow \phi_{2b} < c_2$. This example ellucidates the fact that each atom can be described by a **unique** set of constants.

Given the simplicity of this example it's easy to check that the rest of the properties above mentioned for *atomized semilattices* hold. Nevertheless, it's important to emphasise that the sets of atoms that define each constant could also have non-empty intersections and could have different cardinalities than the ones in this case, as for example if $c_1 = \{\phi_1, \phi_{13}\}$, $c_2 = \{\phi_2\}$ and $c_3 = \{\phi_3, \phi_{13}\}$:

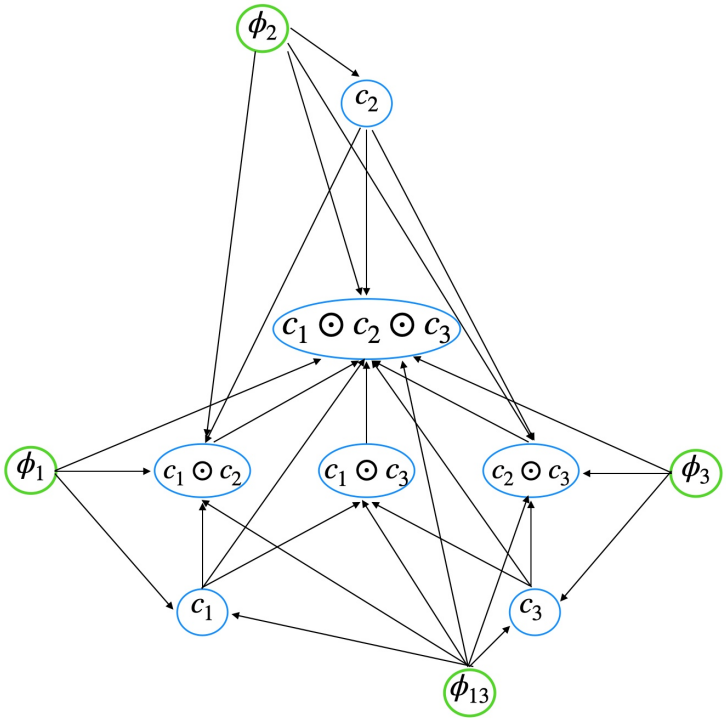


Figure 3: Alternative Atomized Semilattice Partial Order

Now, this alternative example does actually depict an *atomized semilattice* since all properties are satisfied. Therefore we can see the atoms as unique sets of constants: $\phi_1 = \{c_1\}, \phi_{13} = \{c_1, c_3\}$ and $\phi_2 = \{c_2\}$. In this figure, as in the others of the same kind, we see that the graph is transitively closed (if $a \to b$ and $b \to c$ then $a \to c$), which illustrates the transitive property of the inclusion relation.

An *atomized semilattice*, however, is not an extension for the underlying *semilattice* because the "merge" operation is not defined for atoms, thus, its properties don't generally hold in the set $X = X_a \cup X_c$.

The *atomization* of the *semilattice* is useful because it allows to represent each constant by a set of atoms. This representation can change along the training epochs so that it can incorporate knowledge about the images that are provided, by allowing to carry out selected properties from model to model [2].

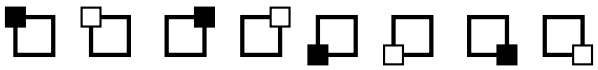
Then, the model we'll be searching for is a specific *atomized semilattice* structure, with a set of constants and atoms, that satisfies **all** the formulas in the *theory of semilattices* extended with the set of literals which will be the training images [1].

One of the examples used so far for explaining the strategies of algebraic machine learning is the toy example of squares with black and white cells. These squares can have dimensions $n \times n$ but to facilitate the explainability here, the smaller cases of $n = 2, 3, 4$ are used. The following exposition of concepts and the example related with the 2×2 case was mostly inspired by the video in [5].

In order for the learning to occur we'll need three core elements, which are **constants**, **terms** and **atoms**. Although some of these elements have already been mentioned, it follows now a brief explanation of their meaning.

Constants

These are symbols to refer to things in the world or to abstract concepts we wish to represent. In the 2×2 black and white squares, the constants can be pictorially represented as:



or just as c_1, c_2, c_3, c_4, c_5, c_6, c_7 and c_8 respectively.

Terms

These are formed by operating constants with the merge operation $\odot$. For example, we can create the following terms:

$T^+ =$  $= c_1 \odot c_5 \odot c_8 \odot c_4$

$T^- =$  $= c_2 \odot c_5 \odot c_8 \odot c_3$

Using the notations defined above, such as the "merge" operation and the inclusion relation, one can make statements about figures such as this ones.

Example: $c_1 < T^+$ since $c_1 \odot T^+ = c_1 \odot (c_1 \odot c_5 \odot c_8 \odot c_4) = c_1 \odot c_5 \odot c_8 \odot c_4 = T^+$

Atoms

These are elements created by the learning algorithm in the construction of the *atomized semilattice* and they'll be represented by greek letters. The name of these elements fits quite well because everything we ought to express in our model's universe will be representable by a set of atoms, which yields a nice parallelism to our "physical" universe.

Given the correspondence between a *semilattice* and a *partially ordered set*, and inspired by the definition in [7], one can define an atom of a *semilattice/partially ordered set* X with a zero element, $\ominus_c$ (the one that is included in all others, $\forall s \in X$: $\ominus_c \leq s$). Thus, $a \in X$ is an atom if, for all $b \in X$: $\ominus_c < b \leq a$ implies $b = a$. Equivalently, $a \in X$ is an atom if $\ominus_c < a$ and there is no $x \in X$ such that $\ominus_c < x < a$.

In the examples above, the *atomization* of the *semilattice* didn't include the zero atom, $\ominus_c$, so that the diagram didn't get too cumbersome, but it can be assumed to be there. As developed in [2], the presence of this zero atom in the *atomization* does not change the *semilattice* formed by the constant elements but is useful to define the concept of *atom*.

According to the axiomatization in [2], which was described above, in an *atomized semilattice*, each regular element (constant) will have at least one atom "inside". Also, since each constant can be represented by the set of atoms it contains, we have that the atoms that represent a constant k are the atoms $\phi \in X_a$ such that $\phi < k$.

Given a model M (for the *theory of semilattices*) and its universe set $X = X_c \cup X_a$, with X_c being the set of constants and X_a the set of atoms that are in the universe, we can define the set of atoms of a term t as:

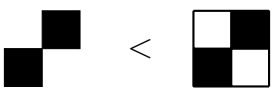
$$Atoms(t) = \{\phi \in X_a : \exists c \in X_c \text{ such that } \phi < c \wedge c < t\}$$

Because of the previous description of terms and atoms, we can state that, for two terms t_1 and t_2, we have the following property:

$$t_1 < t_2 \Leftrightarrow Atoms(t_1) \subseteq Atoms(t_2)$$

Intuitively this means that t_1 is in t_2 if and only if the atoms that "compose" t_1 are in the "composition" of t_2.

Example:



On the left we have a term whose constants are c_{12b} and c_{21b}. Thus, the atoms in that term are $Atoms(c_{12b}) \cup Atoms(c_{21b})$. On the other hand, for the same reason, the atoms in the term on the right are $Atoms(c_{11w}) \cup Atoms(c_{12b}) \cup Atoms(c_{21b}) \cup Atoms(c_{22w})$, which clearly contains the atoms from the term on the left.

How to Learn

Imagining we want to input 2×2 images of black and white cells as examples of a certain class, the point is that the learning algorithm should try to find a model of the theory of semilattices extended with the training set (the images we gave it) that satisfies the relations between each atoms and constants provided by the training.

For the sake of simplicity, in this case we only present the input of positive relations in the learning process and therefore don't show to the algorithm examples of images that do not have the desired characteristic. Hereby we will just input examples of images that are in the positive class (the class that has the desired property). It is proven in [1] that with a sufficient amount of examples, the learning algorithm will find the rule underlying those examples.

In the following toy-example, that is elaborated from the one in [5], we provide examples of squares that have a rule we already know, which in practice won't happen. However, this example is useful for explaining the learning process in Algebraic Machine Learning.

Before proceeding it's useful to explain a few concepts that will be used in the training procedure.

Freest Model

This is the kind of model we are looking for and it is defined as the model that accepts all the given images as being in the positive class and rejects all other images. This is the model that guarantees we can find the rule (given enough examples).

The property we search for:

In the case treated ahead the property p we're looking for is "having a black vertical bar". We create a constant p to describe the property underlying the positive examples. Thus, for every positive example (or image) T_i we'll have:

$$p < T_i$$

which means that $p \odot T_i = T_i$.

The goal is to find the atoms that make up p after we've inputted all the data. The following task is to make sense of the atoms that are in p so that we can understand what they mean. Here, the meaning should be the rule.

Full-Crossing

The construction of the desired model is done using an algebraic operation called Full-Crossing that creates new atoms from the previous ones and from the inputted terms (images). These new atoms and their corresponding relations with the constants of the model will be transformed along the training process to reproduce the rule underlying the data. For two sets of atoms, A and B, the **Full-Crossing** between A and B can be thought of has the cartesian product of the atoms in A with the atoms in B, i.e. for every pair $(\alpha,\beta) \in A \times B$ the full-crossing produces a new atom γ such that γ is the union of α and β (γ is in every constant where α and β are).

In order to produce smaller models that still keep all the relevant information, it is useful to remove atoms that give us information that is already stored in other atoms of the same model. For this purpose the authors in [2] describe what is a redundant atom. To do that, it is first useful to define when is an atom wider than another atom.

Wider atom

We say an atom ϕ is **wider** than atom η if it is different than η and for every constant c:

$$(\eta < c) \implies (\phi < c)$$

Thus, in a more intuitive manner, an atom is wider than another atom if it is in the same constants as the other and also in at least one more (so that they're different). Hence, we can now define a redundant atom.

Redundant atom

An atom ϕ is said to be **redundant** with a model M when for each constant c in that model such that $\phi < c$, there is at least one atom $\eta < c$ in M such that ϕ is **wider** than η.

Given all these concepts presented before we can now sketch the algorithm used to perform the training process using this algebraic operation.

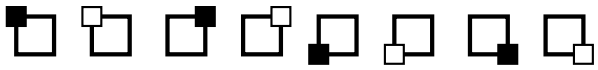
Input: Set of constants $\{c_i : i \in I\}$ and Set of Training Examples, $\{T_i : i \in I'\}$, where I and I' are finite indexings
Output: Freest Model

- Start with the initial model, M, where each constant has one corresponding atom
- For each Training Example, T_i:
  - Compute the set of atoms from the current model that are in the constants that compose the current training image, let that set be A_{image}
  - Compute the atoms from the current model that are in p and not in the constants that compose the current training image, let that set be $A_{p,\neg image}$
  - If $A_{p,\neg image} \neq \varnothing$:
    - $NewAtoms \leftarrow FullCrossing(A_{image}, A_{p,\neg image})$
    - Create the new model, $NewM \leftarrow M$
    - Remove the atoms in $A_{p,\neg image}$ from $NewM$
    - Add the atoms in $NewAtoms$ to $NewM$
    - Remove redundant atoms from $NewM$
    - $M \leftarrow NewM$

Illustrative Example

Here it's presented the complete training procedure for the 2×2 squares with a black vertical bar. The computations done at each step were done with the help of a Python script [1] exclusively created for this purpose.

Initial Model

constants  p

$\vee \quad \vee \quad \vee \quad \vee \quad \vee \quad \vee \quad \vee \quad \vee \quad \vee$

atoms $\phi_1 \quad \phi_2 \quad \phi_3 \quad \phi_4 \quad \phi_5 \quad \phi_6 \quad \phi_7 \quad \phi_8 \quad \phi_p$

The constants that refer to each pixel of the square can also be named, respectively, $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$. Here, we write that an atom is in a constant as, for example,

$\phi_3 <$  or, alternatively, $\phi_3 \rightarrow$ 

So, in total, we have at first 9 (8+1) atoms.

First Training Image

$T_1 =$ 

In this image we have the atoms ϕ_1, ϕ_4, ϕ_5 and ϕ_8. And to be able to learn we must assert that $p < T_1$, given that $\phi_p < p$.

Thus, the first Full-Crossing will yield the following new atoms:

		Atoms in T_1			
		ϕ_1	ϕ_4	ϕ_5	ϕ_8
Atoms in p and not in T_1	ϕ_p	ϕ_{1p}	ϕ_{4p}	ϕ_{5p}	ϕ_{8p}

Now, this new atoms will replace ϕ_p as the atoms in p and, therefore, are included in the new freest model in such a way that: for every $i \in \{1, 4, 5, 8\}$: $\phi_{ip} < p$ and $\phi_{ip} < c_i$ where c_i is the original constant that has ϕ_i. With this substitution the resulting number of atoms is 9 - 1 + 4 = 12.

New Freest Model:

- $\phi_1 < c_1$
- $\phi_2 < c_2$
- $\phi_3 < c_3$
- $\phi_4 < c_4$
- $\phi_5 < c_5$
- $\phi_6 < c_6$
- $\phi_7 < c_7$
- $\phi_8 < c_8$
- $\phi_{1p} < c_1$
- $\phi_{1p} < p$
- $\phi_{4p} < c_4$
- $\phi_{4p} < p$
- $\phi_{5p} < c_5$
- $\phi_{5p} < p$
- $\phi_{8p} < c_8$
- $\phi_{8p} < p$

[1] The script is available at this GitHub repository: https://github.com/MBernardoPereira/IntegratedProjectIST

Second Training Image

$T_2 =$ 

Clearly this image does not share any constants with the previous image, and thus, the atoms in T_2 will not intersect the ones in T_1.

In this image we have the atoms ϕ_2, ϕ_3, ϕ_6 and ϕ_7 and again we need to assert that $p < T_2$, given that, for every $i \in \{1, 4, 5, 8\}$ we have that $\phi_{ip} < p$.

Thus, the second Full-Crossing will yield the following new atoms:

		Atoms in T_2			
		ϕ_2	ϕ_3	ϕ_6	ϕ_7
Atoms in p and not in T_2	ϕ_{1p}	ϕ_{12p}	ϕ_{13p}	ϕ_{16p}	ϕ_{17p}
	ϕ_{4p}	ϕ_{24p}	ϕ_{34p}	ϕ_{46p}	ϕ_{47p}
	ϕ_{5p}	ϕ_{25p}	ϕ_{35p}	ϕ_{56p}	ϕ_{57p}
	ϕ_{8p}	ϕ_{28p}	ϕ_{38p}	ϕ_{68p}	ϕ_{78p}

Now, this new atoms will replace the atoms previously in p and, therefore, are included in the new freest model in such a way that: for every new atom ϕ_{ijp}, we have $\phi_{ijp} < p$ and $\phi_{ijp} < c_i$ and $\phi_{ijp} < c_j$, where c_i, c_j are the original constants that have the atoms ϕ_i, ϕ_j respectively. In total we end up with 24 (8+16) atoms.

The **New Freest Model**, as the one shown before, can be obtained by asserting: for each new atom in the model, $\phi_{n_1 \ldots n_r p}$, we have that $\phi_{n_1 \ldots n_k p} < c_{n_i}$ for every $i \in \{1, \ldots r\}$ and also $\phi_{n_1 \ldots n_k p} < p$. Taking this in consideration, it is sufficient to present the **New Freest Model** has just the atoms that are in it, since the notation for every atom already indicates the inclusion relations that that atom has with the constants.

New Freest Model:

- ϕ_1
- ϕ_2
- ϕ_3
- ϕ_4
- ϕ_5
- ϕ_6
- ϕ_7
- ϕ_8
- ϕ_{12p}
- ϕ_{13p}
- ϕ_{16p}
- ϕ_{17p}
- ϕ_{24p}
- ϕ_{34p}
- ϕ_{46p}
- ϕ_{47p}
- ϕ_{25p}
- ϕ_{35p}
- ϕ_{56p}
- ϕ_{57p}
- ϕ_{28p}
- ϕ_{38p}
- ϕ_{68p}
- ϕ_{78p}

Third Training Image

$T_3 =$ 

This image is comprised by the constants c_1, c_4, c_5, c_7. Thus, every atom in those constants is also in T_3. Thus, the third Full-Crossing will yield the following new atoms:

		Atoms in T_3							
		ϕ_1	ϕ_4	ϕ_5	ϕ_7	ϕ_{21p}	ϕ_{31p}	ϕ_{61p}	ϕ_{71p}
Atoms in p and not in T_3	ϕ_{28p}	ϕ_{128p}	ϕ_{248p}	ϕ_{258p}	ϕ_{278p}	ϕ_{128p}	ϕ_{1238p}	ϕ_{1268p}	ϕ_{1278p}
	ϕ_{38p}	ϕ_{138p}	ϕ_{348p}	ϕ_{358p}	ϕ_{378}	ϕ_{1238p}	ϕ_{138p}	ϕ_{1368p}	ϕ_{1378p}
	ϕ_{68p}	ϕ_{168p}	ϕ_{468p}	ϕ_{568p}	ϕ_{678p}	ϕ_{1268p}	ϕ_{1368p}	ϕ_{168p}	ϕ_{1678p}

		Atoms in T_3								
		ϕ_{24p}	ϕ_{34p}	ϕ_{64p}	ϕ_{74p}	ϕ_{25p}	ϕ_{35p}	ϕ_{65p}	ϕ_{75p}	ϕ_{78p}
Atoms in p and not in T_3	ϕ_{28p}	ϕ_{248p}	ϕ_{2348p}	ϕ_{2468p}	ϕ_{2478p}	ϕ_{258p}	ϕ_{2358p}	ϕ_{2568p}	ϕ_{2578p}	ϕ_{278p}
	ϕ_{38p}	ϕ_{2348p}	ϕ_{348p}	ϕ_{3468p}	ϕ_{3478p}	ϕ_{2358p}	ϕ_{358p}	ϕ_{3568p}	ϕ_{3578p}	ϕ_{378p}
	ϕ_{68p}	ϕ_{2468p}	ϕ_{3468p}	ϕ_{468p}	ϕ_{4678p}	ϕ_{2568p}	ϕ_{3568p}	ϕ_{568p}	ϕ_{6578p}	ϕ_{678p}

In total we produced now 51 ($17\times$ 3) new atoms. Now, we can remove from the model the atoms in p that were used in the full-crossing, thus we remove ϕ_{28p}, ϕ_{38p} and ϕ_{68p}. However, all the atoms produced with this full-crossing are either repeated or redundant in this model. For example:

- Considering ϕ_{348p}, we have that it appears in the crossing of ϕ_4 and ϕ_{38p} but also, repeated, in the crossing of ϕ_{34p} and ϕ_{38p}. Nevertheless,
  - for constant c_3, we already have atom $\phi_3 < c_3$ in the model and ϕ_{348p} is wider than ϕ_3.
  - for constant c_4, we already have atom $\phi_4 < c_4$ in the model and ϕ_{348p} is wider than ϕ_4.
  - for constant c_8, we already have atom $\phi_8 < c_8$ in the model and ϕ_{348p} is wider than ϕ_8.
  - for constant p, we already have atom $\phi_{38p} < p$ in the model and ϕ_{348p} is wider than ϕ_{38p}.

  Thus, this atom is redundant.
- Considering ϕ_{2568p}, we have that:
  - for constant c_2, we already have atom $\phi_2 < c_2$ in the model and ϕ_{2568p} is wider than ϕ_2.
  - for constant c_5, we already have atom $\phi_5 < c_5$ in the model and ϕ_{2568p} is wider than ϕ_5.
  - for constant c_6, we already have atom $\phi_6 < c_6$ in the model and ϕ_{2568p} is wider than ϕ_6.
  - for constant c_8, we already have atom $\phi_8 < c_8$ in the model and ϕ_{2568p} is wider than ϕ_8.
  - for constant p, we already have atom $\phi_{25p} < p$ in the model and ϕ_{2568p} is wider than ϕ_{25p}.

  So, this atom is also redundant.

Clearly, here the trickiest part is checking the existence of a 'narrower' atom for constant p since for the other constants there is always the corresponding initial atoms ($\phi_i : i = 1, \ldots, 8$) for which the given atom is wider. Since a similar argument as the one applied to these could be applied for the rest of the new atoms, we don't add any of them to the model. This way, we stand with a model which has $24 - 3 = 21$ unique atoms.

New Freest Model:

- ϕ_1
- ϕ_2
- ϕ_3
- ϕ_4
- ϕ_5
- ϕ_6
- ϕ_7
- ϕ_8
- ϕ_{12p}
- ϕ_{13p}
- ϕ_{16p}
- ϕ_{17p}
- ϕ_{24p}
- ϕ_{34p}
- ϕ_{46p}
- ϕ_{47p}
- ϕ_{25p}
- ϕ_{35p}
- ϕ_{56p}
- ϕ_{57p}
- ϕ_{78p}

Fourth Training Image

$T_4 =$ 

This image is composed by the constants c_1, c_3, c_5, c_8. Therefore, we can do the following Full-Crossing:

		Atoms in T_4						
		ϕ_1	ϕ_3	ϕ_5	ϕ_8	ϕ_{12p}	ϕ_{13p}	ϕ_{16p}
Atoms in p and not in T_4	ϕ_{24p}	ϕ_{124p}	ϕ_{234p}	ϕ_{245p}	ϕ_{248p}	ϕ_{124p}	ϕ_{1234p}	ϕ_{1246p}
	ϕ_{46p}	ϕ_{146p}	ϕ_{346p}	ϕ_{456p}	ϕ_{468p}	ϕ_{1246p}	ϕ_{1346p}	ϕ_{146p}
	ϕ_{47p}	ϕ_{147p}	ϕ_{347p}	ϕ_{457p}	ϕ_{478p}	ϕ_{1247p}	ϕ_{1347p}	ϕ_{1467p}

		Atoms in T_4						
		ϕ_{17p}	ϕ_{25p}	ϕ_{34p}	ϕ_{35p}	ϕ_{56p}	ϕ_{57p}	ϕ_{78p}
Atoms in p and not in T_4	ϕ_{24p}	ϕ_{1247p}	ϕ_{245p}	ϕ_{234p}	ϕ_{2345p}	ϕ_{2456p}	ϕ_{2457p}	ϕ_{2478p}
	ϕ_{46p}	ϕ_{1467p}	ϕ_{2456p}	ϕ_{346p}	ϕ_{3456p}	ϕ_{456p}	ϕ_{4567p}	ϕ_{4678p}
	ϕ_{47p}	ϕ_{147p}	ϕ_{2457p}	ϕ_{347p}	ϕ_{3457p}	ϕ_{4567p}	ϕ_{457p}	ϕ_{478p}

Then, we can can construct the **New Freest Model** by removing the atoms ϕ_{24p}, ϕ_{46p} and ϕ_{47p} from the model and adding their children which are the atoms produced by the full-crossing that are not repeated nor redundant in this model. These atoms are ϕ_{248p} and ϕ_{468p}. As hinted before, this is because we can't find any atoms in p such that these atoms are wider than them. Using the same sketch as before, but now to prove the non-redundancy of atoms, we obtain:

- For the atom ϕ_{248p}:
  - for constant c_2, we already have atom $\phi_2 < c_2$ in the model and ϕ_{248p} is wider than ϕ_2.
  - for constant c_4, we already have atom $\phi_4 < c_4$ in the model and ϕ_{248p} is wider than ϕ_4.
  - for constant c_8, we already have atom $\phi_8 < c_8$ in the model and ϕ_{248p} is wider than ϕ_8.
  - for constant p, we can't find any atom such that ϕ_{248p} is wider than, because we already eliminated ϕ_{24p}.

  Thus, this atom is not redundant.

- For the atom ϕ_{468p}:
  - for constant c_4, we already have atom $\phi_4 < c_4$ in the model and ϕ_{468p} is wider than ϕ_4.
  - for constant c_6, we already have atom $\phi_6 < c_6$ in the model and ϕ_{468p} is wider than ϕ_6.
  - for constant c_8, we already have atom $\phi_8 < c_8$ in the model and ϕ_{468p} is wider than ϕ_8.
  - for constant p, we can't find any atom such that ϕ_{248p} is wider than, because we already eliminated ϕ_{46p}.

  So, this atom is also not redundant.

Hence, we add these atoms to the new model. It's worth noting that the atom ϕ_{248p} was previously rejected because it was redundant with other atoms in the model. However, in the meantime, those atoms have been erased and now this atom is no longer redundant. At this point we have a model with $21 - 3 + 2 = 20$ atoms.

New Freest Model:

- ϕ_1
- ϕ_2
- ϕ_3
- ϕ_4
- ϕ_5
- ϕ_6
- ϕ_7
- ϕ_8
- ϕ_{12p}
- ϕ_{13p}
- ϕ_{16p}
- ϕ_{17p}
- ϕ_{34p}
- ϕ_{25p}
- ϕ_{35p}
- ϕ_{56p}
- ϕ_{57p}
- ϕ_{78p}
- ϕ_{248p}
- ϕ_{468p}

Fifth Training Image

$T_5 =$ 

This image is composed by the constants c_2, c_3, c_5, c_7. Thus, every atom in those constants is also in T_5.

Doing the Full-Crossing again will yield the following new atoms:

		Atoms in T_5						
		ϕ_2	ϕ_3	ϕ_5	ϕ_7	ϕ_{12p}	ϕ_{13p}	ϕ_{17p}
Atoms in p and not in T_5	ϕ_{16p}	ϕ_{126p}	ϕ_{136p}	ϕ_{156p}	ϕ_{167p}	ϕ_{126p}	ϕ_{136p}	ϕ_{167p}
	ϕ_{468p}	ϕ_{2468p}	ϕ_{3468p}	ϕ_{4568p}	ϕ_{4678p}	ϕ_{12468p}	ϕ_{13468p}	ϕ_{14678p}

		Atoms in T_5						
		ϕ_{25p}	ϕ_{34p}	ϕ_{35p}	ϕ_{56p}	ϕ_{57p}	ϕ_{78p}	ϕ_{248p}
Atoms in p and not in T_5	ϕ_{16p}	ϕ_{1256p}	ϕ_{1346p}	ϕ_{1356p}	ϕ_{156p}	ϕ_{1567p}	ϕ_{1678p}	ϕ_{12468p}
	ϕ_{468p}	ϕ_{24568p}	ϕ_{3468p}	ϕ_{34568p}	ϕ_{4568p}	ϕ_{45678p}	ϕ_{4678p}	ϕ_{2468p}

Now we can remove the atoms ϕ_{16p} and ϕ_{468p} from the model and add the non-repeated and non-redundant atoms produced in this full-crossing. Yet, just like with the training image T_3 all these new atoms are either repeated or redundant, so we don't add any of them once again. Thus the current model is updated to the following model that has $20 - 2 = 18$ atoms.

New Freest Model:

- ϕ_1
- ϕ_2
- ϕ_3
- ϕ_4
- ϕ_5
- ϕ_6
- ϕ_7
- ϕ_8
- ϕ_{12p}
- ϕ_{13p}
- ϕ_{17p}
- ϕ_{34p}
- ϕ_{25p}
- ϕ_{35p}
- ϕ_{56p}
- ϕ_{57p}
- ϕ_{78p}
- ϕ_{248p}

Sixth Training Image

$T_6 =$ 

This image is composed by the constants c_1, c_3, c_6, c_7. Thus, every atom in those constants is also in T_6.

		Atoms in T_6					
		ϕ_1	ϕ_3	ϕ_6	ϕ_7	ϕ_{12p}	ϕ_{13p}
Atoms in p	ϕ_{25p}	ϕ_{125p}	ϕ_{235p}	ϕ_{256p}	ϕ_{257p}	ϕ_{125p}	ϕ_{1235p}
and not in T_6	ϕ_{248p}	ϕ_{1248p}	ϕ_{2348p}	ϕ_{2468p}	ϕ_{2478p}	ϕ_{1248p}	ϕ_{12348p}

		Atoms in T_6					
		ϕ_{17p}	ϕ_{34p}	ϕ_{35p}	ϕ_{56p}	ϕ_{57p}	ϕ_{78p}
Atoms in p	ϕ_{25p}	ϕ_{1257p}	ϕ_{2345p}	ϕ_{235p}	ϕ_{256p}	ϕ_{257p}	ϕ_{2578p}
and not in T_6	ϕ_{248p}	ϕ_{12478p}	ϕ_{2348p}	ϕ_{23458p}	ϕ_{24568p}	ϕ_{24578p}	ϕ_{2478p}

Repeating the same procedure as always, we remove ϕ_{25p} and ϕ_{248p} from the model and add the "relevant" new atoms to the model, which in this is just the atom ϕ_{2468p}.

Then, the **New Freest Model** has now $18 - 2 + 1 = 17$ atoms:

- ϕ_1
- ϕ_2
- ϕ_3
- ϕ_4
- ϕ_5
- ϕ_6
- ϕ_7
- ϕ_8
- ϕ_{12p}
- ϕ_{13p}
- ϕ_{17p}
- ϕ_{34p}
- ϕ_{35p}
- ϕ_{56p}
- ϕ_{57p}
- ϕ_{78p}
- ϕ_{2468p}

Seventh Training Image

$T_7 =$ 

This last image is composed by the constants c_1, c_3, c_5, c_7. Thus, every atom in those constants is also in T_7. Using this information we compute one last time the full-crossing.

		Atoms in T_7					
		ϕ_1	ϕ_3	ϕ_5	ϕ_7	ϕ_{12p}	ϕ_{13p}
Atoms in p and not in T_7	ϕ_{2468p}	ϕ_{12468p}	ϕ_{23468p}	ϕ_{24568p}	ϕ_{24678p}	ϕ_{12468p}	$\phi_{123468p}$

		Atoms in T_7					
		ϕ_{17p}	ϕ_{34p}	ϕ_{35p}	ϕ_{56p}	ϕ_{57p}	ϕ_{78p}
Atoms in p and not in T_7	ϕ_{2468p}	$\phi_{124678p}$	ϕ_{23468p}	$\phi_{234568p}$	ϕ_{24568p}	$\phi_{245678p}$	ϕ_{24678p}

Lastly, we remove ϕ_{2468p} from the model and since, once again, we didn't produce non-redundant atoms, we don't add anything to the final model. This model has $17 - 1 = 16$ atoms.

Final Freest Model

- ϕ_1
- ϕ_2
- ϕ_3
- ϕ_4
- ϕ_5
- ϕ_6
- ϕ_7
- ϕ_8
- ϕ_{12p}
- ϕ_{34p}
- ϕ_{56p}
- ϕ_{78p}
- ϕ_{13p}
- ϕ_{17p}
- ϕ_{35p}
- ϕ_{57p}

After these additions and the corresponding full-crossings, we end up with the **Freest Model**, that accepts every image that has at least one black vertical bar and reject all others. According to [2], this model is guaranteed to find the rule that underlies all this given data. That rule is "hidden" in the atoms that are in the constant p.

Using the notation above mentioned, these atoms will be something like ϕ_{wp} where w is a sequence of numbers, each one representing a constant where that atom belongs to.

In this case the atoms in p are ϕ_{12p}, ϕ_{34p}, ϕ_{56p}, ϕ_{78p}, ϕ_{13p}, ϕ_{17p}, ϕ_{35p} and ϕ_{57p}.

The problem with this set of atoms is that it doesn't give us explicitly the rule but rather in a disguised way.

Extracting Rules

In the previous learning example, the fundamental operation being done over the atoms to produce knowledge was the Full-Crossing algorithm. This algorithm produces many atoms since it produces an atom for every pair of atoms in the Cartesian product of the sets of atoms inputted (albeit some might be repeated). This makes it less practical to use in larger scale applications (which is not the case for this toy-example). For this reason, in reality, a different type of crossing is used, Sparse-Crossing, which is more efficient computationally-wise and produces smaller models in general [1].

Nevertheless, with this machinery at hands, if one would try to use algebraic machine learning to figure out which unknown pattern was common in a certain set of black and white squares, one most certainly wouldn't have access to the complete set of examples that exhibit that same pattern. Fortunately, as it is explained in [1], the atoms that are in the interesting part of the model appear in a very early stage of the training process and seem to be the more stable along the whole training process. This makes it profitable to stop the training earlier and still keep a very good model for the given data. Knowing this fact, and keeping in mind the objective above, one would train the Algebraic Machine Learning system with a sufficient amount of positive examples and not the complete set of them, and then would end-up with a certain model. In this model, one should look for the atoms that are in constant p, specially the smaller ones which are more likely to be generalizations, and with these atoms the goal is, then, to find the smallest set of constants that contain those atoms.

Since the training example here is so small, we show ahead the case of the **Freest Model** obtained after inputting **all** possible positive examples of the class "2×2 images with at least one vertical bar". Then, we show what problem we need to solve after we've found this model.

As was seen above in the Illustrative Example, a model for the theory of atomized semilattices can be described by the inclusion relations between the atoms and the constants. Thus, when evaluating a model we will just be analysing which atoms are in each constant. Since the positive examples are added in the learning process as "being in constant p", the underlying rule among them will be hidden in the atoms that are in constant p at the end of the training process. To find the rule, then, we want to find what set of constants need to be in p so that we can construct the prototypical image(s) that has the given property. But to find this minimal set of constants we need to check in which constants are the atoms that are also in constant p.

In the next figure we present the Final Freest Model for the example detailed before.

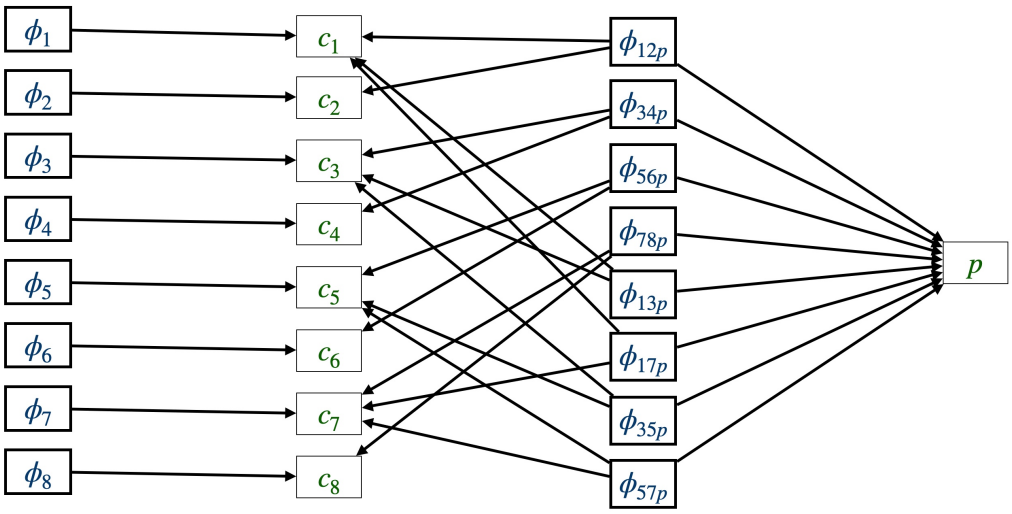


Figure 4: Diagram representation of the Final Freest Model

The prototypical image has all the atoms that are in p. And, by [1], p has an atom ϕ if and only if p contains any of the constants that contain ϕ (excluding p, since that wouldn't provide any new information). As can be seen, the atoms in the left, which are the initial ones, are not useful to obtain the rule since they aren't in p. Thus, we only need to analyse the atoms that are in p and the constants where they belong.

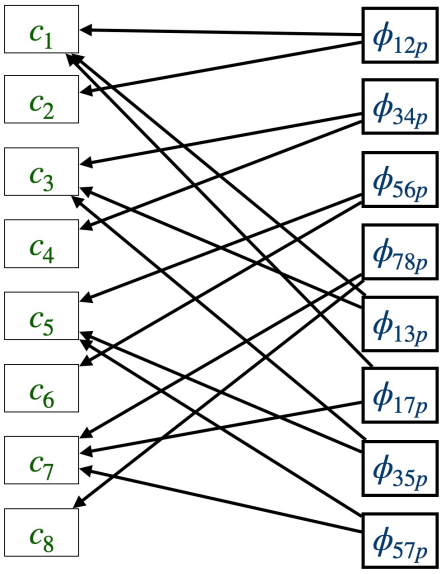
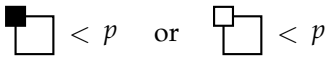
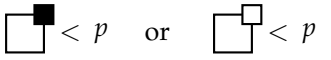
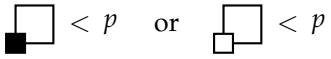
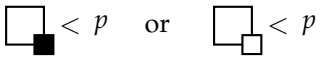


Figure 5: Only the atoms that are in p

Now we see that we have two types of atoms: the ones in "consecutive" and disjoint sets of constants and the others. Analysing the first kind of atoms we get that:

- $\phi_{12p} < p \Leftrightarrow c_1 < p$ or $c_2 < p \quad \Leftrightarrow$  $< p$ or $< p$
- $\phi_{34p} < p \Leftrightarrow c_3 < p$ or $c_4 < p \quad \Leftrightarrow$  $< p$ or $< p$
- $\phi_{56p} < p \Leftrightarrow c_5 < p$ or $c_6 < p \quad \Leftrightarrow$  $< p$ or $< p$
- $\phi_{78p} < p \Leftrightarrow c_7 < p$ or $c_8 < p \quad \Leftrightarrow$  $< p$ or $< p$

Since all these four bullets must hold, then this solely means that "every image that has the rule is an image". This may sound redundant but actually it is just the assertion that every image in the positive class has either a black or a white square in each cell. For this reason, now we restrict the analysis to the other kind of atoms, which we call the *interesting* atoms.

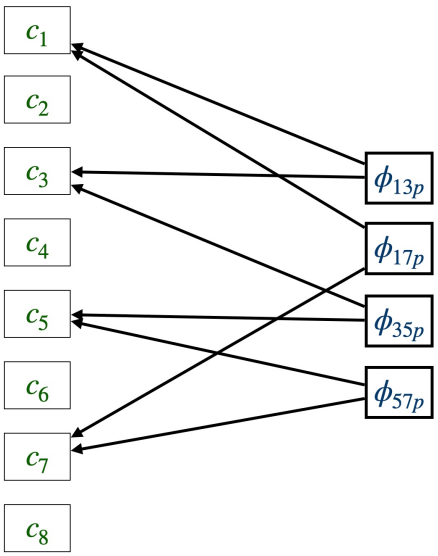
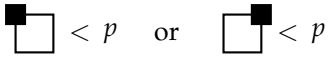
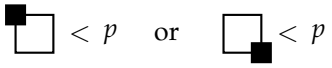


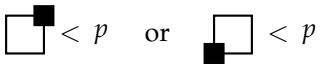
Figure 6: Only the Interesting atoms

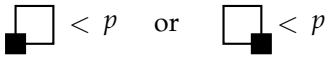
Observing these atoms we see that none of them is in a constant that represents a white cell (c_2, c_4, c_6 and c_8) which indicates that the rule must not involve white cells.

Applying the same reasoning as before we get that:

· $\phi_{13p} < p \Leftrightarrow c_1 < p$ or $c_3 < p$ $\Leftrightarrow$  or

· $\phi_{17p} < p \Leftrightarrow c_1 < p$ or $c_7 < p$ $\Leftrightarrow$  or

· $\phi_{35p} < p \Leftrightarrow c_3 < p$ or $c_5 < p$ $\Leftrightarrow$  or

· $\phi_{57p} < p \Leftrightarrow c_5 < p$ or $c_7 < p$ $\Leftrightarrow$  or

So, combining all this conclusions we obtain:

$$(c_1 < p \vee c_3 < p) \wedge (c_1 < p \vee c_7 < p) \wedge (c_3 < p \vee c_5 < p) \wedge (c_5 < p \vee c_7 < p)$$

which is a logical formula in Conjunctive Normal Form. For this simple example it is easy to transform this formula into a formula in Disjunctive Normal Form. First, for the sake of simplicity of notation, we omit the "$< p$":

$$(c_1 \vee c_3) \wedge (c_1 \vee c_7) \wedge (c_3 \vee c_5) \wedge (c_5 \vee c_7)$$

Now, it's easy to note that:

$$(c_1 \vee c_3) \wedge (c_1 \vee c_7) \wedge (c_3 \vee c_5) \wedge (c_5 \vee c_7) \vDash (c_1 \wedge c_5) \vee (c_3 \wedge c_7)$$

And this is the rule we are looking for. This rule states that the prototypical image has the constants c_1 and c_5 or the constants c_3 and c_7 (i.e. has a left black vertical bar or a right black vertical bar). A pictorial representation of this rule can be obtained with the disentanglement of the graph shown before with the interesting atoms of the model.

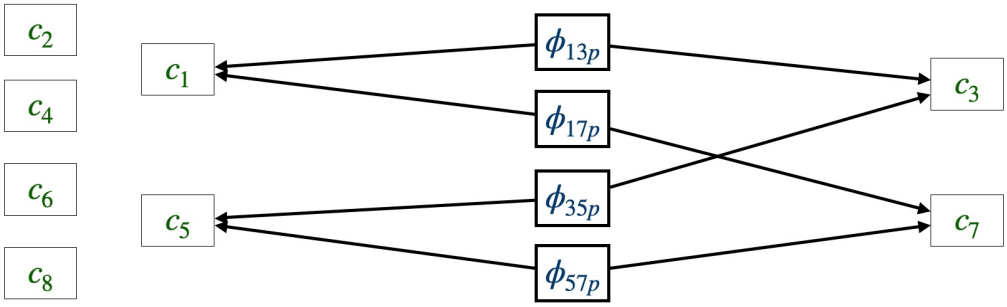


Figure 7: Unfolding of the graph with the interesting atoms

In an exactly analogous way to this 2×2 example, the **Freest Model** found for the 4×4 black and white squares example (which is also depicted in [5]) after performing the training with all possible images with at least one vertical bar, has 304 atoms in total.

Of these 304, 32 are the initial atoms, 16 are atoms that say that each image has in each cell either a black square or a white square, and the rest 256 are the atoms that embody the underlying rule. This atoms can be represented as the merge of the constants they are in:

Although just a few atoms are represented above, the pattern amongst them is that each one has exactly one black cell in each column and every atom is different from the rest. Translating this set of atoms into a logical formula, as was done above, then the conversion of that formula to one in DNF would yield that an image with property p has at least one black vertical bar on one of its columns.

The simplicity of the 2×2 example allowed for a simple retrieval of the rule using just manipulations of logical formulas from the atoms in the model. However, in most useful cases the problem will not be so easy because we'll have many more atoms in the final model.

Despite showing that this problem can be solved using the conversion of logical formulas from CNF to DNF, we can reformulate this problem so that we are able to treat it with some approximation strategy. Therefore this problem can be viewed in the framework of the Minimal Hitting Set Problem.

The Minimal Hitting Set Problem

Let Σ denote the universe of constants and $\{A_i : i = 1, \ldots, k\}$ a collection of k atoms over that universe of constants. Here, an atom is just a set of constants. An hitting set for that collection of atoms is a set of constants $S \subseteq \Sigma$ such that:

$$\forall i \in \{1, \ldots, k\} : S \cap A_i \neq \varnothing$$

That set is minimal if no proper subset of S is also an hitting set, i.e.

$$\forall D \subset S \; \exists i \in \{1, \ldots, k\} : D \cap A_i = \varnothing$$

So, in our example, we want to find a set of constants that hits every atom in our model and that has the least cardinality possible. The problem of finding a minimal hitting set is known to be NP-hard [4], just like the conversion from CNF to DNF. Therefore, for large scale applications one would need a method that uses some kind of heuristic to guide the search for an approximated MHS, despite this heuristics leading sometimes to larger hitting sets. Since this problem emerges in several areas of research, there are different types of algorithms with different approaches to

the construction of MHS's. A list and analysis of some of these approaches is available at [3]. An equivalent formulation for this problem is the Set Cover Problem which is stated as:

The Set Cover Problem

Given an universe of elements $\mathcal{U}$ and a family of subsets $\mathcal{S}$ such that for every $s \in \mathcal{S}$ we have that $s \subset \mathcal{U}$ and $\bigcup_i s_i = \mathcal{U}$. Then, the Set Cover Problem is to find a subset $\mathcal{C} \subseteq \mathcal{S}$, with the least cardinality possible, such that every element in $\mathcal{U}$ belongs to at least one member of $\mathcal{C}$.

This problem was proved to be NP-Complete, and thus NP-hard, in a classical paper by Karp known as "Karp's 21 NP-complete problems" [9].

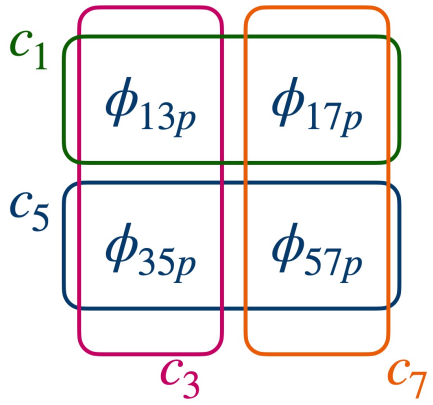


Figure 8: Set Covering representation of the example above

Every instance of a Set Cover Problem can be represented as a (directed) bipartite graph, with the sets represented on one side, the elements represented on the other side, and edges that go from sets to elements. In our case, the translation for this setting forces the constants to be sets of atoms (which, actually, is conceptually concordant with our definition that the atoms are in the constants) and the goal is to find a small set of constants that covers all the atoms. Thus, by swapping the direction of the arrows in figure 6 we can illustrate this representation for the 2×2 toy-example.

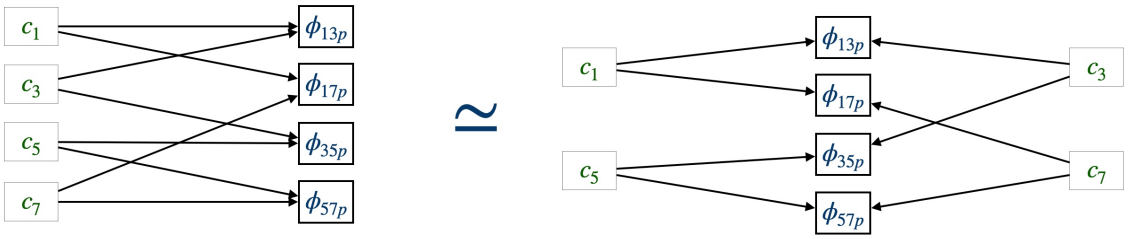


Figure 9: Set Covering bipartite graph representation of the example above

In both these representations of the problem it is clear to observe that the sets $\{c_1, c_5\}$ and $\{c_3, c_7\}$ are Set Covers (with the least cardinality possible).

STACCATO

One of the solutions for the Minimal Hitting Set problem available in the literature was the algorithm STACCATO, which was first proposed in [4]. This algorithm was developed in the context of model-based software fault diagnosis. The original goal was to try to find the minimal set of faulty components in a piece of software given a set of test executions of that software. In each test execution, it is provided the information of which components participated in it and if that execution was successful or not.

The parallel from this context to the context of minimal examples from a given model can be obtained considering the software components as the constants and the test executions as the atoms. In our case the faulty components are the atoms of interest. If we were to apply an algorithm to find the minimal hitting set of a set of atoms produced after a few training epochs then we would have the smaller atoms, that are stable during training, but also some larger atoms that aren't necessarily useful. Thus we could consider a subset of atoms that have smaller sizes, the ones that are more stable, and remove the ones that have an high probability of being in negative examples.

Before explaining the sets of atoms used to retrieve the minimal hitting sets and the respective results, it is worth to briefly explain the algorithm behind STACCATO.

Generally speaking, this algorithm uses the heuristic, $\mathscr{H}$, of "constants that are in the most number of atoms" to guide the search for hitting sets.

Input: Set of Atoms, $\{A_1, \ldots, A_k\}$, largest $|D|$ admissible before last recursion, L, and range of the search, λ
Output: Approximate Minimal Hitting Set D

- Rank the atoms according to the heuristic $\mathscr{H}$, producing the ordered list R
- If there is a constant that takes part in every atom, add it to D
- While the size of D doesn't exceed L, we consider the top λ percentage of constants in R and do the following for each one:
  - Remove the constant from all atoms and remove all atoms that are in that constant
  - Run STACCATO with the new set of atoms
  - Combine the solutions obtained with the current constant and verify if the resulting set is minimal
- The algorithm terminates when the ranking has no more constants left or the limit L has been reached or we've already searched the predefined percentage of constants, λ.

This algorithm was implemented in Python [2] and was tested with two sets of atoms provided by Dr. Gonzalo de Polavieja and also one created by me. These three sets were the following:

- A subset of 37 atoms from an intermediate (with just a few training images inputted) model for the 4×4 black and white squares with at least one black vertical bar.
- The complete (interesting part of the) model for the 3×3 black and white squares with at least one black vertical bar, which contained 27 atoms.
- The complete (interesting part of the) model for the 4×4 black and white squares with at least one black vertical bar, which contained 256 atoms.

Since this algorithm receives as input some tuning parameters, several combinations of these were tested and some relations between those and the running time and correctness of the algorithm were observed. As mentioned in [4], the increase in λ will rise the running time since more constants are being considered but also the decrease in λ implies a less correct output since it may hit very few sets. Thus, a compromise is needed when choosing the optimal value for λ, with the authors suggesting the best value to be 0.5. Also, and as expected, the larger the L, the bigger will be the set outputted. The results for the first set of atoms considered are shown below. As can be seen, in this case, the examples produced don't resemble much of what we would expect since we don't get any black vertical bar. The examples shown were computed with a value for L, which is the limit number of constants allowed to be in the output set before the last recursion, of 4, 5 and 6. By inspection it was noted that with higher values of L we would obtain larger examples, as expected, that had a black vertical bar contained in them. However, we don't aim for the bigger examples because these are the ones that contain more useless information and in the case where we didn't know the rule beforehand we wouldn't be able to discern it from the noise.

Note: The images shown below were produced using the library `matplotlib` from Python and they reflect what was seen before in the sense that: for each cell, the background grey represents no constant in that position, black/white means that there is a black/white square constant in that position and a darker grey means that there is both a black and a white constant in that cell.

[2] The implementation is available at this GitHub repository: https://github.com/MBernardoPereira/IntegratedProjectIST

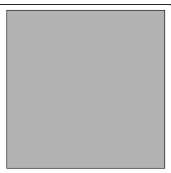
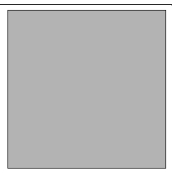
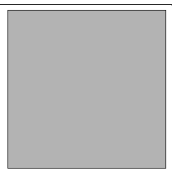
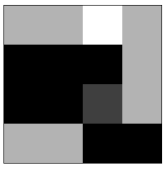
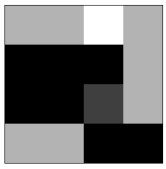
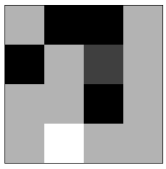
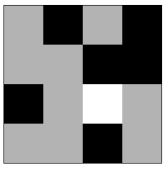
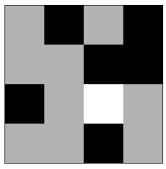
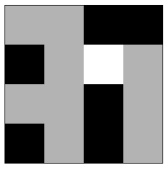
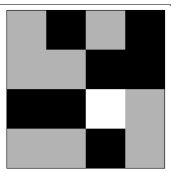
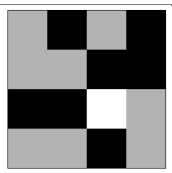
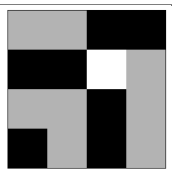
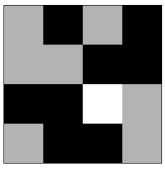
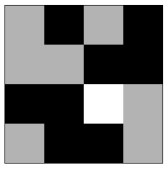
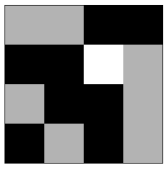
Set of Atoms	**37 atoms from mid-training of 4×4 example**		
L	$L = 4$	$L = 5$	$L = 6$
$\lambda = 0.1$			
$\lambda = 0.2$			
$\lambda = 0.3$			
$\lambda = 0.4$			
$\lambda = 0.5, \ldots, 1$			

Table 1: Minimal Hitting Sets outputted by STACCATO

However, when the complete freest model for the 3×3 and the 4×4 case are given, the results are much better and the algorithm can actually find examples where a clear vertical bar is present although sometimes with some additional noisy atoms. In these examples we knew before-hand the size of the desired examples so the value of L was chosen accordingly to that. It is also clear that as the value for λ increases different vertical bars are identified. This might be due to the fact that the constants are ranked but, in this case, all of them have the same ranking and thus some constants, although having the same relevance, are only considered when the range of the search (λ) is larger.

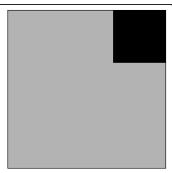
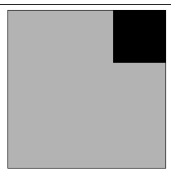
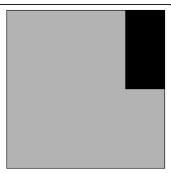
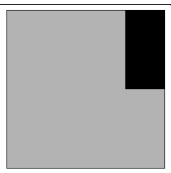
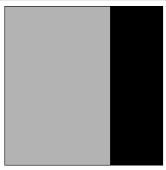
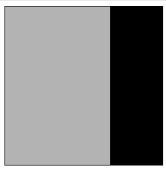
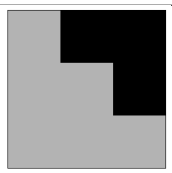
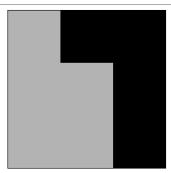
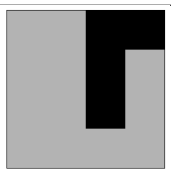
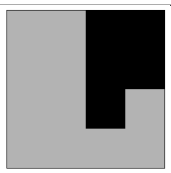
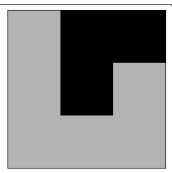
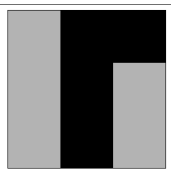
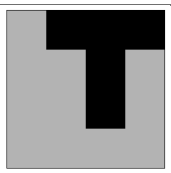
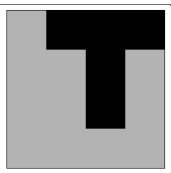
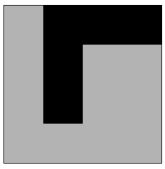
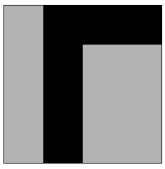
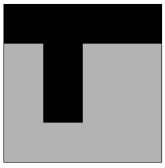
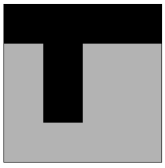
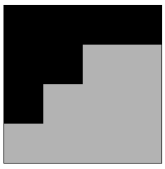
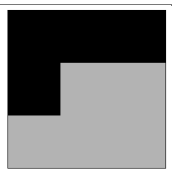
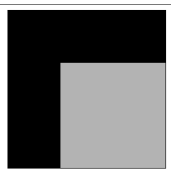
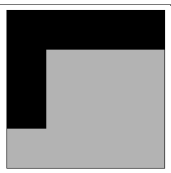
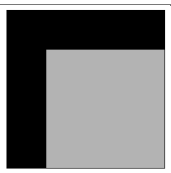
Set of Atoms	3×3 **complete model**		4×4 **complete model**	
L	$L = 2$	$L = 3$	$L = 3$	$L = 4$
$\lambda = 0.1$				
$\lambda = 0.2$				
$\lambda = 0.3$				
$\lambda = 0.4$				
$\lambda = 0.5$				
$\lambda = 0.6$				
$\lambda = 0.7$				
$\lambda = 0.8, \ldots, 1$				

Table 2: Minimal Hitting Sets outputted by STACCATO

Hitman

Lastly, it's relevant to also mention that for comparison purposes the Python library *PySAT*, and more specifically its module `Hitman`, was used to compute minimal hitting sets for these three cases [3]. The outputs using this module were, in fact, much better than the ones retrieved by STACCATO, given that, for the complete models it found the examples for all the exact black vertical bars and for the 37 atoms subset it found some small examples whose main characteristic

[3]The implementation of this approach is also available at this GitHub repository: https://github.com/MBernardoPereira/IntegratedProjectIST

was the third (counting from the left) black vertical bar. The inspection of the algorithms used in this module could be the target of further research on the development of more accurate and efficient algorithms to produce simple examples from a given model.

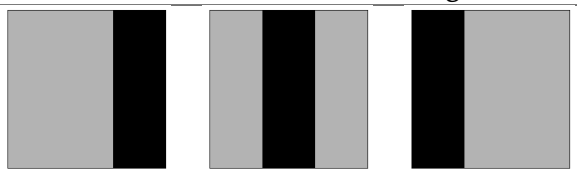
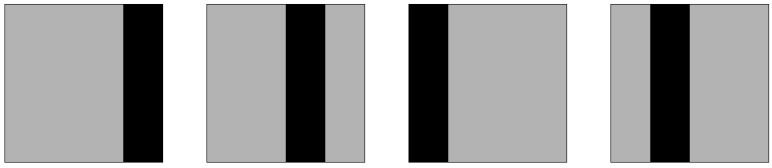
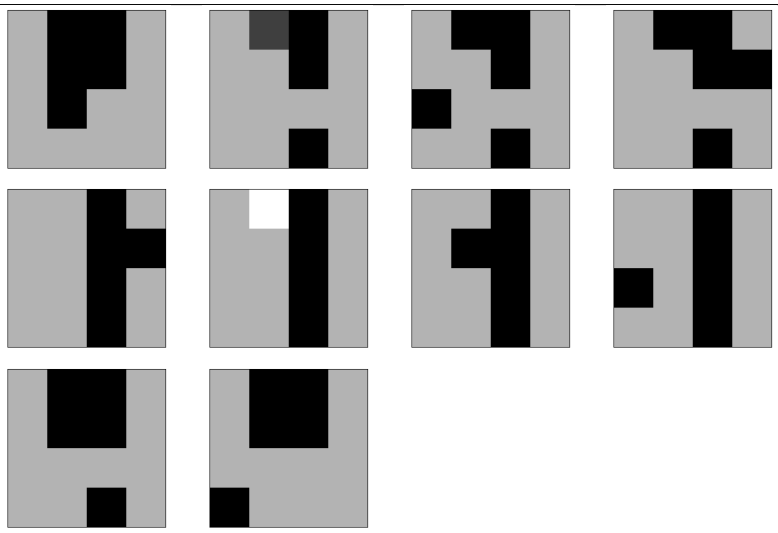
Set of Atoms	**Minimal Hitting Sets**
3×3 complete model	
4×4 complete model	
37 atom subset from 4×4 case	

Table 3: Minimal Hitting Sets outputted by Hitman

Discussion and Final Thoughts

The theme treated in this work has much more detail, depth and beauty than what I managed to concentrate here. The main problem addressed is somewhat parallel to the core mathematical development of the theory behind AML. However, it was very useful for me to get a first glimpse of what AML is and experiment some parallel challenges that arise in every scientific endeavour.

In the end, that main problem addressed here can be summarised as finding a minimal set of constants (logical literals) that hit (satisfy) the set of atoms in the final model (conjunction of disjunctions of literals). So, the problem arises in both a set-theoretic and logical format. This duality on the formulation of the problem also echoes in the solutions found. The main solution implemented, STACCATO, approaches the problem by dealing with a set-construction of a solution, whereas the best performance solution, the one in the Hitman package, approaches the problem with an algorithm designed to solve MaxSAT instances [10]. The Max(imum)SAT(ISFIABILITY) problem, in logic, addresses the satisfiability of CNF formulas by considering *hard* clauses and *soft* clauses. This distinction relaxes the usual SAT formulation so that not every clause needs to be satisfied. Sometimes this classification can be translated into a weighted approach where each clause is associated with a weight that determines its importance in the satisfiability problem. As stated in the last chapter, a deeper analysis of these kind of approach could lead to a better overall procedure for finding rules in models obtained with Algebraic Machine Learning.

Nevertheless, this work has served greatly to immerse myself in a completely new (and unexpected, from my narrow point of view) area of knowledge, and way of learning, just for the sake of curiosity and discovery, which was a radically new experience for me.

Last, but not least, I'd like to thank Professor Pedro A. Santos for always demonstrating his willingness to help me since the beginning of the semester and for the fruitful suggestions. Because this is a very recent field and with few contributors I had the opportunity to be guided by one of its co-authors, Dr. Gonzalo de Polavieja, whom I thank for the interesting discussions, for allowing me to have my first contact with scientific research, for teaching me the essentials of this area and for guiding me through some of its boundaries, always with such a kind attitude and predisposition. I hope it was worth it.

References

[1] Fernando Martin-Maroto and Gonzalo G. de Polavieja, Algebraic Machine Learning, https://arxiv.org/abs/1803.05252, 2018.

[2] Fernando Martin-Maroto and Gonzalo G. de Polavieja, Finite Atomized Semilattices, https://arxiv.org/abs/2102.08050, 2021.

[3] Andrew Gainer-Dewar and Paola Vera-Licona, The minimal hitting set generation problem: algorithms and computation, https://arxiv.org/abs/1601.02939, 2016.

[4] Rui Abreu and Arjan J.C. van Gemund, A Low-Cost Approximate Minimal Hitting Set Algorithm and its Application to Model-Based Diagnosis, 2009.

[5] Champalimaud Research Symposium 2021 - Gonzalo Polavieja, https://www.youtube.com/watch?v=afwFXKL-fyQ (Accessed in: 2 July 2022).

[6] David Marker, Model Theory : An Introduction, Graduate Texts in Mathematics, Springer, 2002, https://doi.org/10.1007/b98860.

[7] Rudolf Lidl and Gunter Pilz, Applied Abstract Algebra, Undergraduate texts in Mathematics, Springer, 1984, https://doi.org/10.1007/978-1-4615-6465-2.

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila and Francisco Herrera, Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI, https://arxiv.org/abs/1910.10045v1, 2019.

[9] Richard M. Karp, Reducibility among Combinatorial Problems, https://doi.org/10.1007/978-1-4684-2001-2_9, 1972.

[10] Alexey Ignatiev, Antonio Morgado and João Marques-Silva, RC2: an Efficient MaxSAT Solver, Journal on Satisfiability, Boolean Modeling, and Computation 11 pp.53-64, https://www.researchgate.net/publication/338092543_RC2_an_Efficient_MaxSAT_Solver/fulltext/5dfd7fba299bf10bc36bc58f/RC2-an-Efficient-MaxSAT-Solver.pdf?origin=publication_detail, 2019