# AML - Notes

June 2, 2021

## 1 The basics of Semilattices

**Definition:** A binary operation $R$ in $E$ is an **order** if:

- $R$ is reflexive: $xRx$
- $R$ is anti-symetric: for any $x \in E$, and $y \in E$, if $xRy$ and $y \neq x$, then $y \not R x$.
- $R$ is transitive: if $xRy$ and $yRz$ then $xRz$.

**Definition:** A binary operation $R$ in $E$ is a **total order** if it is an order and:

$\forall (x, y) \in E^2 / x \neq y$: $xRy$ or $yRx$.

**Definition:** A binary operation $R$ in $E$ is a **partial order** if it is an order but not a total one.

**Definition:** If an order $R$ is not reflexive, it is a **strict order** of $E$.

**Definition:** A semilattice is a structure $\boldsymbol{S} =< S, \cdot >$ where $\cdot$ is a binary operation, called the semilattice operation, such that:

- $\cdot$ is associative: $(xy)z = x(yz)$
- $\cdot$ is commutative: $xy = yx$
- $\cdot$ is idempotent: $xx = x$

**Definition:** Let $\boldsymbol{S}$ and $\boldsymbol{T}$ be semilattices. A **morphism** from $\boldsymbol{S}$ to $\boldsymbol{T}$ is a function $h : S \to T$ that is a homomorphism; ie:

$h(xy) = h(x)h(y)$

**Definition:** A **join-semilattice** is a structure $\boldsymbol{S} =< S, \vee >$, where $\vee$ is a binary operation, called the **join**, such that:

- $\leq$ is a partial order, where $x \leq y \Leftrightarrow x \vee y = y$

**Definition:** A **meet-semilattice** is a structure $\boldsymbol{S} =< S, \wedge >$, where $\wedge$ is a binary operation, called the **meet**, such that:

- $\leq$ is a partial order, where $x \leq y \Leftrightarrow x \wedge y = x$

## 2 The basics of AML

### 2.1 Elements of the Algebra

The first paper describing AML uses semilattices to embed problems into an algebra. The binary operation is called "**merge**" (symbol $\odot$) and the set $S$ has three types of of elements: **constants, terms and atoms**.

Terms should describe the objects of the problem. For exemple, terms can represent images as matrices. Here's a simple exemple of a 4-pixel image described by a term $T$:

$$T = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$$

Here $c_1, c_2, c_3$ and $c_4$ are constants, the primitive description elements of any embedding problem, and so the previous representation of $T$ in the algebra $S$ is:

$$T = c_1 \odot c_2 \odot c_3 \odot c_4$$

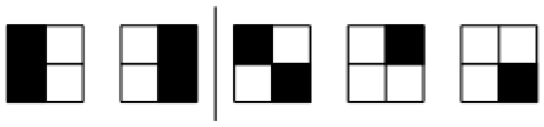
Atoms are elements created by the learning algorithm, represented by greek letters, and each constant is a merge of atoms. Here, $T$ is therefore a merge of atoms too.

Given that $T$ and $c_1$ are both elements of the algebra, it is clear that this algebra defines a partial order because the merge of $T$ and $c_1$ is still $T$ (ie. $T = T \odot c_1$). We can therefore establish the inclusion relationship $<$ between elements $a$ and $b$ of the algebra:

$$a < b \Leftrightarrow b = b \odot a \tag{1}$$

### 2.2 Training set

The toy exemple of the article is to classify 2x2 images in which pixels can be either black or white into images having a black vertical bar and images that don't. The training data is the following:



Where the two examples on the left are labeled "positive" because they have a black vertical bar (they are nammed $T_1^+$ and $T_2^+$) and the rest is labeled "negative" ($T_3^-$, $T_4^-$ and $T_5^-$) because they don't.

So the "training set" of the algebra consists of a set $R$ of positive and negative relations of the form $(v < T_1^+)$ or $\neg(v < T_3^-)$, where $v$ is a constant, one we want to describe to our algebra by using examples and counterexamples.

## 2.3 Graph of the algebra

The graph of the algebra is a representation of the inclusion relationships we just described. The graph has nodes only for the subset of terms in the trainning data, plus the "pinning terms" that are calculated by the algorithm (those will be introduced later).

This means that $a \to b \Rightarrow a < b$, but there is no if and only if, because not all terms should be in the graph.

If a term $T$ is *defined* as the merge of constants $c_i$ then:

$$T = \bigodot_i c_i \Rightarrow \forall i, c_i \to T$$

If all components of a term $T$ are also components constants of another term $S$, then there is an edge between the two:

$$T < S \Rightarrow T \to S$$

For atoms only, we have:

$$\phi \to b \Leftrightarrow \phi < b$$

For the graph of the algebra, the **partial order** $<$ is defined as:

$$a < b \Leftrightarrow \forall \phi, ((\phi \not\to a) \vee (\phi \to b)) \tag{2}$$

where the universal quantifier runs over all atoms.

This formula states that $a < b$ if and only if all the atoms edged to $a$ are also edged to $b$; given that for each atom, either the atom is not edged to $a$ but edged to $b$ or it is not edged to both $b$ and $a$.

**Proof of the partial order**

i. Reflexion.

Let $x$ be a node of the graph but not an atom.
$\forall \phi, (\phi \not\to x) \vee (\phi \to x) \equiv \neg(\phi \to x) \vee (\phi \to x) = \top$

ii. Antisymetry:

Let's proove that if $a < b$ and $a \neq b$ then $b \not< a$. Or, by the definition of the relation:

$$\text{for } a \neq b,\ a < b \Rightarrow (\exists \phi, \phi \not\to a \wedge \phi \to b)$$

This formula means that if $a$ is included in $b$, and if $a$ is not $b$, then there must be at least one atom that $b$ has but $a$ doesn't.

To proove it, we define $a$ and $b$ such that $a < b$ and $a \neq b$. Then, we suppose the negation of the right side of the implication to be true. Therefore we have, for all $\phi$:

$$\phi \not\to a \vee \phi \to b$$

and,

$$\phi \to a \vee \phi \not\to b$$

- $\phi \not\to a \wedge \phi \to a$ and $\phi \not\to b \wedge \phi \to b$ is trivialy contradictory.
- $\phi \not\to a \wedge \phi \not\to b$ is impossible because each constant must be a merge of atoms.
- $\phi \to a \wedge \phi \to b$ is impossible because it would mean that $a = b$.

iii. Transitivity:

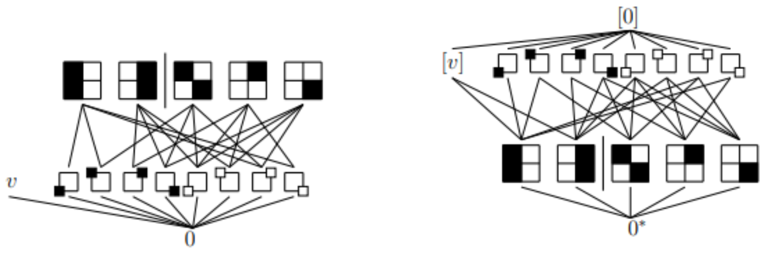
To prove the transitivity proprety we must proove that:

$$\forall \phi, (\phi \not\to a \vee \phi \to b) \wedge (\phi \not\to b \vee \phi \to c) \Rightarrow (\phi \not\to a \vee \phi \to c)$$

Or, it's contraposition:

$$\exists \phi, (\phi \to a \wedge \phi \not\to c) \Rightarrow (\phi \to a \wedge \phi \not\to b) \vee (\phi \to b \wedge \phi \not\to c)$$

Let's suppose an atom $\phi$ for witch $\phi \to a \wedge \phi \not\to c$ exists. This means that an atom that is edged to $a$ but not edged to $c$ exists. Let's consider two cases: one where $\phi$ is edged to $b$ and one where it's not. If $\phi$ is edged to $b$, then we have $\phi \to b \wedge \phi \not\to c$. Therefore the contraposition is verified because of it's right term. If $\phi$ is not edged to $b$, we still have $\phi \to a \wedge \phi \not\to b$. The contraposition is verified because of it's left term and the transitivity is therefore proved.



(a) graph of M (b) graph of M*

Figure 1: The master algebra and the dual algebra

When the graph is transitively closed it describes an algebra called $M$. This algebra evolves during the learning process producing a model of the training relations $R$ at the end of the embedding. Therefore, in the future, $M$ means "the algebra described by the graph at a given stage of the algorithm".

## 2.4 The dual algebra

The algebraic manipulations are easier to perform using not only the algebra $M$ but also an auxiliary structure $M^*$. This $M^*$ is a semilattice closely related (but different) to the dual of $M$, simply called "the dual".

To "build" the dual, let us define an algebraic structre $S$ that contains both semilattices $M$ and $M^*$, as well as an unary function [ ] defined for S, that maps elements of M into elements of $M^*$ ($[\ ] : M \to M^*$).

The duals of constants and terms are always constants and **the dual of atoms are a new kind of element we name "dual-of-atom"**. $M^*$ has constants, dual-of-atoms and atoms but **it does not contain terms**. Atoms of $M^*$ are not duals of any element of $M$. We refer to $M^*$ as the dual algebra and to $M$ as "the master" algebra.

The algebra $S$ is now characterized by the transitive, noncommutative relation $\to$, the partial order $<$ and the unary operator [ ]. Besides the transitiviry of $\to$ and the definition of $<$ given by (2), an additional axiom is introduced:

$$a \to b \Rightarrow [b] \to [a] \tag{3}$$

This is what makes $M^*$ closely related to the "mathematical" dual of $M$ but as it was said, $M^*$ is not quite the dual of $M$. **The important differences is that $M^*$ has addition edges for the positive order relations of R** (the set of positive and negative relations of the form $(v < T_1^+)$ or $\neg(v < T_3^-)$):

$$[T_1^+] < [v] \tag{4}$$

In figure (1), we can see that $v$ is edged to the two positive examples of our training set. Atoms of $M$ (0) are now the duals-of-atoms of M ([0]) and $M^*$'s own atoms (0*) are also represented.

## 2.5 Atomized models

### 2.5.1 Defining the $\boldsymbol{GL^a}$ set

To help formalization, $\boldsymbol{C}(S)$ and $\boldsymbol{A}(S)$ respectively refers to the subset of constants and atoms of a set $S$. Note that $\boldsymbol{C}(M)$ and $\boldsymbol{A}(M)$ can also apply to a model $M$.

The lower segement $\boldsymbol{L}$ of an element $x$ is $\boldsymbol{L}(x) = \{y | y \leq x\}$.

To distinguish an algebra from its graph we use the prefix $\boldsymbol{G}$, so the lower segement for the graph is defined as $\boldsymbol{GL}(x) = \{y | (y \to x) \vee (y = x)\}$

1

$\boldsymbol{GL}(x)$ therefore contains all the elements edged to $x$ from the "lower" side of the graph.

Finaly a superscrit $a$ is used to denote the intersection with the atoms; and we get the final $\boldsymbol{GL^a}$ set we will use.

Formaly, for $x \in M$:

$$\boldsymbol{GL^a}(x) = \boldsymbol{GL}(x) \cap \boldsymbol{A}(M) \tag{5}$$

Therefore, $\boldsymbol{GL^a}(x)$ is simply the intersection between the elements that are lower than $x$ and all the atoms of a model $M$: **so we end up geting the atoms that are edged to** $x$.

One important thing to point out, is the case of atoms **in M** ($x \in \boldsymbol{A}(M)$) because we then have $\boldsymbol{GL}(x) = \emptyset$ (by definition of the atoms) and so $\boldsymbol{GL^a}$ becomes the set of all atoms of $M$:

$$\forall \phi \in \boldsymbol{A}(M), \boldsymbol{GL^a}(\phi) = \boldsymbol{A}(M) \tag{6}$$

### 2.5.2 Formal definition of the merge operation in an atomized model

Equation (2) defines how to derive the partial order from the transitive, noncommutative edge relation $\to$ and atoms. **A model for which there is a description of the partial order in terms of a set of atoms is an <u>atomized</u> model**.

Even if we know how to derive the partial order from the atoms and edges, we have not yet a definition for the idempotent operator.

In order to have a more formal definition of the idempotent operator, we can use the set $\boldsymbol{GL^a}(x)$ that is defined, as always, only when the graph is transitively closed.

Definition: The merge of $a$ and $b$ is the element $a \odot b$ which verifies:

$$\boldsymbol{GL^a}(a \odot b) = \boldsymbol{GL^a}(a) \cup \boldsymbol{GL^a}(b) \tag{7}$$

$\odot$ is therefore the binary operation defined by:

$$a \odot b = c \Leftrightarrow \boldsymbol{GL^a}(c) = \boldsymbol{GL^a}(a) \cup \boldsymbol{GL^a}(b)$$

Because the idempotent operator becomes a trivial set union of atoms it is clear that this operation is idempotent, commutative and associative. It is also consistent with the partial order given in equation (2):

$$(a \odot b = b) \Leftrightarrow (a < b) \Leftrightarrow (\forall \phi, ((\phi \nrightarrow a) \vee (\phi \rightarrow b)))$$

From these equivalences the relationship between the partial order and $\boldsymbol{GL^a}$ is clear that $a < b \Leftrightarrow \boldsymbol{GL^a}(a) \cup \boldsymbol{GL^a}(b) = \boldsymbol{GL^a}(b) \Leftrightarrow \boldsymbol{GL^a}(a) \subset \boldsymbol{GL^a}(b)$; so:

**Proprety 1 of $GL^a$**

$$a < b \Leftrightarrow \boldsymbol{GL^a}(a) \subset \boldsymbol{GL^a}(b) \tag{8}$$

[**Note** The use of a strict $\subset$ is not consistant with the idempotent proprety $x \odot x = x$ because $x \odot x = x \Leftrightarrow x < x \Leftrightarrow \boldsymbol{GL^a}(x) \subset \boldsymbol{GL^a}(x)$ is technically false when it should be true, given that $x < x \Leftrightarrow \forall \phi, (\neg(\phi \rightarrow x) \vee (\phi \rightarrow x)) = \top$. This might just be a choice of notations but in that case $\subset$ should be understood as $\subseteq$ ]

For our images problem, we want a model for which $v$ is a set included in the positive training images, $T_1^+$ and $T_2^+$ as

$$v < T_i^+ \Leftrightarrow \boldsymbol{GL^a}(v) \subset \boldsymbol{GL^a}(T_i^+)$$

We will also be looking for a particular atomic model for which the atoms of constant $v$ are not all in the terms corresponding with negative training examples:

$$v \nless T_i^- \Leftrightarrow \boldsymbol{GL^a}(v) \not\subset \boldsymbol{GL^a}(T_i^-)$$

## 2.6 Trace and trace constrains

### 2.6.1 Definitions and propreties

The trace is central for the embedding procedure as a guiding tool for algebraic transformations. By operating the algebra while keeping the trace of some elements invariant, we can control the global effects caused by our local changes.

The trace $\mathbf{Tr}(x)$ maps an element $x \in M$ to a set of atoms in $M^*$. To calculate the trace of $x$, we find its atoms in the graph of $M$ ($\boldsymbol{GL^a}(x)$); since atoms are minima of $M$, dual-of-atoms are maxima of $M^*$, so for each atom $\phi_i$ of $x$ there is a dual-of-atom at the top of the graph of $M^*$, $[\phi_i]$. Each of these $[\phi_i]$ also have an associated set of atoms in $M^*$, $\boldsymbol{GL^a}([\phi_i])$. The trace of $x$ is defined as the intersection all the atoms of $[\phi_i]$ in $M^*$. Formaly:

For $x \in M$, the trace of $x$ is defined as:

$$\mathbf{Tr}(x) = \bigcap_{\phi \in \boldsymbol{GL^a}(x)} \boldsymbol{GL^a}([\phi]) \tag{9}$$

Here are some propreties of the trace:

**Proprety 1 of the trace**

$$\mathbf{Tr}(a \odot b) = \mathbf{Tr}(a) \cap \mathbf{Tr}(b) \tag{10}$$

<u>Proof:</u>

$$\begin{aligned}
\mathbf{Tr}(a \odot b) &= \bigcap_{\phi \in \boldsymbol{GL^a}(a \odot b)} \boldsymbol{GL^a}([\phi]) \\
&= \bigcap_{\phi \in \boldsymbol{GL^a}(a) \cup \boldsymbol{GL^a}(b)} \boldsymbol{GL^a}([\phi]) \\
&= \bigcap_{\phi \in \boldsymbol{GL^a}(a)} \boldsymbol{GL^a}([\phi]) \ \cap \bigcap_{\psi \in \boldsymbol{GL^a}(b)} \boldsymbol{GL^a}([\psi]) \\
&= \mathbf{Tr}(a) \cap \mathbf{Tr}(b)
\end{aligned}$$

**Proprety 2 of the trace**

$$a < b \Rightarrow \mathbf{Tr}(b) \subset \mathbf{Tr}(a) \tag{11}$$

<u>Proof:</u>

From (2) we have:

$$a < b \Leftrightarrow a \odot b = b$$

And from the proprety (1) of the trace:

$$a \odot b = b \Rightarrow \mathbf{Tr}(a \odot b) = \mathbf{Tr}(a) \cap \mathbf{Tr}(b) = \mathbf{Tr}(b) \Rightarrow \mathbf{Tr}(b) \subset \mathbf{Tr}(a)$$

This makes a correspondence between order relations in $M$ and trace interrelations between $M$ and $M^*$ that are called **trace constrains**. For our image problem, we are intrested in obeying trace constrains for the positive training examples, $v < T_i^+$, for which we then need to enforce $\mathbf{Tr}(T_i^+) \subset \mathbf{Tr}(v)$.

For negative training examples, $T_i^-$, we want $\neg(v < T_i^-)$ so we want to enforce $\mathbf{Tr}(T_i^-) \not\subset \mathbf{Tr}(v)$.

**There is actualy no inplicationryuiop between $\mathbf{Tr}(T_i^+) \subset \mathbf{Tr}(v)$ and $v < T_i^+$** (given that there is no equivalence) **but** this still provides a necessary starting point, giving that once the trace constraint is met, no transformation of $M$ can produce $v < T_i^-$:

$$\mathbf{Tr}(T_i^-) \not\subset \mathbf{Tr}(v) \Rightarrow \neg(v < T_i^-) \tag{12}$$

[**Note**: Authors say that this inclusion does not follow from (9), yet it seems to be the contraposition of (11), which follows from (1) and (10), wich does follow from (9):]

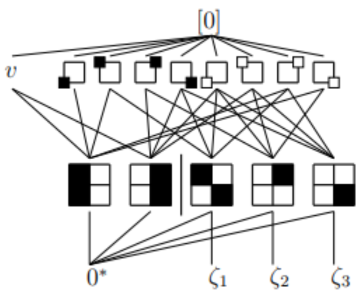
Page 11-12: *While the operator [ ] does not really map M into its dual semilattice, the traces of the elements of M form an algebra that very much resembles the dual of M. This new algebra has trace constraints in the place of order relations and set intersections in the place of set unions. There are still some subtle differences between a proper dual of M and the dual algebra provided by the trace. For example, the trace is defined with the atoms of M instead of the constants of M, so it depends on the particular atomization of M. While finding a proper dual of M amounts in difficulty to calculate M itself, enforcing the trace constraints is easier because we have the extra freedom of introducing new atoms in M. In addition, we do not have restrictions for the size of the traces. We do not care if the traces are large or small. We want an atomization for M but first we have to calculate an atomization for M*. The atomization we are going to build for M does not correspond with the dual of M* , neither it corresponds with the dual of the algebra defined by the trace. It corresponds with an algebra freer than the algebra described by the traces. In Section 3.4 we explain the role that algebraic freedom plays as a counterbalance to cardinal minimization.*

### 2.6.2 Enforcing trace constrains

**A first optional step**

To enforce trace constrains we can start, although this step is optinal, by enforcing $\neg([T_i^-] < [v])$ by adding an atom to $[T_i^-]$ in $M^*$ so we enforce $\neg(v < T_i^-)$ in M.

In our image exemple, for every negative example $T_i^-$ we add atoms $\zeta_i$, $i \in [|1,3|]$, so that $\forall i \in [|1,3|], \zeta_i \to [T_i^-]$:



The new atoms are not in $\boldsymbol{GL^a}([v])$ so the reverted negative relations are satisfied:

$$\neg([T_i^-] < [v]) \Leftrightarrow \boldsymbol{GL^a}([T_i^-]) \not\subset \boldsymbol{GL^a}([v]) = \{0^*, \zeta_i\} \not\subset \{0^*\}$$

In fact all reverted relations, positive and negative, are satisfied at this point (remenber that for the authors $\subset$ means $\subseteq$).

$$[v] < [T_i^+] \Leftrightarrow \boldsymbol{GL^a}([v]) \subset \boldsymbol{GL^a}([T_i^+]) = \{0^*\} \subset \{0^*\}$$

## Preprocessing

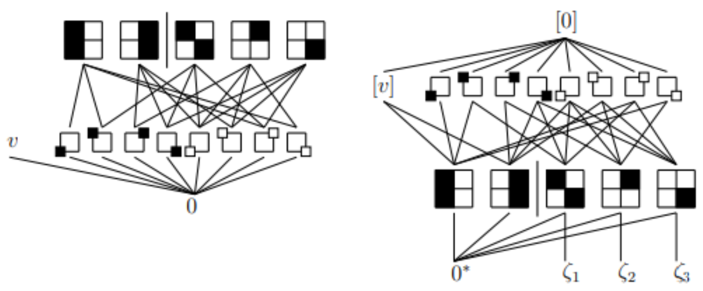
We have now the chance to detect if the input order relations are inconsistent. First, we make sure that for each couple of terms $T_1$ and $T_2$ mentionned in the input order relations such that the component constants of $T_1$ are a subset of those of $T_2$ we have added the edge $[T_2] \rightarrow [T_1]$.

If there are edges pointing in both directions ? between two elements of $M^*$ we can identify them as the same element. **Two or more elements of $M$ may share the same dual.**

This completes the preprocessing step that speeds up the enforcing of the trace constrains and validates the consistency of the embedding.

## Enforcing the trace constrains

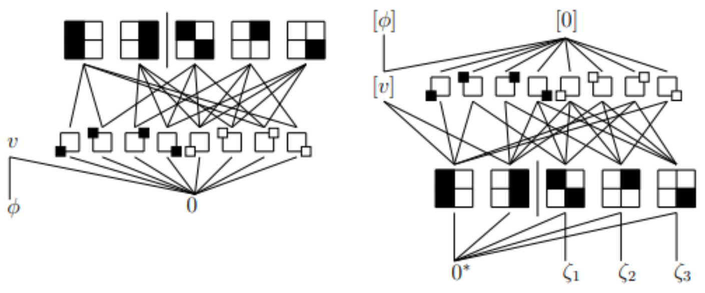
We can now enforce trace constrains for the negative examples, $\mathbf{Tr}(T_i^-) \not\subset \mathbf{Tr}(v)$. To compute the trace, we place the graph for $M$ and for $M^*$ side to side, to left and right respectively.



We start with the negative trace constrains. The trace for the negative training examples is $\mathbf{Tr}(T_i^-) = \boldsymbol{GL^a}([0]) = \{0^*, \zeta_1, \zeta_2, \zeta_3\}$. For constant $v$ we also have $\mathbf{Tr}(v) = \boldsymbol{GL^a}([0])$, so it does not obeyed $\mathbf{Tr}(T_i^-) \not\subset \mathbf{Tr}(v)$; **we need to enforce it**.

For this, we need to choose a constant $c \in M$ equal to $v$ or such that $[c]$ recieves edges from $[v]$ and not from $[T_i^-]$. We then need to **add an atom** $\phi \rightarrow c$.

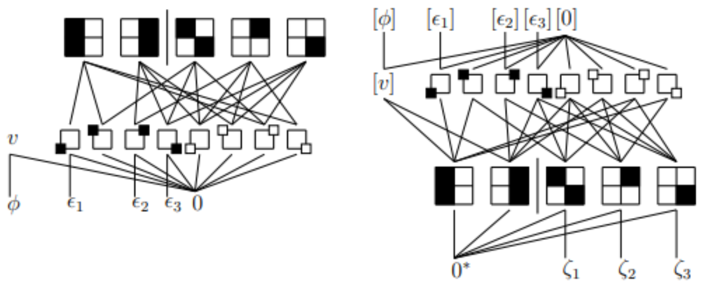
Note that this constant $c$ has propreties directly fulfuled by $v$ (the case where $c \in M$ is equal to $v$) so we can add $\phi \rightarrow v$ and the corresponding dual-of-atom $[v] \rightarrow [\phi]$ in $M^*$.



Now we have $\mathbf{Tr}(T_i^-) = \boldsymbol{GL^a}([0]) = \{0^*, \zeta_1, \zeta_2, \zeta_3\}$ but $\mathbf{Tr}(v) = \boldsymbol{GL^a}([0]) \cap \boldsymbol{GL^a}([\phi]) = \{0^*\}$, because $\boldsymbol{GL^a}([\phi]) = \{0^*\}$ so $\mathbf{Tr}(T_i^-) = \{0^*, \zeta_1, \zeta_2, \zeta_3\} \not\subset \mathbf{Tr}(v) = \{0^*\}$ is verified.

Now, for positive trace constrains, we want $v < T_i^+$ so we need to enforce $\mathbf{Tr}(T_i^+) \subset \mathbf{Tr}(v)$. First we check if this condition is satisfied: $\mathbf{Tr}(T_i^+) = \boldsymbol{GL^a}([0]) = \{0^*, \zeta_1, \zeta_2, \zeta_3\}$ and $\mathbf{Tr}(v) = \boldsymbol{GL^a}([0]) \cap \boldsymbol{GL^a}([\phi]) = \{0^*\}$. The condition is not satisfied.

Therefore, we add atoms $\epsilon_i$ to the constants $c_i$ until $\mathbf{Tr}(T_i^+)$ equals $\mathbf{Tr}(v)$. For the first term $T_1^+$ we just need to edge an atom to the first constant, $\epsilon_1 \to c_1$; for the secound training example, $T_2^+$ we need to add one atom for each of its constants, $\epsilon_2 \to c_3$ and $\epsilon_3 \to c_4$. After imposing the trace constraints the graphs $M$ and $M^*$ look like this:



Now we have:

$$\mathbf{Tr}(v) = \{0^*, \zeta_1, \zeta_2, \zeta_3\} \cap \{0^*\} = \{0^*\}$$
$$\mathbf{Tr}(T_1^+) = \boldsymbol{GL^a}([0]) \cap \boldsymbol{GL^a}([\epsilon_1]) = \{0^*, \zeta_1, \zeta_2, \zeta_3\} \cap \{0^*\} = \{0^*\}$$
$$\mathbf{Tr}(T_2^+) = \boldsymbol{GL^a}([0]) \cap \boldsymbol{GL^a}([\epsilon_2]) \cap \boldsymbol{GL^a}([\epsilon_3]) = \{0^*, \zeta_1, \zeta_2, \zeta_3\} \cap \{0^*, \zeta_2\} \cap \{0^*, \zeta_3\} = \{0^*\}$$

Giving us the desired result: $\mathbf{Tr}(T_i^+) \subset \mathbf{Tr}(v)$.

In practice , enforcing trace constrains is achieved through algorithms (two algorithms; one for positive trace constrains and negative trace constrains) that we will present later.

## 2.7 Full and Sparse Crossing operations

After enforcing the trace constraints, all negative relations $v \not< T_i^-$ are already satisfied in $M$. This will always be the case as long as the trace is unchanged (equation (12)). To build an atomized model that also satisfies the positive relations $v < T_i^+$, we use the Sparse Crossing operation.

This trace-invariant operation replaces the atoms of $v$ for others that are also in $T_i^+$ without interfering with previously enforced positive or negative relations and without changing the traces of any element of $M$.

### 2.7.1 Definition of the crossing operation

Let's consider two elements $a$ and $b$ with atoms:

$$\boldsymbol{GL^a}(a) = \{\alpha, \beta, \chi\}$$
$$\boldsymbol{GL^a}(b) = \{\chi, \delta, \epsilon\}$$

and suppose we want to enforce $a < b$. **Extend the graph appending new atoms and edges as**

$$\phi, \varphi, \gamma \to \alpha$$
$$\pi, \omega, \theta \to \beta$$
$$\chi', \phi, \pi \to \chi$$
$$\delta', \varphi, \omega \to \delta$$
$$\epsilon', \gamma, \theta \to \epsilon$$

Now close the graph by transitive closure and delete $\alpha, \beta, \chi, \delta$ and $\epsilon$ from the graph. Then it holds that:

$$\boldsymbol{GL^a}(a) = \{\phi, \varphi, \gamma, \pi, \omega, \theta\} \subset \boldsymbol{GL^a}(b) = \{\chi', \delta', \epsilon', \phi, \varphi, \gamma, \pi, \omega, \theta\}$$

and therefore $a < b$. This is the **Full Crossing of** $a$ **into** $b$ and it can represented with the table:

|  | $\chi$ | $\delta$ | $\epsilon$ |
|---|---|---|---|
|  | $\chi'$ | $\delta'$ | $\epsilon'$ |
| $\alpha$ | $\phi$ | $\varphi$ | $\gamma$ |
| $\beta$ | $\pi$ | $\omega$ | $\theta$ |

We say that we have "crossed" atoms $\alpha$ and $\beta$ into $b$. Note that the first square in the first line of the table is blank because there is no need to "cross" atom $\chi$ of $a$ as it is already in $b$.

Crossing made so that $b$ now shares all atoms that are edged to $a$, plus the initial atoms (though they are not quite the same) in $\boldsymbol{GL^a}(b)$.

In practice, since the crossing is an expensive operation that multiplies the number of atoms, we instead do a Sparse Crossing. The idea is that, as long as we check that all involved atoms remain trace-invariant, the Sparse Crossing operation still enforces $a < b$ and preserves all positive relations. In addition, it also preserves negative relations as long as they are "protected" by its corresponding negative trace constraint. Details of how sparse crossing works will be given later.

### 2.7.2 Full crossing operation propreties

Formaly, the full crossing operation transforms one graph into another and after transitive closure, it also maps one algebra into another. This mapping function commutes with both operators $\odot$ and $[\,]$ so it is a homomorphism, which gives us one proprety of the full crossing operation:

**Proprety 1 of full crossing**

The full crossing operator is a homomorphism of the graph algebra, ie. if we call $f$ the mapping function of full crossing:

$$\forall a, b \in M/\boldsymbol{A}(M), f(a \odot b) = f(a) \odot f(b) \tag{13}$$

We cannot take atoms into consideration, given that full crossing removes some atoms out of the graph.

**This formula means that if $a \odot b = c$ is true in $M$ then it will also be true after full crossing.**

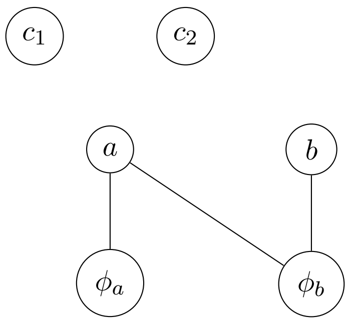
Given that $<$ is defined from $\odot$, we immediately get a corollary of this proposition:

$$\forall a, b \in M/\boldsymbol{A}(M), f(a < b) = f(a) < f(b) \tag{14}$$

Proof:

Let $T, c_1$ and $c_2$ be elements of $M/\boldsymbol{A}(M)$ such that $T = c_1 \odot c_2$. We want to prove that the full crossing operation of element $a$ into $b$ does not change the fact that $T = c_1 \odot c_2$. Because of the definition of $\odot$ (equation (7)), we know that:

$$T = c_1 \odot c_2 \Leftrightarrow \boldsymbol{GL^a}(T) = \boldsymbol{GL^a}(c_1) \cup \boldsymbol{GL^a}(c_2)$$



**Proprety 2 of full crossing – Theorem 1**

Full crossing of $a$ into $b$ leaves the trace of all atoms unchanged if and only if the positive trace constraint for $a < b$, ie. $\mathbf{Tr}(b) \subset \mathbf{Tr}(a)$ is satisfied.

Formaly, if $f$ is the mapping function of full crossing of $a$ into $b$:

$$\forall \phi \in \boldsymbol{A}(M), \mathbf{Tr}(f(\phi)) = \mathbf{Tr}(\phi) \Leftrightarrow \mathbf{Tr}(b) \subset \mathbf{Tr}(a) \tag{15}$$

Proof:

Let $a$ and $b$ be to elements of the graph. Let $\Phi$ be the set of atoms involved in the full crossing of $a$ **in** $b$. $\Phi$ is entirely defined as:

$$\Phi = \boldsymbol{GL^a}(a) \Delta \boldsymbol{GL^a}(b)$$

Where $\Delta$ is the symmetric difference of sets.

For each $\phi \in \Phi$; crossing of $a$ in $b$ will result in the creation of a family of atoms $(\phi_j)_{j \in N}$ such that $\phi = \bigodot_j \phi_j$.

Before crossing $\phi$ is a minima and by definition $\mathbf{Tr}(\phi) = \boldsymbol{GL^a}([\phi])$. After the crossing the new minima are $(\phi_j)$ and so $\mathbf{Tr}(\bigodot_j \phi_j) = \bigcap_j \boldsymbol{GL^a}([\phi_j])$. The left side of the equivalence therefore becomes:

$$\forall \phi \in \Phi, \boldsymbol{GL^a}([\phi]) = \bigcap_j \boldsymbol{GL^a}([\phi_j])$$

Since $\phi$ is not in both $a$ and $b$, we have either $(\phi < a) \wedge \neg(\phi < b)$ or $\neg(\phi < a) \wedge (\phi < b)$; meaning we have either $\phi$ edged to $a$ or $\phi$ edged to $b$.

if $\phi$ is edged to $b$:

**An atom $\phi$ initialy edged to $b$ always has a corresponding element $\phi'$ edged only to $\phi$**; meaning that the family that is edged to $\phi$ contains one atom $\phi'$ that verifies: $\boldsymbol{GL^a}([\phi']) = \boldsymbol{GL^a}([\phi])$. In addition, because atoms $\phi_j$ can only be edged to $a$ and $b$ (or stricly $b$); we always have more (or stricly the same) atoms in $\boldsymbol{GL^a}([\phi_j])$ than we have in $\boldsymbol{GL^a}([\phi'])$. So:

$$\forall \phi_j, \boldsymbol{GL^a}([\phi']) \subset \boldsymbol{GL^a}([\phi_j]) \Rightarrow \bigcap_j \boldsymbol{GL^a}([\phi_j]) = \boldsymbol{GL^a}([\phi']) = \boldsymbol{GL^a}([\phi])$$

From the previous equation we get the desired result **with or without $\mathbf{Tr}(b) \subset \mathbf{Tr}(a)$**:

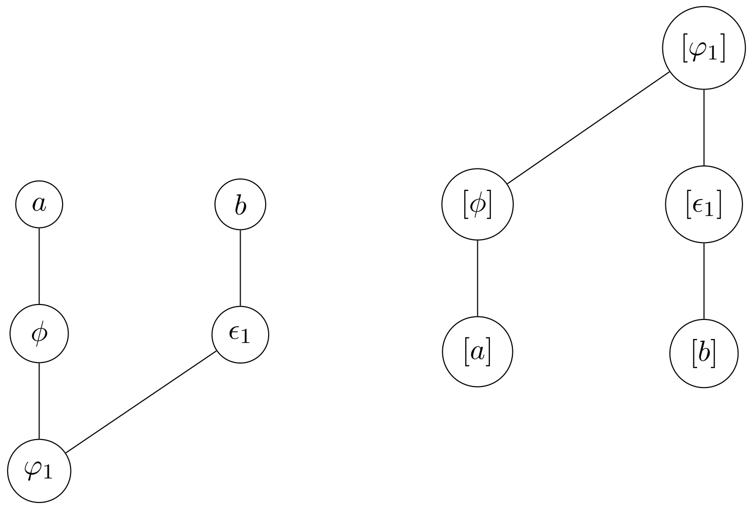
$$\forall \phi \in \boldsymbol{GL^a}(b), \boldsymbol{GL^a}([\phi]) = \bigcap_j \boldsymbol{GL^a}([\phi_j]) \text{ is true for any crossing}$$

if $\phi$ is edged to $a$:

Let's say $b$ was atomized as $b = \bigodot_j \epsilon_j$, before crossing. For each $\epsilon_j$, a new atom $\varphi_j$ is created and edges $(\varphi_j \to \phi \wedge \varphi_j \to \epsilon_j)$ are appended to the graph. New edges are appended to the graph of $M^*$: $([\phi] \to [\varphi_j] \wedge [\epsilon_j \to [\varphi_j])$. Since no other elements are edged to $[\varphi_j]$, then:

$$\boldsymbol{GL^a}([\varphi_j]) = \boldsymbol{GL^a}([\phi]) \cup \boldsymbol{GL^a}([\epsilon_j])$$

For $j = 1$, we can visualize the situation for a better understanding:



The previous equation then gives us:

$$\begin{aligned}\mathbf{Tr}(\bigodot_j \varphi_j) &= \bigcap_j \boldsymbol{GL^a}([\varphi_j]) \\ &= \bigcap_j \{\boldsymbol{GL^a}([\phi]) \cup \boldsymbol{GL^a}([\epsilon_j])\} \\ &= \boldsymbol{GL^a}([\phi]) \cup \{\bigcap_j \boldsymbol{GL^a}([\epsilon_j])\} \\ &= \boldsymbol{GL^a}([\phi]) \cup \mathbf{Tr}(b)\end{aligned}$$

This means that (given that before crossing, $\phi$ is a minima and by definition $\mathbf{Tr}(\phi) = \boldsymbol{GL^a}([\phi])$), $\forall\phi \in \boldsymbol{GL^a}(a)$:

$$\mathbf{Tr}(f(\phi)) = \mathbf{Tr}(\phi) \cup \mathbf{Tr}(b)$$

**In other words: when $\phi$ is crossed into $b$, the trace of $\phi$ gains the set $\mathbf{Tr}(b)$. Meaning that $\mathbf{Tr}(\phi)$ remains invariant if and only if it contained $\mathbf{Tr}(b)$ before crossing.** For a formal proof of the equivalence:

Proof of que equivalence:

Let's suppose that $\forall\phi \in \boldsymbol{GL^a}(a), \mathbf{Tr}(f(\phi)) = \mathbf{Tr}(\phi)$; then the previous equation becomes:

$$\forall\phi \in \boldsymbol{GL^a}(a), \mathbf{Tr}(\phi) = \mathbf{Tr}(\phi) \cup \mathbf{Tr}(b) \Rightarrow \mathbf{Tr}(b) \subset \mathbf{Tr}(\phi)$$

By definition:

$$\mathbf{Tr}(a) = \bigcap_{\phi\in\boldsymbol{GL^a}(a)} \mathbf{Tr}(\phi)$$

And so we get the final result we were looking for:

$$\forall\phi \in \boldsymbol{GL^a}(a), \mathbf{Tr}(f(\phi)) = \mathbf{Tr}(\phi) \Rightarrow \mathbf{Tr}(b) \subset \mathbf{Tr}(a)$$

Let's now suppose that $\mathbf{Tr}(b) \subset \mathbf{Tr}(a)$; then:

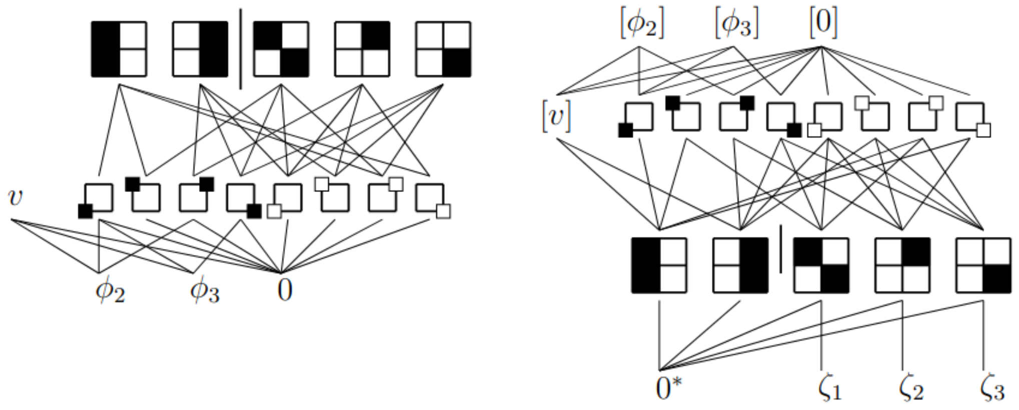
$$\mathbf{Tr}(b) \subset \bigcap_{\phi \in \boldsymbol{GL^a}(a)} \mathbf{Tr}(\phi) \Rightarrow \forall \phi \in \boldsymbol{GL^a}(a), \mathbf{Tr}(b) \subset \mathbf{Tr}(\phi).$$

So:

$$\forall \phi \in \boldsymbol{GL^a}(a), \mathbf{Tr}(f(\phi)) = \mathbf{Tr}(\phi) \cup \mathbf{Tr}(b) = \mathbf{Tr}(\phi)$$

### 2.7.3 After Sparse Crossing

After the sparse crossing algorithm is applied in our image example, we get the following state of the algebras:



The Sparse Crossing operation has worked; the positive training examples all obey $v < T_i^+$ and the negatives ones $v < T_j^-$. The atoms of $v$ are $\boldsymbol{GL^a}(v) = \{0, \phi_2, \phi_3\}$ and the atoms for the positive training examples are also $\boldsymbol{GL^a}(T_{1,2}^+) = \{0, \phi_2, \phi_3\}$, while for the negative examples we have $\boldsymbol{GL^a}(T_{1,3}^-) = \{0, \phi_3\}$ and $\boldsymbol{GL^a}(T_2^-) = \{0, \phi_2\}$.

## 2.8 Reduction operation

TODO

The trace-preserving reduction scheme presented in this section works well in combination-with Sparse Crossing and finds small, generalizing models efficiently.

## 2.9 Batch training

So we have seen how to learn an atomized model from a set $R$ of positive and negative examples. In practice, we would check the accuracy of the learned model in test data. If the accuracy is below some desired level, we would continue training with a new set of ewamples. Therefore, rather than having a single set $R$, we have a series of **batches** $R_0, R_1, ..., R_n$ where the subscript corresponds with the training epoch.