

On the relationship between distance and generalisation

V. Estruch C. Ferri J. Hernández-Orallo M.J. Ramírez-Quintana

Abstract

Distance-based methods are a successful family of machine learning techniques since the inception of the discipline. Basically, the classification or clustering of a new individual is determined by the distance to one or more prototypes. From a comprehensibility point of view, this is not especially problematic in propositional learning where prototypes can be regarded as a good generalisation (pattern) of a group of elements. However, for scenarios with structured data this is no longer the case. In recent work, we have developed a framework to determine whether a pattern computed by a generalisation operator is consistent w.r.t a distance. In this way, we can determine which patterns can provide a good representation of a group of individuals belonging to a metric space. In this work we apply this framework to analyse and define minimal distance-based generalisation operators (*mg* operator) for first-order data. We show that Plotkin’s *lgg* is a *mg* operator for atoms under the distance introduced by [13]. We also show that this is not the case for clauses with the distance introduced in [12]. Consequently, we introduce a new *mg* operator for clauses, which could be used as a base to adapt existing bottom-up methods in ILP.

1 Introduction

Learning from complex data is one of the main challenges in machine learning (e.g. distance-based and kernel-based methods for structured data [8]). But learning from complex data while preserving comprehensibility is even more challenging, and has mainly been addressed in the area of ILP [9]. For instance, despite the fact that distance-based methods are quite intuitive and have successfully been tested in several domains, a model which explains why a new example belongs to one class or another is missing.

Roughly speaking, this is due to the fact that the information about the matches between two objects (e.g. two molecules) is lost when are encoded by a number (their distance). Unfortunately, this lack of explanatory patterns is incompatible with many application contexts. Imagine in molecule classification, how interesting it would be to describe a cluster of molecules by saying what chemical structures these molecules have in common instead of saying that they are close to one given prototype. We addressed the possibility of this kind of descriptions for distance-based algorithms in [2], where the concept of distance-based binary generalisation operators is introduced. Basically, the term ‘distance-based’ means that the operator computes patterns which are “consistent” with the distance employed. For instance, let (Σ^*, d) be the space of words defined over the alphabet $\Sigma = \{a, b, c\}$ and d the edit distance. Given the words $w_1 = cabab$ and $w_2 = ababc$ a distance-based generalisation operator could compute $*abab*$. That is, all the words having the subsequence $abab$.

This pattern somehow shows why $d(w_1, w_2) = 2$ because the subsequence $abab$ has been taken into account in the best match to obtain the distance. However, this is not the case for another operator computing $*c*$ (all the words having the symbol c) since the common sequence c is not considered to compute the distance.

Unfortunately, if we want to use these generalisation operators in a real context, we need to be able to generalise more than two elements. In this work, we introduce this idea for n -ary operators and we also study the idea of minimality. Minimality is important to avoid underfitting in the search of patterns which are consistent with the underlying distance. For instance, the pattern $*ab*$ obtained by generalising the words w_1 and w_2 looks excessively general w.r.t another “consistent” pattern such as $*abab*$. Although the idea of generality has been deeply studied when data is represented by means of first-order logic [11] it does not happen the same for the rest of sorts of data, and especially when data is in a metric space. Thus, we propose a general way (inspired on the MML/MDL principle [14]) for defining minimal distance-based generalisation operators (mg operators). We have applied this framework to several data sorts: sets, lists, graphs,... (see [3, 4]).

In this paper, we focus on first-order objects (atoms and Horn clauses), which are embedded in a metric space. We show that Plotkin’s lgg [11] is a mg operator for atoms using the metric defined in [13]. This means that the mg patterns computed by lgg can be used as a consistent explanation for clustering data employing this distance. Then, we try to extend this result to Horn clauses (more precisely, sets of atoms) and we show that the direct use of the lgg for clauses does not yield a distance-based generalisation operator using the metric defined in [12]. Consequently, we introduce a new mg for clauses. This sets out a scenario where some (but not all) generalisation operators and some (but not all) metric spaces used in ILP work well together. This suggests the applicability of other generalisation operators (as the one introduced in this paper) in ILP.

The paper is organised as follows. Section 2 presents some classical definitions for first-order objects. Section 3 introduces the framework for distance-based generalisation operators. The notion of mg is studied in the next section. Section 5 analyses several mg for atoms, Plotkin’s lgg being one of them. Section 6 extends the result to sets of atoms (i.e. clauses), through the definition of a new mg which cannot be the lgg for clauses, since the latter is not distance-based. Finally, the last section presents the conclusions, and mentions several possibilities for applications, some open problems and future work.

2 Preliminaries

The most common metric spaces (e.g. the real numbers using the absolute difference for distance) normally have some properties (e.g. completeness) upon which new useful concepts (e.g. Cauchy sequence) and operations (e.g. to compute the limit of a sequence) can be established. Therefore, if we aim to define a generalisation operator for data embedded in a metric space, we could wonder whether this metric space we are working with should satisfy certain conditions.

In this line, we propose the following reasoning: given two objects, x and y , we notice that a concept z generalises x and y if z somehow collects the common features of x and y . In fact, z is likely to represent other elements besides x and y . That is, more formally, z can be viewed as a set including x , y and probably other elements. For instance, imagine that x is a regular pentagon and y is an equilateral triangle. Then, z could be the concept of regular polygons with $3 \leq n \leq 5$ sides. Furthermore, if we analyse those elements included in a generalisation, we can see that a generalisation contains transformations which permits gradually convert an element e into other

and vice versa. Namely, the pentagon of our example can be transformed into a regular triangle by means of two basic steps. First, we convert the pentagon into a square (the square is a particular case of z) by removing one of its sides and then we do the same over the square. From this observation, we are interested in those metric spaces upon which the notion of gradual transformation can be formalised. In advance, by gradually transformation we mean a sequence of elements $\{x_i\}_{i=1}^n$ in a metric space such that for every i , x_i is quite close, and hopefully quite similar, to $x_{(i+1)}$.

Traditionally, the spaces where the notion of gradual transformation can be more naturally defined are the continuous ones (e.x. \mathcal{R}^2 with the Euclidean distance), since a continuous curve linking two given elements x_1 and x_n can be understood as a gradual transformation of x_1 into x_n . However, the metric spaces employed in structured learning are usually discrete (discontinuous), for instance, the space of lists of symbols using the edit distance. Then the concept of continuous curve is missing. But if we require the space of lists to be mathematically continuous, this would imply to map lists into elements belonging to a continuous space (e.g. \mathcal{R}^n). This strategy sounds similar to how kernel methods work [7]. The problem is that defining a generalisation operator in a feature space may not have an easy interpretation in the input space. Hence, it seems to be better, at least for the sake of the comprehensibility, to work on the input space.

The proposal is based on the well-known topological concept of connexion: we intuitively consider that a set is connected when it is not made of separated pieces, i.e. it is not divided. The idea of “naturally divisible” can be easily handled in a continuous space. For instance, if we observe Figure 1, it is intuitively clear that the square $X = [0, 1] \times [0, 1]$ is made of only one piece. However, it does not happen the same with the object $Y = Y_1 \cup Y_2 \cup Y_3$. It is impossible to go, in a continuous way, from an element in Y_1 to an element in Y_2 without “stepping” outside Y_1 , Y_2 and Y_3 . Necessarily, we have to jump from one “piece” or component Y_i to another one Y_j , missing the elements in between.

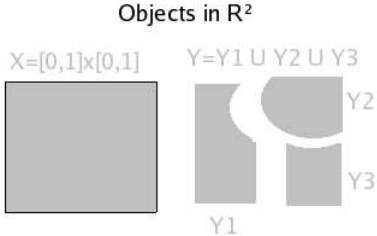


Figure 1: **(Left picture)** The closed square is not a divided object. We can connect any two points by means of a continuous path without leaving the object. **(Right picture)** In this case, the object is made of three separated pieces ($\{Y_1, Y_2, Y_3\}$). There does not exist a continuous path included in Y which connects two points belonging to different components.

Let us see what happens with discrete spaces. By definition a discrete metric space is divided because it is made up of separated pieces, its elements and, between them, there is nothing. But if the generalisations we are looking for implicitly hide a “gradual transformation”, we need a not divided space. The trick will consist of revisiting the concept of “divided” in the following way. Ideally, we could consider that two parts are divided if they are further than the shortest step which is possible in the space. For example, let us consider the finite metric spaces (X, d) and (Y, d) depicted in Figure

2. Although both spaces are discrete, the distribution of their elements is completely different. The elements of X are grouped into clusters whereas the elements of Y are not. The clusters will play the role of the separated pieces in the left figure. Note that it is impossible to “travel” from one element to another belonging to a different cluster taking minimum length steps. A longer step is always needed. It contrasts with space Y , where these minimum steps are always possible.

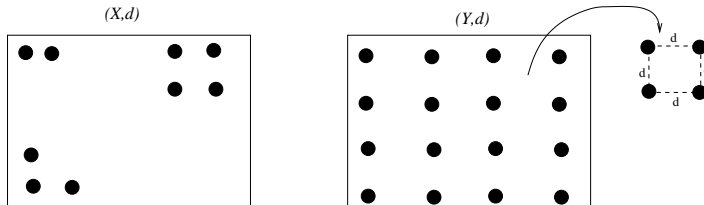


Figure 2: **(Left picture)** The elements of X are grouped into clusters. **(Right picture)** The elements of Y are uniformly distributed.

The concept of a “gradual transformation” is more intuitive in Y since equally significant changes can convert one element into another. In other words, spaces like Y have the advantage that the generalisations can be interpreted in a more reasonable way. Let us illustrate this claim with the example in Figure 3. We want generalise the elements a and b . If these elements belong to X , not too much intuitive generalisations will be obtained because the way in which the elements of X are distributed. Note that a generalisation tries to stand out what properties two elements have in common. In this case, a generalisation should include some elements close to a (e.g. $a1$), others close to b (e.g. $b1$) and those which let one go from a to b step by step. But latter do not exist and, thus, it is not possible gradually to go from a to b , there is no notion of middle point and it is more difficult to find the similar traits between a and b . However, this problem does not occur in space Y . We can always find a sequence of elements, quite close one to each other, to go from a to b . For example, in Figure 3, the most basic sequence is formed by a , c and b .

Therefore, we are focusing on those metric spaces whose elements allow gradual transformations (in other words, the elements are “uniformly distributed”) since generalisations can be performed more easily, and as we will see, the extracted patterns can be more regular and more comprehensible. The following definitions let us formalise this property.

Definition 1 (Infimum Distance) Let (X, d) be a metric space. We define the infimum distance of X , and we denote it by $I(X)$, as

$$I(X) = \inf\{d(x, y) : \forall x, y \in X, x \neq y\},$$

where \inf stands for the infimum of a set. That is, $I(X)$ is the greatest lower bound of X not necessarily belonging to it.

Note that if (X, d) is a continuous metric space then $I(X) = 0$. However, the reciprocal is not true. There are discrete metric spaces where $I(X) = 0$, as we show in Example 1.

Example 1 Let (X, d) be a metric space where $X = \Sigma^*$ ($\Sigma = \{a, b\}$) and d is defined as follows¹. Given to words $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_m$ ($n, m > 0$) belonging to X ,

¹This distance is a slight variation of the fractal distance [].

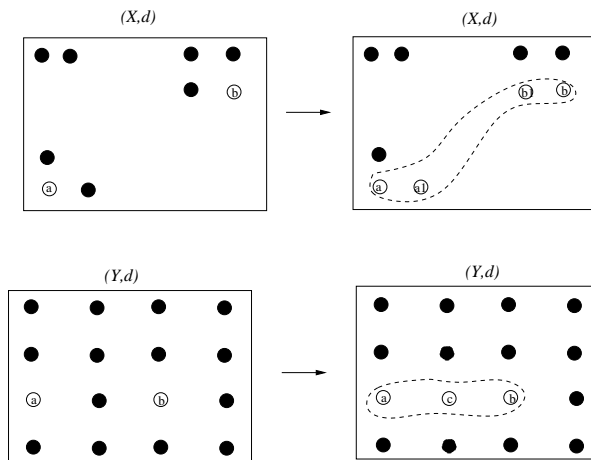


Figure 3: Generalisation of the elements a and b . **(Top-left picture)** The elements belong to different clusters. **(Top-right picture)** A generalisation should include the elements a , $a1$, b and $b1$. **(Bottom-left picture)** Now, the elements a and b belong to Y . **(Bottom-right picture)** Intuitively, the intermediate element c permits to directly connect a and b . Therefore, these elements could be included by a generalisation.

$$d(x, y) = \begin{cases} 0 & , \text{ if } x=y \\ \frac{1}{2^i} & , \text{ if } x_j = y_j \ (j = 1, \dots, i) \text{ and } x_{i+1} \neq y_{i+1} \end{cases}$$

The infimum distance of X is equal to zero since for any $\epsilon > 0$, we can always find two words x and y belonging to X such that $d(x, y) < \epsilon$. The space (X, d) is not locally connected.

Unfortunately, some real metric spaces might have small deviations (measurement errors, small random effects, etc.), and the infimum distance happens with some ideally cases. In these situations, it might be more flexible to be able to fix this “infimum” proximity value which we call resolution and we denote by ρ . Unless specified otherwise, $\rho = I(X)$.

Next we introduce the concept of δ -path. It will be employed to formalise the intuitive idea of “gradual transformation”.

Definition 2 (δ -path) Let (X, d) be a metric space and let δ be a real number greater than zero. We will say that a finite sequence of elements $P = \{x_i\}_{i=0}^{n>0}$ belonging to X is a δ -path if $d(x_i, x_{i+1}) \leq \delta$ for all $0 \leq i \leq n-1$. We can represent this sequence as $x_0 \rightarrow x_1 \rightarrow \dots \rightarrow x_n$

In what follows, we will say that the elements x and y are δ -path connected when there exists a δ -path $P = \{x_i\}_{i=0}^{n>0}$ such that $x_0 = x$ and $x_n = y$. Intuitively, it means that we can “transform” x into y (through P) by means of successive changes which must be equal or smaller than δ .

Example 2 Let (X, d) be a metric space where X is the set of finite words over the two-symbol alphabet $\Sigma = \{a, b\}$ and d the edit distance permitting insertions and deletions only. If $x = aab$ and

$y = bba$, then we can find a sequence according to Definition 2, such that:

$$x = aab \rightarrow ab \rightarrow b \rightarrow bb \rightarrow bba = y,$$

and this sequence is a 1-path.

The idea from this example is that we can set $\delta = I(X)$ and hence we have the definition of a path with minimum steps. However, this idea would only work with those metric spaces X such that $I(X) > 0$. In order to cover every metric space, it will be enough to state what a 0-path is:

Definition 3 (0-path) Let (X, d) be a metric space with $I(X) = 0$. We will say that a sequence of points P belonging to X is a 0-path, if P is the image of a continuous function γ defined over the closed interval $[0, 1]$ if the space is continuous or over the rational number in $[0, 1]$ otherwise, such that $\gamma([0, 1]) = P$.

Summing up, a 0-path is just the well-known concept of a curve (indiscrete or discrete). That is, given two points x and y belonging to X , we will say that x and y are 0-path connected, if there exists a continuous function γ such that $\gamma(0) = x$ and $\gamma(1) = y$ (see Figure 4).

For the sake of simplicity, in what follows we call δ -path to every path with $\delta \geq 0$. When it is needed, we explicitly distinguish between $\delta > 0$ and $\delta = 0$.

A useful property concerning a δ -path, and that we will need in the next section, is its length.

Definition 4 (Length of a δ -path) Let (X, d) be a metric space and let $P = \{x_i\}_{i=0}^{n \geq 0}$ be a δ -path in X , then the length of P ($L(P)$) is defined as follows²:

$$L(P) = \begin{cases} \sum_{i=0}^{n-1} d(x_i, x_{i+1}), & \text{if } \delta > 0 \\ \sup\{\sum_{i=0}^{n-1} d(\gamma(t_i), \gamma(t_{i+1})) : \forall n \in \mathbb{N} \text{ and } 0 = t_0 < t_1 < \dots < t_n = 1\}, & \text{if } \delta = 0 \end{cases}$$

For instance, all paths in Figure 4 have length 5.

We are now ready to define the metric spaces we will work with.

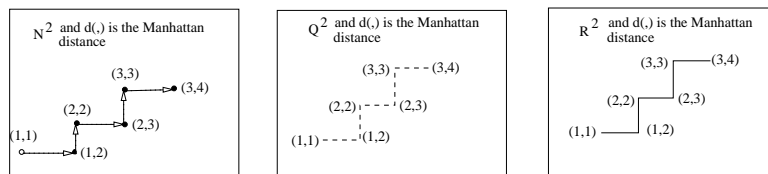


Figure 4: **(Left picture)** A 1-path (a finite collection of points) connecting the elements $(1, 1)$ and $(3, 4)$. **(Centre picture)** A 0-path (a discrete curve) connecting the elements $(1, 1)$ and $(3, 4)$. **(Right picture)** A 0-path (an indiscrete curve) connecting the elements $(1, 1)$ and $(3, 4)$.

²sup stands for the supremum of a set, that is, its least upper bound.

Definition 5 (ρ -Globally connected³) Let (X, d) be a metric space. We will say that X is a globally-connected metric space at resolution ρ or ρ -globally-connected, if for every pair of items x and y belonging to X , they are δ -path connected where $\delta \leq \rho$.

Again, if this definition holds for $\rho = I(X)$, we simply say that X is a globally connected metric space.

Example 3 The metric space used in the Example 2 is a globally-connected space. Immediately, $I(X) = 1$, and given two finite words $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ ($n, m > 0$), we can find a trivial 1-path connecting them:

$$x = x_1 \dots x_n \rightarrow x_2 \dots x_n \rightarrow \dots \rightarrow x_n \rightarrow x_n y_1 \rightarrow y_1 \rightarrow \dots \rightarrow y_1 \dots y_{m-1} \rightarrow y = y_1 \dots y_m$$

For indiscrete metric spaces, Definition 5 matches the well-known topological definition of a *path-connected* metric space. As the metric spaces we will work with will be always *globally-connected*, for the sake of simplicity, we will simply refer to them as *connected* metric spaces.

Before going on, it is enlightening to see that some “dense” spaces are not necessarily connected. For instance, the following example shows that “strange” spaces are discarded by Definition 5.

Example 4 The metric space introduced in Example 1 is not connected. Given any two words x and y belonging to the space, it is impossible to find a continuous function γ such that $\gamma(0) = x$ and $\gamma(1) = y$. From a practical point of view, this latter distance is quite useful if we pursue to distinguish those words beginning by a or by b . Fortunately, although it is out of the scope of this paper, it can be redefined in such a way that the elements of the space (X, d) are uniformly distributed.

Summing up, Definition 5 means that the space is made of one piece, as we discussed at the beginning. This idea is key to introduce the concept of connected sets in discrete metric spaces, i.e. the extension of the idea to groups of objects in the space.

Definition 6 (ρ -Connected set) Let (X, d) be a connected metric space. We will say that $Y \subset X$ is a ρ -connected set, if for every pair of elements x and y belonging to Y there exists a ρ -path P connecting them and P is included in Y . As $\rho = I(X)$ we simply say that Y is a connected set.

Informally speaking, a connected set is a set made of one piece. Obviously, every set in the metric space is made of one piece (connected) or several pieces (disconnected).

Example 5 Let us illustrate what a connected set is in a discrete space X . For this purpose, we will take the space depicted in the Figure 3. This space is $I(X)$ -globally connected thus we set $\rho = I(X)$. As we can appreciate in the left picture (see Figure 5), the set A is connected since for every pair of elements belonging to A , there exists at least one ρ -path (dashed lines) included in A connecting them. However, the set B on the right is disconnected. The elements x and y are not ρ -path connected.

³Note that there exist some peculiar distance functions in the sense that the return value is a tuple of numbers instead of one single number (see the distance function defined in [13]). Definition 5 can be easily adapted to take these cases into account, since the image set of the mentioned distance function is endowed a total order.

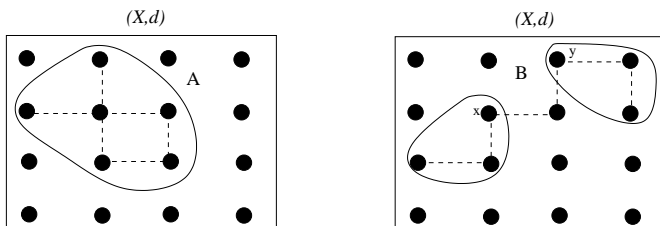


Figure 5: **(Left picture)** The set A is connected since its elements are δ -path connected. **(Right picture)** Nevertheless, the set B is disconnected because the elements x and y are clearly not ρ -path connected. Note that this set B has two connected components.

3 Distance-based generalisation operators

In our approach, connected sets play a fundamental role in order to define a generalisation operator since a generalisation of a group of elements will be just a connected set containing them. Along with the possibility this property (connection) gives in the sense that a generalisation can be viewed as a collection of continuous transformations among the generalised elements, there is another stronger justification which motivates the use of connected sets. From the inductive learning point of view, connection turns interesting because it will permit us reject much too specific generalisations. See both examples below.

Example 6 *The condition of being connected makes full sense, for example, in the context of conceptual clustering. Basically, clustering is a division of data into sets of similar objects, the so-called clusters [15]. Each cluster is usually represented by a prototype. Intuitively, those elements of the metric space, which potentially will be associated to one given prototype, would be in a set which is made of one "piece". Then, it seems natural a generalisation of a set of elements belonging to one cluster to be a connected set rather than a disconnected one.*

More graphically, if a generalisation G of a set of elements in the cluster E is a disconnected set, namely, $G = \cup_{i=1}^n G_i$ where G_i are the different connected components (see Figure 6), then there might be some elements of the metric space which are non-covered by G and are placed among the connected components (in Figure 6, these elements are represented by black circles). Some of these non-covered elements will be nearer, and consequently, will be more similar to the elements in E than some of the elements belonging to G_1 or G_3 are. Therefore, the generalisation of E should include them.

Example 7 *Let us suppose we are clustering graphs with the distance d for graphs defined in [1]. Imagine that each graph represents an organic compound and we would be interested in extracting some patterns saying which kind of molecules can be found in a cluster. One of the obtained clusters consisting of two molecules (m_1 and m_2) is depicted on the top Figure 7.*

Let us obtain a pattern explaining the data distribution in this cluster. For this purpose, one could think on the pattern p (bottom-right Figure 7) saying all the molecules with a cyclopropane structure and an extra atom. But, is this pattern much too specific? Considering that the molecules are really graphs in the space (G, d_1) where G is the set of all the possible labelled graphs over an alphabet of labels, we could think that the pattern p overfits the data since the cyclopropane molecule, which

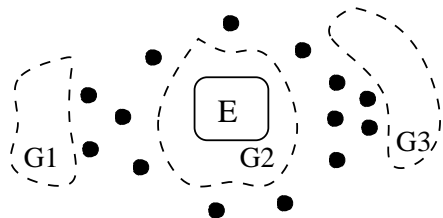


Figure 6: Using a non-connected set as a generalisation of a cluster of elements E

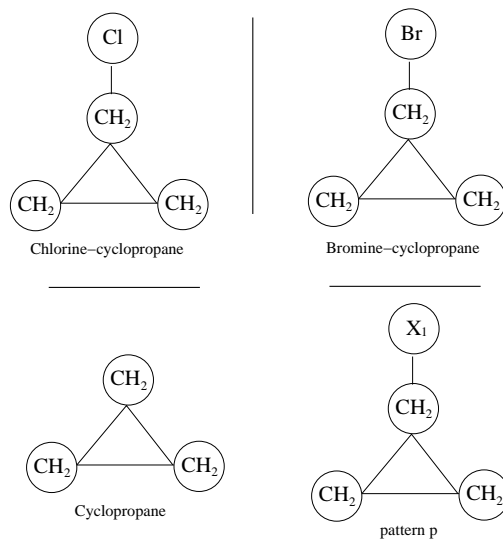


Figure 7: The pattern p does not cover the cyclopropane molecule.

would be placed “between”⁴ m_1 and m_2 , that is, $d_1(m_1, \text{cyclopropane}) = d_1(\text{cyclopropane}, m_2) = 1$, is not covered by this pattern. Perhaps, a more natural pattern would be that one saying “all the molecules built from cyclopropane”.

Both overfitting problems described in the two previous examples can be analysed in terms of connected sets. For instance, regarding the latter one, we know that (G, d) is a 1-connected space (see Proposition 1 in [5]), and clearly, the set given by $Set(p)$ is not connected because the elements m_1 and m_2 are not connected by means of a 1-path included in $Set(p)$. That is, m_1 and m_2 are, at least, 2-path connected since $d_1(m_1, m_2) = 2$. However, it is easy to see that the Set (“all the molecules built from cyclopropane”) would be connected.

Now, we are in conditions of introducing the formal definition of a generalisation operator.

⁴Formally, given three elements x, y and z belonging to a metric space (X, d) , we say that z is “between” x and y if $d(x, y) = d(x, z) + d(z, y)$.

Definition 7 (Generalisation operator) Let (X, d) be a connected metric. For every finite set E of elements in X , a generalisation operator Δ is a function $\Delta : E \rightarrow S \in 2^X$ such that $E \subset S$ and S is a connected set.

Intuitively, the generalisation of a finite set of elements E in a connected metric space (X, d) could be extensionally defined as a connected set that contains E . But, from a comprehensibility point of view, we must highlight that our concept of generalisation should be associated to a family of patterns \mathcal{L} , where each pattern represents a set in the connected metric space (X, d) . Note that a generalisation which is just expressed as a set of elements (e.g. $\{2, 4, 6, \dots\}$) is much less useful than a generalisation which has an associated pattern (e.g. $\{x : \text{odd}(x)\}$). That is, a pattern $p \in \mathcal{L}$ will be an intensional (and hopefully “comprehensible”) manner of denoting the set of all the elements in X which are covered by p ($\text{Set}(p)$). Additionally, viewing patterns as the sets they denote, we can use the well-known mathematical operations for sets. For example, we can say that a pattern p_1 is included in a pattern p_2 if $\text{Set}(p_1) \subseteq \text{Set}(p_2)$, or we can say that an element $x \in X$ belongs to the pattern p (or is covered by the pattern p), if $x \in p$. Note that the same set can have several patterns which denote it. Furthermore, depending on our pattern language \mathcal{L} we will be able to express some sets as generalisations but some others not. For instance, if our family of patterns in the metric space \mathcal{R}^2 is made of all the possible squares of size 1×1 , the concept of a 2×2 square can not be expressed. An important reason to introduce the notion of \mathcal{L} is because not all the connected sets in one given space X will usually have an intuitive pattern associated to them (just figure out all the possible connected sets in \mathcal{R}^2). For this purpose, \mathcal{L} will be defined according to the problem to be solved, and very specially, on the kind of patterns the user can understand. Nevertheless, in the worst case, \mathcal{L} can always be defined as 2^X if we do not have any representation bias. So, instead of extensional generalisation operators, we will work with intensional operators.

Definition 8 (intensional generalisation operator) Let (X, d) be a connected metric space and let \mathcal{L} be a pattern language. For every finite set E of elements in X , an intensional generalisation operator Δ is a function $\Delta : E \rightarrow p \in \mathcal{L}$ such that $E \subset \text{Set}(p)$ and $\text{Set}(p)$ is a connected set.

Therefore, an intensional generalisation operator simply maps sets of elements E into patterns representing connected sets. In what follows, although we do not explicitly indicate it, we will always work with intensional operators.

Note that this latter definition permits avoid some specific generalisations but not others, which would hardly be guided by the data. Regarding Figure 8, the generalisation of two elements belonging to \mathcal{R}^2 could be something as simple as a straight line or something as complicated as the most intricate curve connecting them. Both ones are connected sets, but the curve depicted on the left seems much too complicated as a result of the generalisation of A and B . In next section, this problem will be addressed in-depth detail.

At this point, the most interesting aspect from the figure above is that the generalisation of A and B expressed as a straight line somehow explains the value of $d(A, B)$. But this feature does not turn out specially clear in the curve depicted on the left. It is due to the straight line trivially contains a path P such that $L(P) = d(A, B)$, or put in a different way, the generalisation covers those elements exactly placed between A and B . This will be the criterion, we are going to use, in order to formalise the concept of distance-based generalisation operators.

Initially, the definition for the specific case of binary generalisation operators $\Delta(\{A, B\})$ seems immediate, in the sense that, $\Delta(\{A, B\}) = p$ is distance-based if $\text{Set}(p)$ is connected and includes

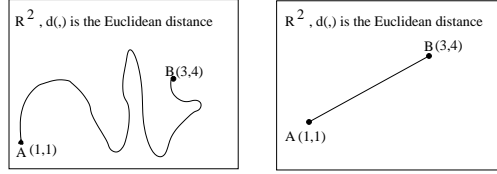


Figure 8: Generalising the elements $A(1,1)$ and $B(3,4)$. **(Left picture)** The generalisation of A and B is given by a curve. **(Right picture)** The generalisation of the elements A and B is the straight line connecting both.

all the elements between A and B . But, what if the set of elements to be generalised is greater than 2. Which paths should be included by $Set(p)$?

It is quite reasonable to assume that the problem where Δ is going to be used forces to include some specific paths. For instance, in clustering, an element is compared to the prototypes before being assigned to one specific cluster. So, the elements in a cluster E have been compared only to its prototype but not among them. Therefore, if $\Delta(E) = p$ then $Set(p)$ should include those shortest paths between the prototype and the rest of elements.

As it is convenient to indicate which paths must be included, the concept of “nerve” of a set of elements E is needed. In this way, a nerve of E , denoted by $N(E)$, is simply a connected⁵ graph whose vertices are the elements belonging to E .

Definition 9 (Nerve function) Let (X, d) be a metric space and let S_G be a set of undirected graphs, a nerve function $N(\cdot) : 2^X \rightarrow S_G$ maps every finite set $E \subset 2^X$ into a graph $G \in S_G$, such that each element e in E is unequivocally represented by a vertex in G and vice versa. We say the obtained graph $N(E)$ is the nerve of E .

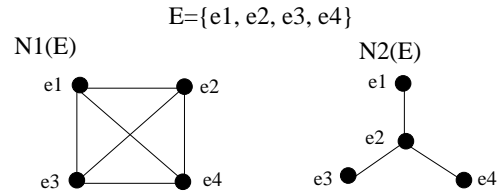


Figure 9: Some examples of nerves for the set $E = \{e_1, \dots, e_4\}$. **(Left picture)** $N_1(E)$ computes a complete graph. **(Right picture)** $N_2(E)$ computes a 3-star graph

The nerve will simply indicate which paths connecting the elements in E should be included by $Set(\Delta(E))$.

Definition 10 (Distance-based generalisation operator) Let (X, d) be a connected metric space and let \mathcal{L} be a pattern language. Given a generalisation operator Δ , we will say that Δ is

⁵Here, the term connected refers to the well-known property for graphs.

a distance-based generalisation operator, if for every $E \subseteq X$ there exists a nerve $N(E)$ such that, for every pair of elements x, y in E which are directly linked in $N(E)$, $Set(\Delta(E))$ includes **all** $I(X)$ -path P connecting x and y such that $d(x, y) = L(P)$.

Let us see an example for different sets of elements in (\mathcal{R}^2, d) where d is the Manhattan distance⁶. For the binary set $E = \{A(1, 1), B(3, 4)\}$, the only possible nerve $N(E)$ is the graph with a single edge connecting the vertices which represents the elements A and B . A distance-based generalisation operator for $E = \{A, B\}$ must compute a generalisation $\Delta(\{A, B\}) = p$ such that $Set(p)$ include all the elements in between A and B . Unlike the metric space defined by the Euclidean distance, all the elements between A and B are placed in the rectangle delimited by A and B . Thus, $Set(p)$ must included this rectangle.

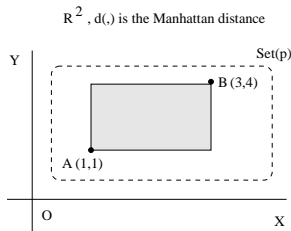


Figure 10: Generalising the elements $A(1, 1)$ and $B(3, 4)$.

For more than two elements $E = \{A, B, C\}$, put another example !!!!!!!!!!!!!!!!!!!!!!!
As we said, the nerve function $N(\cdot)$ can be defined depending on the problem we are addressing. In fact, it will be the common way of working. That is, first a nerve function $N(\cdot)$ is fixed, and then a generalisation operator Δ is defined to fit N . The notation Δ_N will be employed to make clear that Δ is distance-based for the function nerve N .

For instance, for a conceptual clustering task, a reasonable $N(\cdot)$ definition would be that one computing a star graph with the elements of a cluster E such that the prototype of the cluster would be mapped into the central vertex in the graph. The edges would represent the pair of elements in the cluster which have been compared.

4 Minimal distance-based generalisation operators

Another issue related to the generalisation operator is to determine when it performs the least general generalisation (*lgg*, in short). It will be an important issue if we want the generalisations to “fit” a group of elements as much as possible. Despite the *lgg* is a widely studied concept in the field of *Inductive Logic Programming* (ILP) [10], it does not happen the same when data is not described by means of first-order atoms. Thus, the following observations are in some way an attempt to extend the notion of *lgg* for different sorts of data, specially when data is in a metric space.

Here, we will proceed as follows. First, we will establish a criterion to determine, given two patterns computed by the distance-based generalisation operators $\Delta(E)$ and $\Delta'(E)$ respectively, which one is less general. Finally, the least general distance-based generalisation operator Δ will be

⁶Given two points $A(a_1, a_2)$ and $B(b_1, b_2)$, the Manhattan distance is defined as $d(A, B) = |a_1 - b_1| + |a_2 - b_2|$.

that one such that for every set E and for every distance-based operator Δ' , the pattern $\Delta(E)$ is less general than $\Delta'(E)$. It might the least general Δ not be unique. For this reason we say that Δ is a minimal distance-based generalisation operator (*mg operator*, in what follows).

In order to formalise our proposal, we could utilise the inclusion operation between sets (\subset) as a “mechanism” to compare how general two generalisations are. That is, a generalisation of E computed by $\Delta(E)$ is less general than a generalisation computed by $\Delta'(E)$, if $Set(\Delta(E)) \subset Set(\Delta'(E))$. However, this leads to several problems.

- Most generalisations are incomparable, since neither $Set(\Delta(E)) \subseteq Set(\Delta'(E))$ or vice versa: It is due to the inclusion operator between sets (\subset) ignores the underlying distance. Thus, it is possible that for two non-comparable generalisations of a set E given by the patterns $\Delta_1(E) = p_1$ and $\Delta(p_2)$, $Set(p_1)$ fits better E than $Set(p_2)$ does, as we show next. It is illustrated in Figure 11. Both patterns are not comparable via the inclusion operator. However, taking the distances from the elements to the border of the $Set(p_i)$ into account (the arrows labelled as d_{ij}), we could discriminate between both patterns.

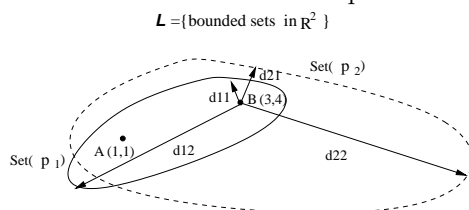


Figure 11: Two patterns p_1 and p_2 generalising $E = \{A, B\}$. The pattern p_1 fits E in a better way. However, both patterns are incomparable via the inclusion operator.

- The minimal generalisation may not exist for some pattern language \mathcal{L} : For instance, consider \mathcal{R}^2 with the Euclidean distance and \mathcal{L} as the set of all the rectangles in \mathcal{R}^2 . Then, the generalisation of n points belonging to \mathcal{R}^2 would be something as simple as a rectangle containing them (see left Figure 12). However, in some contexts, it would be preferable to obtain some slightly more elaborated generalisations. For instance, the one depicted on the right of Figure 12. Note that, in this case, a more expressive pattern language is needed (\mathcal{L}' is the set of all the rectangles in \mathcal{R}^2 along with their finite unions). But the more complex a generalisation is, the less intelligible it is and, the higher the chance of overfitting is.

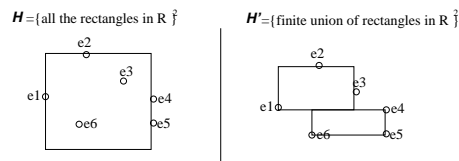


Figure 12: **(Left picture)** A naive generalisation of the elements $\{e_1, \dots, e_6\}$. **(Right picture)** A more elaborated generalisation considering a more expressive family of patterns.

But working with \mathcal{L}' , the following situation can be given. If we look at Figure 13 given

the pattern p_0 computed by $\Delta_0(E)$, we can always define another operator Δ_1 such that $Set(\Delta_1(E)) \subset Set(\Delta_0(E))$ and so on. Note that, it is enough to draw a connected chain of smaller rectangles which is included in the previous generalisation and links both A and B . Therefore, if we define the generality in terms of the inclusion between sets, then the mg operator does not exist in this case.

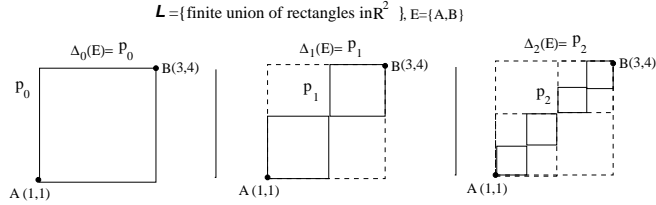


Figure 13: Generalising $E = \{A(1, 1), B(3, 4)\}$ by means of Δ_i .

Therefore, both drawbacks lead to introduce a more abstract generality criterion. by a more abstract principle. The possibility that we explore is the next one. Note that the level of “complexity” of a pattern is reasonable only if a sufficient number of examples justify it, as the *MDL/MML* principle states. In the case above, a pattern made of thousand of small rectangles becomes hardly comprehensible, even if it fits the evidence E really well. Hence, it could be more interesting to reach a trade-off between minimality and comprehensibility. For this purpose, we introduce a special function, called the cost function and denoted by $k(E, p)$. This takes both the complexity of the pattern p and how good the pattern fits E into account. Returning to Figure 13, the cost function $k(E, p)$ should be able to discriminate those unnecessary complicated patterns. Hence, the specificity of a generalisation will be defined in terms of a cost function instead of the inclusion between sets.

Now, we are in conditions of formally introducing the concept of mg operator. Previously, we will define what a cost function is.

Definition 11 (cost function) Let (X, d) and \mathcal{L} be a ρ -connected metric space and a pattern language, respectively. We will say that the mapping $k(E, p) : (E, p) \in 2^X \times \mathcal{L} \rightarrow \mathcal{R}$ is a cost function, if for every pattern $p \in \mathcal{L}$ such that $E \subset Set(p) \subsetneq X$ and $Set(p)$ is ρ -connected, then $k(E, p) < \inf$.

Most of the $k(E, p)$ functions that we are going to use, will be expressed as the sum of the auxiliary functions $c(p)$ (it measures how complicated the pattern is) and $c(E|p)$ (it measures how the pattern fits the data E). One novel point in our approach is that this latter function is expressed in terms of the distance employed. A coherent cost function $k(E, p)$ expressed as the sum of $c(E)$ and $c(E|p)$ should satisfy one of the two properties below:

1. Covering the sample: for every set E , given two patterns $p_0 = \Delta_0(E)$ and $p_1 = \Delta_1(E)$ such that $Set(p_0) \subset Set(p_1)$ and $c(p_0) = c(p_1)$ then, $c(E|p_0) \leq c(E|p_1)$.
2. Covering the sample using the minimum hypothesis: for every set E , given two patterns $p_0 = \Delta_0(E)$ and $p_1 = \Delta_1(E)$ such that $Set(p_0) \subset Set(p_1)$ and $k(E, p_0)$ performs the minimum. If $c(p_0) = c(p_1)$, then $c(E|p_0) \leq c(E|p_1)$.

Property 1 solves to the following problem: if we have two equally ”complicated” hypothesis, which one should be chosen? Intuitively, the one fitting the sample better. However, this might be difficult

to achieve sometimes. Then, we could relax this condition by forcing that only the optimal pattern satisfies it. This leads to Property 2. In other words, if p_0 is an optimal pattern covering E , then there should not exist any equally complicated hypothesis included in p_0 .

Before going on, let us introduce some interesting definitions of $k(E, p)$ and show how they work. Concretely, we are focusing on those cost functions expressed in terms of $c(p)$ and $c(p|E)$. As $c(p)$ measures how complex a pattern is, this function will strongly depend on the sort of data and the pattern space \mathcal{L} we are handling. For instance, consider a closed interval. If the generalisation of two real numbers is a closed interval containing them, then the complexity of the interval could be its length. On the contrary consider a graph, if the elements to be generalised are two graphs, and the generalisation is based on the idea of common subgraph, then the number of cycles in the shared subgraph could be a measure of its complexity (see Table 1).

Sort of data	\mathcal{L}	$c(p)$	Example
Numerical	Closed intervals	Length of the interval	$c([a, b]) = a - b $
Finite lists over an alphabet of symbols Σ	Lists built from an alphabet Σ and a special alphabet V of variables.	Number of items in the list	$c(X_0abX_1b) = 5$
First order predicates	Herbrand base with variables	Number of different variables	$c(p(X, X, Y, a)) = 2$
"	"	Number of symbols	$c(p(X, X, Y, a)) = 5$
Any	Any	Constant function	$\forall p \in \mathcal{L}, c(p) = k$

Table 1: Some definitions of the function $c(p)$ for several sorts of data.

Now, let us see some definitions of $c(E|p)$. All of them must be based on the underlying distance. In fact, all the definitions we present here are based on the well-known concept of border of a set⁷. Intuitively, if a pattern p_1 fits better E than a pattern p_2 , then the border of p_1 (∂p_1) will somehow be nearer to E than the border of p_2 (∂p_2) (see Figure below).

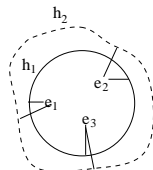


Figure 14: The pattern p_1 fits better E than p_2 and consequently, ∂p_1 is “nearer” to E than ∂p_2 .

As the border of a set exists in every metric space, the function $c(E|p)$ will be much more independent of the sort of data we are handling than $c(p)$ is. Therefore, it motivates that several definitions of $c(E|p)$ can be employed for different sorts of data, as we show in Table 2.

⁷Let (X, d) be a metric space. We will say that an element e belonging to set $A \subseteq X$ is a border point, if for every $\epsilon > 0$, $B(e, \epsilon)$ is not totally included in A . According to the standard notation, the border of a set A will be denoted by ∂A .

Sort of data	\mathcal{L}	$c(E p)$	Example
Any	Any	$\sum_{\forall e \in E} r_e$ $r_e = \inf_{r \in RB(e, r_e)} r_e \not\subset Set(p)$	Figure 15 (left)
Any	Any	$\sum_{\forall e \in E} r_e$ $r_e = \sup_{r \in RB(e, r_e)} r_e \subset Set(p)$	Figure 15 (left)
Any	Any	$\sum_{\forall e \in E} \min_{e' \in \partial Set(p)} d(e, e')$	Figure 15 (centre)
Any	$Set(p)$ represents an bound set	$\sum_{\forall e \in E} \min_{e' \in \partial Set(p)} d(e, e') + d(e, \text{furthest point} \in \partial Set(p))$	Figure 15 (right)

Table 2: Some definitions of the function $c(E|p)$.

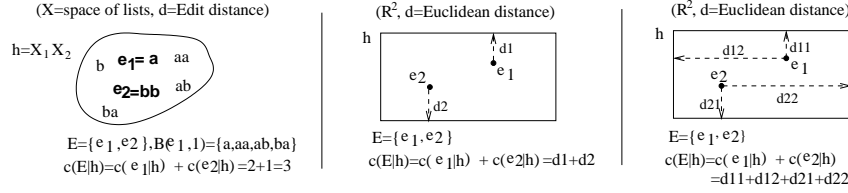


Figure 15: Some pictures illustrating each definition of $c(E|p)$ collected in Table 2. **(Left picture)** The space of finite lists with the Edit distance without substitution. The closed ball $B(e_1, 1)$ is included in the pattern h . **(Middle picture)** The pattern h is a rectangle covering e_1 and e_2 . The terms d_i represents the distance from e_i to its nearest point in ∂h . **(Right figure)** This time, both the distance from e_i to its nearest and furthest point in ∂h (d_{i1} and d_{i2}) are taken into account.

In general, the functions $c(p)$ and $c(E|p)$ can be combined obtaining a more flexible way of defining $k(E, p)$ (see Example ??). The next immediate proposition will become really useful throughout the remaining sections of this work.

Proposition 1 *Let $k(\cdot, \cdot)$ be the cost function defined as $k(E, p) = c(p) + c(E|p)$ where $c(p)$ is the constant function and $c(E|p)$ is one of the functions described above. Given two patterns $\Delta(E) = p$ and $\Delta'(E) = p'$, if $Set(p) \subset Set(p')$ then $k(E, p) \leq k(E, p')$.*

Proof 1 *It is directly from the definition of the cost function employed in this proposition. Clearly, if $Set(p) \subset Set(p')$ then $c(E|p) \leq c(E|p')$ and as $c(p)$ is the constant function, we have that $k(E, p) \leq k(E, p')$.*

Now, we are in conditions of introducing the definition of mg operator.

Definition 12 (Minimal distance-based generalisation operator) *Let (X, d) be a ρ -connected metric space and let Δ be a distance-based generalisation operator defined in X using the pattern language \mathcal{L} . Given a finite set of elements $E \subset X$ and a cost function $k(E, p)$, we will say that Δ is a mg operator for $k(\cdot, \cdot)$ in \mathcal{L} , if for every distance-based generalisation operator Δ' ,*

$$k(E, \Delta(E)) \leq k(E, \Delta'(E)), \text{ for every finite set } E \in X.$$

Computing the mg operator will be complicated in general. The definition above implicitly forces to know some property about the set of all the possible distance-based generalisations $\{\Delta'_i\}_{i \geq 1}$ in

order to ensure that one of them is just the *mg*. This is not immediate because of the high variety of nerve functions $N(\cdot)$ which can be defined. For a set E , several operators $\Delta(E)_{N_1(E)}$ can be obtained by simply changing the definition of $N(\cdot)$. Next, we present two proposals to overcome or, at least, partially overcome this drawback.

Observe that a generalisation operator can be defined via a cost function $k(\cdot, \cdot)$ and a “proper” set of patterns \mathcal{P} . By “proper” we mean a set satisfying both conditions below:

$$\forall p \in \mathcal{P}, \text{Set}(p) \text{ is a connected set.} \quad (1)$$

$$\forall \text{ finite set } E \subset X, \exists p \in \mathcal{P} : E \subset \text{Set}(p) \quad (2)$$

This is expressed in the following definition.

Definition 13 (Generalisation operator induced by a cost function) *Let (X, d) and \mathcal{L} be a ρ -connected metric space and a pattern language, respectively. Given a set of patterns \mathcal{P} satisfying the conditions (1) and (2) and a cost function $k(\cdot, \cdot)$, we will say that $\Delta_{k, \mathcal{P}}$ is a generalisation operator induced by $k(\cdot, \cdot)$ if,*

$$\forall E, \Delta_{k, \mathcal{P}}(E) = \operatorname{argmin}_{p \in \mathcal{P}} k(E, p)$$

The Definition 12 is a particular case of Definition 13 by setting \mathcal{P}

$$\mathcal{P} = \{p \in \mathcal{L} : \exists E \subset X \text{ and a distance-based } \Delta \text{ such that } \Delta(E) = p\}$$

As a result of this we can affirm that,

Proposition 2 *Let \mathcal{P} be the set of patterns above and let \mathcal{P}' be a pattern of sets satisfying the conditions (1) and (2). If $\mathcal{P} \subset \mathcal{P}'$ and $\Delta_{k, \mathcal{P}'}$ is distance-based then $\Delta_{k, \mathcal{P}}$ is a *mg* operator.*

Proof 2 *To do ... !!!!*

The advantage of the Proposition 2 is that in some cases \mathcal{P}' can be easily defined (e.g. $\mathcal{P}' = \mathcal{L}$) and consequently this proposition provides a more operative way of computing *mg* operators. The downside is that $\Delta_{k, \mathcal{P}}$ might not be distance-based. Then, the solution is to relax the optimisation problem. As we mentioned, sometimes it is convenient first to fix $N(\cdot)$ and then to define an appropriate distance-based Δ which fits $N(\cdot)$. In this case, we might not be interested in computing the global *mg* operator but the *mg* operator relative to one specific nerve function. This optimisation problem turns out easier because for some $N(\cdot)$ the set $\{\Delta_{N(\cdot)}\}$ can be explicitly defined.

Definition 14 (mg operator relative to one $N(\cdot)$) *Let (X, d) be a ρ -connected metric space and let Δ be a distance-based generalisation operator defined in X using the pattern language \mathcal{L} . Given a finite set of elements $E \subset X$, a cost function $k(\cdot, \cdot)$ and a nerve function $N(\cdot)$, we will say that Δ_N is a *mg* operator for $k(\cdot, \cdot)$ in \mathcal{L} relative to $N(\cdot)$, if for every distance-based generalisation operator Δ'_N ,*

$$k(E, \Delta_{N(E)}(E)) \leq k(E, \Delta'_{N(E)}(E)), \text{ for every finite set } E \in X.$$

Next, we will illustrate all the main concepts which have been introduced in Sections ?? and ?? in order to compute mg operators for the concrete case of first-order logic data (atoms and clauses) embedded in a metric space. We will see that the Plotkin's lgg is a particular case of this setting because the classical lgg for atoms is a mg operator considering the cost function $k(\cdot, \cdot)$. Furthermore, a different mg operator for atoms will be obtained by changing the cost function. As for clauses, the Plotkin's lgg is not a mg operator for a particular metric space. Due to the complexity of this space, a mg operator relative to one specific nerve function will be defined.

5 Minimal distance-based generalisations for atoms

The goal of this section is to compute mg operators for atoms embedded in a particular metric space. To this end a distance function, a pattern language and a cost function are defined.

5.1 The metric space

The distance function d we are going to employ is defined in [13]. Basically, this distance returns an ordered pair of integer values (i, j) . This pair expresses how different two atoms are in terms of function symbols and of variable symbols respectively. An auxiliary function, the so-called $size(e) = (F, V)$, is required to compute d . This function encodes the structure of one atom e . That is, F is a function counting the number of function symbols occurring in e and V returns the sum of the squared number of occurrences of each variable in e . Finally, given the atoms e_1 and e_2 , $d(e_1, e_2) = [size(e_1) - size(lgg(e_1, e_2))] + [size(e_2) - size(lgg(e_1, e_2))]$

For instance, if $e_1 = q(a, f(a))$ and $e_2 = q(b, f(X))$ and knowing that $lgg(e_1, e_2) = q(Y, f(Z))$, $size(e_1) = (3, 0)$, and $size(e_2) = (2, 1)$, $size(lgg(e_1, e_2)) = (1, 2)$, the distance between e_1 and e_2 is given by the expression: $d(e_1, e_2) = [(3, 0) - (1, 2)] + [(2, 1) - (1, 2)] = (2, -2) + (1, -1) = (3, -3)$

For non-unifiable atoms, the distance is defined by means of introducing an artificial second-order symbol \top , which is considered the most general ⁸ element, such that $size(\top) = (0, 1)$. Note that a total order relation (lexicographic order), defined over the set of ordered pairs, is needed to express how far two atoms are. Given two ordered pairs $A = (F_1, V_1)$ and $B = (F_2, V_2)$, $A < B$ iff $F_1 < F_2$ or $F_1 = F_2$ and $V_1 < V_2$. As the set of tuples are ordered, it permits us to handle these objects as they were real numbers. For this reason, all the definitions of our framework can be automatically extended for this special case.

In what follows, (X_a, d_a) denotes the metric space where X_a is the Herbrand Base with variables induced by the signature and d_a the distance described above.

5.2 The pattern language and the cost function

The pattern language will be the Herbrand base with variables induced by the signature. For example, let $\mathcal{C} = \{a, b\}$ be a set of constants, $\mathcal{F} = \{f/1\}$ a set of function symbols, $\mathcal{X} = \{X_1, X_2, \dots\}$ a numerable set of variables and $\Pi = \{p/1, q/1\}$ a set of predicate symbols. Then, $\mathcal{L}_1 = \{p(a, X_1), p(X_1, a), p(X_1, X_2), p(f(a), b), \dots, q(a, X_1), q(X_1, a), \dots\}$,

Given a pattern p , $Set(p)$ denotes all the atoms in X_a which are logical consequence of p . For example, $p(a) \in Set(p(X))$.

⁸By general we mean the well-known concept from logic programming.

Regarding the cost function, this is defined as $k(E, p) = c(p) + c(E|p)$ where $c(p) = (\text{constant}, 0)$ and $c(E, p)$ is the first function in Table 2. Clearly, it is a cost function for (X_a, d_a) and \mathcal{L} since for a finite set of elements $E \subset X_a$ and a pattern p covering E , $k(E, p) = \infty$ iff $\text{Set}(p) = X_a$.

5.3 Computing mg operators

We proved in [2] that the lgg for two atoms is a binary distance-based generalisation operator for (X_a, d_a) . Taking this previous result into account, we can demonstrate that $lgg(E)$, where E is a finite set of two or more atoms, is the mg for this metric space and this cost function (see Proposition 8 in the Appendix).

Observe that this result does not necessarily hold when the cost function or even the pattern language is changed. For instance, if we set $k_2(E, p) = c_2(E) + c(E|p)$ where $c_2(p) = (0, v)$ denoting v the number of different variable symbols in p and $c(E|p)$ is the same one we have used above, then $lgg(E)$ is not a mg operator. In [4], we explore the combination of different cost functions and pattern languages in further detail.

Example 8 *Given the atoms $e_1 = p(f(a, b), a)$ and $e_2 = p(f(b, a), a)$, we have that $p = lgg(e_1, e_2) = p(f(X, Y), a)$. However, the pattern $p' = p(X, a)$ covers E as well, but $k_2(\{e_1, e_2\}, p') < k_2(\{e_1, e_2\}, p)$. That is, clearly $c(p) = (0, 2)$ and $c(p') = (0, 1)$. Now, let us consider the atoms $e'_1 = p(f(a, b), X)$ and $e'_2 = p(f(b, a), X)$. Neither e'_1 nor e'_2 are covered by p , and it occurs the same for p' . Additionally, we have that $d_a(e_1, e'_1) = (1, -1)$ and $d_a(e_2, e'_2) = (1, -1)$, but as for any two elements x and y in X_a $d_a(x, y) \geq (1, -1)$, we can ensure that $c(E|p) = 2 \cdot (1, -1) = c(E|p')$. Therefore, $k_2(E, p') = (0, 1) + 2 \cdot (1, -1)$ is less than $k_2(E, p) = (0, 2) + 2(1, -1)$. This cost function guides an mg operator which is obtained by removing some (sub-)terms t occurring in the Plotkin's $lgg(\cdot, \cdot)$ such that t contains different variables not occurring in other (sub)-terms.*

The explanation behind this latter example is as follows. In principle, in order to compute the pattern minimising $k_2(E, h)$, we should exhaustively explore the search space \mathcal{L}_{1_E} . However, it might be impracticable from a computational point of view because of the size of \mathcal{L}_{1_E} . It is easy to see that the size of \mathcal{L}_{1_E} grows in a factorial way w.r.t. the number of non-variable symbols at $lgg(E)$. Therefore, an heuristic search becomes fundamental. Note that, not all the predicates in \mathcal{L}_{1_E} performs better than $lgg(E)$, for instance, for the most general pattern $p(X_1, \dots, X_n)$, we have that $k_2(E, p(X_1, \dots, X_n)) = \infty$. The following reflection will lead us to sketch a proper heuristic search.

Imagine a pattern $h \in \mathcal{L}_{1_E}$ where $h \neq lgg(E)$. We know that $\text{Set}(lgg(E)) \subset \text{Set}(h)$ and then $c(E|h) \geq c(E|lgg(E))$. If the inequality $k_2(E, h) < k_2(E, lgg(E))$ holds, it necessary implies that $c(E, h) = c(E|lgg(E))$. Then, the inequality is motivated by the fact of h having less different variables than $lgg(E)$. That is, the pattern h is obtained from the pattern $lgg(E)$ by changing some of the (sub-)terms at $lgg(E)$ containing two or more different variables by a new variable symbol (see example below).

Example 9 *Let $E = \{e_1 = p(f(a, b), g(r(t(a), t(b)), a)), e_2 = p(f(b, a), g(r(t(b), t(a)), b))\}$ ($p/2, , f/2, g/3, r/2, t/1$) be a set of ground atoms. We know that $lgg(E) = p(f(X_1, X_2), g(r(t(X_3), t(X_4)), X_5))$ and that $mg(E)_{k_2, \mathcal{L}_1}$ belongs to the search space \mathcal{L}_{1_E} . In this case, \mathcal{L}_{1_E} would be as follows:*

$$H = \{lgg(E), p(Y_1, g(r(t(X_3), t(X_4)), X_5)), p(f(X_1, X_2), Y_1, X_5), \dots\}$$

However, as we said, not all the patterns in \mathcal{L}_{1E} are good candidates to be the minimal generalisation. For instance, for the pattern $h = p(f(X_1, X_2), g(r(Y_1, t(X_4)), X_5))$, $k_2(E, h)$ is not minimal since h and $\text{lgg}(E)$ has the same number of different variables. Attending to this criterion, the candidates could be the patterns collected in Table 3:

Pattern	$c_2(E)$	$c(E h)$	$k_2(E, h)$
$h_1 = p(Y_1, g(r(t(X_3), t(X_4)), X_5))$	(0, 4)	$2 \cdot (2, -1)$	$(0, 4) + 2 \cdot (2, -1) = (4, 2)$
$h_2 = p(f(X_1, X_2), g(Y_1, Y_2))$	(0, 4)	$2 \cdot (3, -1)$	$(0, 4) + 2 \cdot (3, -1) = (6, 2)$
$h_3 = p(f(X_1, X_2), Y_1)$	(0, 3)	$2 \cdot (3, -1)$	$(0, 3) + 2 \cdot (3, -1) = (6, 1)$

Table 3: Patterns candidates to minimise the cost function $k_2(E, h)$.

The minimal generalisation will be among the candidates atoms. In this example, the minimal generalisation is the pattern h_1 .

Observe that the heuristic only informs, for a given pattern h , about what (sub-)term at h are candidates to be substituted in order to obtain a better pattern (equivalently, less general pattern), not necessarily the minimal. Obtaining the minimal generalisation may require to repeat this process several times. Thus, the inverse substitution σ^{-1} leading to a minimal generalisation will be built incrementally.

The algorithm is founded on the following two propositions. Roughly speaking, first proposition says that those (sub-)terms at $\text{lgg}(E)$ containing non-different variables can be ignored by the inverse substitution. The second proposition deals with registering the different inverse substitutions since in case equal (sub-)terms at $\text{lgg}(E)$ are substituted, they must be changed by the same variable symbol.

Proposition 3 *Let E be a finite set of first-order ground atoms and let $\sigma^{-1} = \{o_1/X_1, \dots, o_n/X_n\}$ be an inverse substitution such that $h = \text{lgg}(E)\sigma^{-1}$ is a minimal generalisation of E . For every pair of occurrences o_i and o_j involved in σ^{-1} such that $o_i \neq o_j$, if $\text{lgg}(E)|_{o_i} = \text{lgg}(E)|_{o_j}$ then $X_i = X_j$.*

Proof 3 *We will proceed by contradiction. For the sake of simplicity, we will assume that there only exists two occurrences, o_i and o_j , satisfying the condition specified by the proposition. However, the reasoning would be the same in case there have been more than two occurrences.*

Let us suppose that both variables X_i and X_j are different. Next, we define a new inverse substitution from σ^{-1} , namely σ'^{-1} , by forcing the variables X_i and X_j to be equal. Then, $h' = \text{lgg}(E)\sigma'^{-1}$ is a new pattern covering E since we know that for every predicate p in E , $p|_{o_1} = p|_{o_2}$. Now, we can say that $\text{Set}(h)' \subset \text{Set}(h)$ and $c(E|h') \leq c(E|h)$. As h' contains less different variables than h because $X_i = X_j$, we have that $c_2(h') < c_2(h)$, and consequently, $k_2(E, h') < k_2(E, h)$. But it is not possible since h is a minimal generalisation. Therefore, the variables X_i and X_j are equal.

Proposition 4 *Let E be a finite set of first-order ground atoms and let $\sigma^{-1} = \{o_1/X_1, \dots, o_n/X_n\}$ be an inverse substitution such that $h = \text{lgg}(E)\sigma^{-1}$ is a minimal generalisation. If we define the set $I(h) = \{o_{i_j}\}_{j=1}^{m \leq n}$ as the set of all the occurrences in σ^{-1} such that $\text{lgg}(E)|_{o_{i_j}}$ is a (sub-)term containing non-different variables ($\forall o_{i_j} \in I(h)$), then there exists an inverse substitution $\sigma'^{-1} \subset \sigma^{-1}$ such that $h' = \text{lgg}(E)\sigma'^{-1}$ is a minimal pattern and $I(h') = \emptyset$.*

Proof 4 The idea is to define a new minimal pattern h' from h such that $I(h') = \emptyset$ from h . Let us define the set $\sigma'^{-1} = \{o_{k_1}/X_{k_1}, \dots, o_{k_p}/X_{k_p}\}$ ($p = n - m$) such that for every pair o_{k_i}/X_{k_i} , we have that $o_{k_i}/X_{k_i} \in \sigma^{-1}$ and the occurrence $o_{k_i} \notin I(h)$. Clearly, σ'^{-1} is an inverse substitution for $\text{lgg}(E)$ because $\sigma'^{-1} \subset \sigma^{-1}$. By setting $h' = \text{lgg}(E)\sigma'^{-1}$, we have that $\text{Set}(\text{lgg}(E)) \subset \text{Set}(h') \subset \text{Set}(h)$, then $c(E|h') = c(E|h)$.

Note that the difference between h' and h is that for every $o_{i_j} \in I(h)$, $h'|_{o_{i_j}} = \text{lgg}(E)|_{o_{i_j}} = t_{i_j}$ and $h|_{o_{i_j}} \neq \text{lgg}(E)|_{o_{i_j}}$ being $h|_{o_{i_j}}$ a variable symbol. Hence, we can write that,

$$\text{Var}(h') = (\text{Var}(h) - \{X_{i_j}\}_{j=1}^{m \leq n}) \bigcup_{\forall o_{i_j} \in I(h)} \text{Var}(t_{i_j}),$$

and

$$c_2(h') = c_2(h) - m + r \quad (0 \leq r \leq m),$$

where r is the number of different variables in the set $\bigcup_{\forall o_{i_j} \in I(h)} \text{Var}(t_{i_j})$. Consequently, $c_2(h') \leq c_2(h)$ and $k_2(E, h') \leq k_2(E, h)$. As the pattern h' is a minimal generalisation, then $k_2(h') = k_2(h)$.

Now, we are in conditions of sketching the search algorithm. This algorithm takes three input arguments: a finite set E of ground predicates, a pattern h covering E and a two-column table T mapping (sub)-terms with variables (this table is a direct consequence of Proposition 3). The objective is to obtain a new pattern h' from h by previously computing a proper inverse substitution σ^{-1} . This process will be driven by the cost function $k_2(E, h)$: first, only those (sub)-terms containing different variables will initially be considered, being their respective occurrences stored at list L (Proposition 4). Then, only those terms from L whose substitution leads a new pattern h' such that $c(E|h') = c(E|h)$ will be really explored. The pairs o_i/X_i involved in an inverse substitution will be stored at T as follows: a cell placed on the left column of T will contain the (sub)-term $h|_{o_i}$ and its correspondent right cell the variable X_i . Additionally, two retrieval operations over T will be needed. First, given a (sub)-term t , the expression $t \in T$ will return *true* or *false* depending on whether t is stored at T or not. Secondly, if $t \in T$ is *true*, the expression $T[t]$ will return its correspondent variable in the inverse substitution (see Algorithm 1).

Proposition 5 Let E be a finite set of first-order ground predicates. Then, $\text{searchMinimal}(E, \text{lgg}(E), T = \emptyset) = \text{mg}(E)_{k_2, \mathcal{L}_1}$.

Proof 5 In the first place, we are going to show that there exists a minimal generalisation which is not pruned by the heuristic search. Then, we will prove that a minimal generalisation is computed.

The algorithm only explores those (sub)-terms at $\text{lgg}(E)$ containing two or more than two different variables, since those occurrences at $\text{lgg}(E)$ representing (sub)-terms with non-different variables are not stored at list L . Therefore, according to Proposition 4, not all the minimal generalisations are pruned.

In addition, given an occurrence $o \in \text{Occ}(\text{lgg}(E))$ such that $c(E|\text{lgg}(E)\sigma^{-1}) > c(E|\text{lgg}(E))$ where $\sigma^{-1} = \{o'/X\}$ ($X \notin \text{Var}(\text{lgg}(E))$), the algorithm will never substitute those (sub)-terms at $\text{lgg}(E)$ whose occurrence $o' \in \text{Pre}(o)$ because clearly $c(E|\text{lgg}(E)\sigma'^{-1}) > c(E|\text{lgg}(E)\sigma^{-1}) > c(E|\text{lgg}(E))$ where $\sigma'^{-1} = \{o'/X\}$. Therefore, by doing that, no minimal generalisations are pruned.

Finally, those (sub)-terms t_1 and t_2 to be changed such that $t_1 = t_2$ are substituted by the same variable symbol. then both variables must be equal in order to achieve a minimal generalisation

```

Data:  $E$  (a finite set of ground predicates),  $h$  (predicate covering  $E$ ) and  $T$  (a two-column
table indexed by (sub-)terms)
Result:  $mg(E)_{k_2, \mathcal{L}_1}$ 
 $searchMinimal(E, h, T)$ 
begin
   $S \leftarrow \{h\}$  //  $S$  is the set of patterns to be considered;
   $L \leftarrow \{o \in Occ(h) : h|_o \text{ does not have (sub-)terms with two or more different variables}\};$ 
   $(c, v) \leftarrow c(E|h);$ 
  for  $1 \leq i \leq length(L)$  do
    if  $h|_{L[i]} \in T$  then  $h' \leftarrow h[T[h|_{L[i]}]/L[i]];$ 
    else  $h' \leftarrow h[X/L[i]]$ , such that  $X \notin Var(h);$ 
    if  $c(E|h') = (c, v)$  then
       $T' \leftarrow T$  // copy table  $T$  in  $T'$ ;
      if  $h|_{L[i]} \notin T$  then  $T'[h|_{L[i]}] \leftarrow X;$ 
       $S \leftarrow S \cup \{searchMinimal(E, h', T')\};$ 
    end
  end
   $mg(E)_{k_2, \mathcal{L}_1} \leftarrow \min\{c_2(s) : \forall s \in S\};$ 
  return  $mg(E)_{k_2, \mathcal{L}_1};$ 
end

```

Algorithm 1: A pattern extraction algorithm for first-order ground predicates based on an heuristic search.

(Proposition 3). This is done by storing in table T all the changed (sub-)terms and consulting T before substituting another (sub-)term.

Now, we must prove that $searchMinimal(E, lgg(E), \emptyset)$ returns a minimal generalisation of E . From what we said above, we can affirm that the set S contains at least a minimal generalisation of E when the loop finishes. Furthermore, we can ensure that the set S contains a finite set of patterns covering E such that $c(E|h) = c(E|lgg(E))$ for every pattern s in S . Hence, a pattern $s \in S$ will be a minimal generalisation of E , if $c_2(s) \leq c_2(s')$ for every $s' \in S$.

Next, an example is given in order to illustrate the algorithm.

Example 10 Let us consider the set $E = \{p(f(a), g(a, b, c), g(a, b, c), a), p(f(b), g(a, c, b), g(a, c, b), a), p(f(b), g(c, b, a), g(c, b, a), a)\}$ ($p/3, f/1, g/3$) of ground predicates. Clearly, $lgg(E) = p(f(X), g(Y, Z, T), g(Y, Z, T)a)$. We are going to look inside the different recursive calls generated by $searchMinimal(E, lgg(E), \emptyset)$.

- (**call 1**) $searchMinimal(E, lgg(E), \emptyset)$. This is the initial call. The variables, before reaching the loop, take the values:

$$S = \{lgg(E)\}, L = \{2, 3\} \text{ and } (c, v) = (1, -1)$$

In what follows, both variables, patterns and inverse substitutions will be indexed depending on the occurrence at L to be treated. Therefore, according to the occurrences at L , the inverse

substitutions to be considered are $\sigma_2^{-1} = \{2/X_1\}$ and $\sigma_3^{-1} = \{3/X_1\}$. They leads to the patterns $h_2 = \text{lgg}(E)\sigma_2^{-1} = p(f(X), X_1, g(Y, Z, T), a)$ with $T_2 = \sigma_1^{-1}$ and $h_3 = \text{lgg}(E)\sigma_3^{-1} = p(f(X), g(Y, Z, T), X_2, a)$ with $T_3 = \sigma_3^{-1}$, respectively.

As $c(E|h_2) = c(E|\text{lgg}(E))$ and $c(E|h_3) = c(E|\text{lgg}(E))$, two new calls ($\text{searchMinimal}(E, h_2, T_2)$ and $\text{searchMinimal}(E, h_3, T_3)$) will be generated.

- **(call 2)** $\text{searchMinimal}(E, h_2, T_2)$: The variables for this call, before reaching the loop, are instantiated as follows:

$$T_2 = \{(g(Y, Z, T), X_1)\}$$

$$S = \{h'_1\}, L = \{3\} \text{ and } (c, v) = (1, -1)$$

At this call, the table (T_2) is not empty and before substituting the occurrence 3 by a variable symbol, we must check if the (sub-)term $h_2|_3$ is stored at T_2 . As we can see, it is, being the resulting inverse substitution $\sigma_3^{-1} = \{3/X_1\}$ and the new pattern $h_3 = p(f(X), X_1, X_1, a)$.

Although another call ($\text{searchMinimal}(E, h_3, T_2)$) would be generated, it will not be unfolded. Note that h_3 has not any (sub-)term with different variables, then this call simply returns h_3 as a possible minimal generalisation. In its turn, the current call (call 2) returns will return h_3 to call 1.

- **(call 3)** The process is the same that the one in call 2. Repeating all the steps, we have that the pattern $p(f(X), X_1, X_1, a)$ is computed and returned to call 1.
- **(call 1)** Going back at this point, we have that after the loop,

$$S = \{\text{lgg}(E), p(f(X), X_1, X_1, a)\},$$

therefore, $\text{mg}(E)_{k_2, \mathcal{L}_1} = p(f(X), X_1, X_1, a)$.

Following with this example, by computing all the possible inverse substitutions (exhaustive search), 16 patterns would have been tried. However, using this heuristic, only 2 trials have been required.

Of course, some immediate algorithmic considerations could be taken into account in order to improve the computational efficiency of the method. Next, some interesting remarks in this line are treated.

(Remark 1) In general, the function $c(E|h)$ has a computational cost given by $|E| \cdot N$ where $|E|$ is the size of the set E and N is the number of symbols of the largest pattern in E . However, some specific useful cases can be detected. For instance, if h has two identical variables or constant symbols, then automatically $c(E|h) = |E| \cdot (1, -1)$.

(Remark 2) Note that the algorithm can yield the same pattern h by means of different recursive calls. In order to avoid computing $c(E|h)$ several times, the pair $(h, c(E|h))$ could be memorised in a hash table.

6 Minimal distance-based generalisations for clauses

From a practical point of view, finite sets of literals (interpreted as clauses) allow us to express real-world objects more accurately than single literals do. A set of literals can represent not only

the different parts of an object but also the relationships among them. A clause interprets these literals as a disjunction and it is usually expressed in logic programming, with all the positive literals in the head as a disjunction and all the negative literals in the body as a conjunction. For instance $C = \{class(X, c_1), \neg molec(x), \neg atom(X, h)\}$ can be interpreted and represented as: $class(X, c_1) : \neg molec(x) \wedge atom(X, h)$.

In order to determine *mg* operators for clauses, we need to establish a distance function, a pattern language and a cost function for this sort of data.

6.1 The metric space

The distance we are going to use is defined in [12]. Roughly speaking, this distance is based on minimal matchings⁹ over sets and requires the elements of the sets to be embedded in a metric space as well. Given two sets A and B and the elements $a \in A$ and $b \in B$, we will say that the ordered pair (a, b) belongs to the matching $\alpha_{A,B}$ between A and B (a subset of $A \times B$), if $\alpha_{A,B}(a) = b$. By $D(\alpha_{A,B})$, we denote the domain of the matching, that is, $D(\alpha_{A,B}) = \{a \in A : \exists (a, b) \in \alpha_{A,B}\}$, and by $\alpha_{A,B}(A) = \{b \in B | (a, b) \in \alpha_{A,B} \wedge a \in A\}$ we denote the codomain of the matching.

Thus, given two sets A and B and a matching $\alpha_{A,B}$, a similarity measure $d(\alpha_{A,B}, A, B)$ can be defined by summing the distances between the elements from the ordered pairs belonging to $\alpha_{A,B}$ and adding a penalty $M/2$ for each element in A and B not included in the matching. More formally,

$$d(\alpha_{A,B}, A, B) = \sum_{\forall (a_i, b_j) \in \alpha_{A,B}} d(a_i, b_j) + \frac{M}{2} (|B - \alpha_{A,B}(A)| + |A - D(\alpha_{A,B})|) \quad (3)$$

Finally, the distance between A and B is given by the optimal (minimal) matching among all the possible ones.

$$d_m(A, B) = \min_{\forall \alpha_{A,B}} d(\alpha_{A,B}, A, B) \quad (4)$$

Unless we say otherwise, $(2^X, d_m)$ denotes the metric space of sets and (X, d) the metric space of the elements of the sets. According to [12], the constant M must be greater or equal to the maximal distance between two elements in X in order to have a $d_m(\cdot, \cdot)$ which satisfies all the axioms of a metric. This restriction forces us to bound (restrict) the space X . In our case (X, d) will be (X_l, d_l) , same as previous section but extending the space and distances to both positive and negative atoms, i.e. literals. This extension is trivial, since $p(\cdot, \cdot)$ and $\neg p(\cdot, \cdot)$ are considered incompatible literals, and, hence it is like treating them as different predicates. Then the restriction will consist of setting a threshold for the number of symbols in an atom, namely $R/2$. Only atoms with less than $R/2$ symbols will be permitted. Thus, let \bar{X}_l and $M = (R, R)$ be the bounded space and the penalty respectively¹⁰.

Example 11 Given the sets $A = \{a_1 = p(g(a), e), a_2 = p(f(a), f(b)), a_3 = p(a, a)\}$ and $B = \{b_1 = p(f(b), f(a)), b_2 = p(f(a), e)\}$ and according to the distances among all the atoms, the optimal matching $\alpha_{A,B}$ is $\alpha_{A,B} = \{(a_1, b_2), (a_2, b_1)\}$. Then, the distance between the sets is given by $d_m(A, B) = d(a_1, b_2) + d(a_2, b_1) + \frac{1}{2}(R, R) = (8, -6) + (\frac{R}{2}, \frac{R}{2})$.

⁹A matching from the set A to the set B is an injective mapping not necessarily defined over all the elements in A .

¹⁰It is finally neither a real nor theoretical problem. A representation of a real-life object requires always a finite number of symbols and all the results concerning (X_l, d_l) holds for (\bar{X}_l, d_l) as well.

6.2 The pattern language and the cost function

Thus we define \mathcal{L} as the set of all the logic programs we can define given a signature (see Section 2). Some examples of patterns could be,

$$\begin{aligned} p_1 &\equiv \text{class}(X, c_1) : \neg \text{molec}(X), \text{atom}(X, Y, h). \\ &\quad \text{class}(X, c_1) : \neg \text{molec}(X), \text{atom}(X, Y, o) \\ p_2 &\equiv \text{class}(X, c_2) : \neg \text{molec}(Y), \text{atom}(Y, Z, c) \end{aligned}$$

The pattern p_1 says that a molecule belongs to the class/cluster c_1 if it has an atom of hydrogen or oxygen. Of course, a pattern can also be viewed as a set of clauses. For example, $p_1 = \{C_{11} = \{\text{class}(X, c_1), \neg \text{molec}(X), \neg \text{atom}(X, h)\}, C_{12} = \{\text{class}(X, c_1), \neg \text{molec}(X), \neg \text{atom}(X, o)\}\}$ and $p_2 = \{C_{21} = \{\text{class}(X, c_2), \neg \text{molec}(Y), \neg \text{atom}(Y, c)\}\}$. From this point of view, patterns can be combined by means of the union operator (\cup). Thus, the pattern $p_3 = p_1 \cup p_2$ is $p_3 = \{C_{11}, C_{12}, C_{21}\}$. Moreover, each clause C in p is a pattern as well, which is denoted as $\{C\}$.

Finally, given a pattern $p \in \mathcal{L}$, $\text{Set}(p)$ represents all those clauses in the metric space $2^{\bar{X}^i}$ which are a logical consequence of p . Therefore, the clause $\{\text{class}(m_1, c_1), \neg \text{molec}(m_1), \neg \text{atom}(m_1, h)\}$ belongs to $\text{Set}(p_1)$.

As for the cost function, this is defined as $k(E, p) = c(p) + c(E|p)$ where $c(p) = (\text{constant}, 0)$ and $c(E, p)$ is the first function in Table 2. Clearly, it is a cost function for $(2^{\bar{X}^i}, d_m)$ and \mathcal{L} since for a finite set of elements E and a pattern p covering E , $k(E, p) = \infty$ iff $\text{Set}(p) = 2^{\bar{X}^i}$.

6.3 Computing mg operators

Unlike (X_a, d_a) , let us first see that $\Delta(E) = \text{lgg}(E)$ (where lgg is the *least general generalisation* for clauses [11]) is not a mg operator in this metric space. Although it can easily be shown that $\text{lgg}(E)$ is a minimal pattern in our framework, $\text{lgg}(E)$ is not distance-based for $d_m(\cdot, \cdot)$. That is, given two clauses A and B there exists a clause C such that $d(A, C) + d(C, B) = d(A, B)$ and C is not covered by $\text{lgg}(A, B)$ (see Example 12).

Example 12 Given the sets $A = \{\neg p(g(a), e), \neg p(f(a), f(b))\}$, $B = \{\neg p(f(b), f(a)), \neg p(f(a), e)\}$ and $C = \{\neg p(f(b), f(b)), \neg p(g(a), e)\}$. The optimal mappings from A to C and from C to B , respectively, are depicted below.

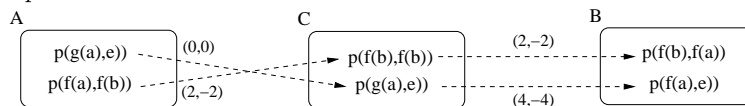


Figure 16: The arrows indicate the optimal mappings between the different pair of sets.

We can easily see that $d_m(A, B) = (8, -8) = d_m(A, C) + d_m(C, B)$. However,

$$\begin{aligned} \text{lgg}(A, B) &= \{\neg p(f(X), f(Y)), \neg p(Z, e), \neg p(f(a), T)\} \equiv \\ &: \neg p(f(X), f(Y)), p(Z, e), p(f(a), T), \end{aligned}$$

but $C \equiv: \neg p(f(b), f(b)), p(g(a), e)$ is not a logical consequence of $\text{lgg}(A, B)$.

Unfortunately, computing mg operators in this space will not be as intuitive as was in the previous section. We tackle the problem in a different way. First, we focus on determining binary mg operators. And then we study if combining these binary mg operators, we can obtain n -ary mg operators. Proposition 11 (see Appendix) along with the fact that $lgg(\cdot)$ for atoms is distance-based, provides a strategy to define distance-based binary generalisation operators: given two clauses A and B in $2^{\bar{X}_l}$, we could initially define $\Delta(A, B) = \{\{lgg(a_i, b_j) : (a_i, b_j) \in \alpha_{A, B}\}\}$ where $\alpha_{A, B}$ is an optimal mapping. A distance-based operator must compute a pattern covering the elements C between A and B . Proposition 11 (see appendix) states that if C is between A and B , then C contains atoms c_k which are also between a_i and b_j , for every (a_i, b_j) in an optimal $\alpha_{A, B}$. But as lgg for atoms is distance-based for the distance d_l , if c_k is between the atoms a_i and b_j , then $c_k \in Set(lgg(a_i, b_j))$. At first sight, this definition of Δ seems to be distance-based. However, two drawbacks must be analysed.

1. Variables occurring in the different $lgg(a_i, b_j)$ must be independent (i.e. never repeated). Otherwise the correspondent pattern might not be distance-based (for further details see [4]). We will tackle this crucial issue at the end of this section.
2. More than one optimal matching can be given for $d_m(A, B)$. Of course, if the matchings lead to different patterns p_1 and p_2 such as $Set(p_1) \neq Set(p_2)$, there will be elements between A and B which will not belong to $Set(p_1)$ but also $Set(p_2)$ and vice-versa. Hence, all the optimal matchings must be taken into account. Of course, this has a negative effect on the efficiency of computing distance-based operators.

Taking both observations above into account, Proposition 13 (see Appendix) characterises the family of all the distance-based binary generalisation operators. Roughly speaking, this proposition shows that a binary generalisation operator $\Delta(A, B)$ is distance-based if $Set(\Delta^*(A, B)) \subset Set(\Delta(A, B))$, where $\Delta^*(A, B)$ represents the union of all patterns p_i obtained by taking all the optimal matchings between A and B into account.

Now, we must determine the mg binary operator. It is direct from Propositions 13 and 1, since according them for every distance-based operator $\Delta(A, B)$ and for every pair of elements A and B , we know that $Set(\Delta^*(A, B) = p) \subset Set(\Delta(A, B))$ and then $k(\{A, B\}, p) \leq K(\{A, B\}, \Delta(A, B))$. Then, $\Delta^*(A, B)$ is the mg .

Given that the patterns p_i can be combined by means of the union operator, a distance-based operator can be defined for more than two elements by defining a distance-based binary operator, namely Δ' , and fixing a nerve-function $N(\cdot)$. Given the set $E = \{e_1, \dots, e_n\}$, $\Delta_{N(E)}(E) = \bigcup_{\forall (e_i, e_j) \in N(E)} \Delta'(e_i, e_j)$.

The distance-based operator which is minimal can be determined by exploring all the possible nerves $N(E)$ for a set of elements E . On the other hand, we may only be interested in computing the mg relative to a specific nerve function. Then, Proposition 14 (see Appendix) states that if $\Delta' = \Delta^*$ in the expression above, then $\Delta_{N(E)}(E)$ is a mg relative to a nerve function $N(E)$. That is, it is the minimal distance-based operator in $\mathcal{F}_{N(\cdot)}$.

Before concluding, note that a pattern computed by a mg can not contain repeated variables in the different $lgg(a_i, b_j)$ from a same clause. On the one hand, it makes sense since the metric does not capture the semantic of repeated variables occurring in different atoms. Hence, for the sets $A = \{p(a), q(a)\}$, $B = \{p(b), q(b)\}$, $C = \{p(c), q(d)\}$, we have that $d_l(A, B) = d_l(A, C)$ when intuitively B should be more similar to A . This is a strong constraint in order to express some real-world properties. But this concerns only the mg operator. That is, it does not mean that distance-

based operators cannot contain repeated variables in different atoms in general. For instance, the Proposition 13 suggests that we could adapt a bottom-up ILP inference algorithm to take as input the pattern (clauses) computed by the mg . The output of the algorithm will be a pattern more general than the input pattern, will be distance-based as well and perfectly can contain repeated variables among different atoms. Furthermore, this fact stands out that some adaptations of the ILP algorithms can be viewed as distance-based operators.

7 Conclusions and future work

This work introduces the notion of mg operator for every sort of data which is embedded in a metric space. Here we present a detailed definition of the framework, including the notion of generality in terms of the MDL/MML principle, and we define several minimal generalisation operators for different sorts of data. Additionally, we prove that the lgg is a particular case of mg operator for atoms embedded in the metric space defined in [13]. ***** A expandir: (i) Dir q. altres operadors “lgg” poden obtenir per atom quan es canvia la funcio $k()$ (ii) Dir q. hem estudiat el cas del espai metric de clausules (iii) Dir q. el marc en general esta definit sobre altre tipus de dades (nominals, numerics, grafs). (iv) Podriem comentar el tema de la experimentacio els resultats del Δ serveixquen com a entrada a un sistema bottom-up de ILP i estudiar altres funcions $k()$.

References

- [1] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(9):689–694, 1997.
- [2] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Distance based generalisation. In *Proc. of the 15th International Conference on Inductive Logic Programming, ILP*, volume 3625 of *LNCS*, pages 87–102. Springer, 2005.
- [3] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Distance based generalisation for graphs. In *Proc. of the WS of Machine and Learning with Graphs, MLG06*, 2006.
- [4] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. On the relationship between distance and generalisation. Technical report, DSIC, UPV, <http://www.dsic.upv.es/~flip/#Papers>, 2006.
- [5] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Some results about generalisations of graphs embedded in metric spaces. Technical report, DSIC, UPV, 2006.
- [6] P.A. Flach and N. Lachiche. Naive bayesian classification of structured data. *Machine Learning Journal*, 57:233–269, 2004.
- [7] T. Gaertner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58, 2003.
- [8] T. Gaertner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.
- [9] S. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.

- [10] S. H. Muggleton. Inductive logic programming: Issues, results, and the challenge of learning language in logic. *Artificial Intelligence*, 114(1–2):283–296, 1999.
- [11] G. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.
- [12] J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In *In Proceed. of International Conference on Inductive Logic Programming, ILP*, volume 1446 of *LNCS*, pages 271–280, 1998.
- [13] J. Ramon, M. Bruynooghe, and W. Van Laer. Distance measures between atoms. In *ComputingNet Area Meeting on Computational Logic and Machine Learning*, pages 35–41. University of Manchester, UK, 1998.
- [14] C. S. Wallace and D. L. Dowe. Minimum message length and Kolmogorov complexity. *The Computer Journal*, 42(4):270–283, 1999.
- [15] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks.*, 16(3), 2005.

8 Appendix

In this section we include some propositions required to better understand the paper.

Proposition 6 *Let (X, d) be a ρ -connected metric space. Given two connected sets A, B in X such that $A \cap B \neq \emptyset$, then $A \cup B$ is a connected set.*

Proof 6 *If $z \in A \cap B$, then for every $x, y \in A \cup B$, they are path connected via the element z .*

Proposition 7 *The metric space (X_a, d_a) is $(2, 0)$ -connected.*

Proof 7 *Let $P = \{x_i\}_{i=1}^n$ be a finite set of elements in X_a such that x_{i+1} is obtained from x_i by either change a repeated variable symbol by a new one or changing the deepest non-variable term t occurring at x_i by a variable symbol (not necessarily different). In both cases $d_a(x_i, x_{i+1}) < (2, 0)$, and clearly P is a $(2, 0)$ -path in the space. These paths can be defined when x_n is more general than x_1 , because x_n can be obtained from x_1 by replacing terms by variable symbols.*

Now, let e_i and e_j two elements in X_a . If $\text{lgg}(e_i, e_j) \neq \top$, applying the reasoning above, we know that e_i and $\text{lgg}(e_i, e_j)$ (idem e_j and $\text{lgg}(e_i, e_j)$) are $(2, 0)$ -path connected. Concatenating both paths we conclude that e_i and e_j are $(2, 0)$ -path connected.

If $\text{lgg}(e_i, e_j) = \top$, it means that both elements have different predicate symbol, namely p/n and q/m . The elements e_i and $p(X_1, \dots, X_n)$ (idem e_j and $q(X_1, \dots, X_m)$) are $(2, 0)$ -path connected. As $d_a(p(X_1, \dots, X_n), q(X_1, \dots, X_m)) \leq (2, 0)$ both paths can be concatenated and therefore e_i and e_j are $(2, 0)$ -path connected.

Proposition 8 *Given the metric space (X_a, d_a) . If \mathcal{L} is the base of Herbrand with variables of the atoms in the signature, $k(E, p) = c(p) + c(E|p)$ where $c(p)$ is the constant function and $c(E|p) = \sum_{\forall e \in E} r_e$ (being $r_e = \inf_{r \in \mathcal{R}B(e, r_e)} \not\subset \text{Set}(p)$), then $\Delta(E) = \text{lgg}(E)$ is a mg operator for d_a, \mathcal{L} and k .*

Proof 8 *First, let us show that $\text{lgg}(E)$ performs minimal patterns according to the cost function. For every $p \in \mathcal{L}$, we have by definition that $\text{Set}(\text{lgg}(E)) \subset \text{Set}(p)$, and by Proposition 1, $k(E, \text{lgg}(E)) \leq k(E, p)$ for every p .*

Secondly, let us see that $\text{lgg}(E)$ is distance-based. Next, for every two elements $e_i, e_j \in E$, $\text{Set}(\text{lgg}(e_i, e_j)) \subset \text{lgg}(E)$. According to Proposition 6 in [2], $\text{lgg}(e_i, e_j)$ is distance-based in (X_a, d_a) and i.e. $\text{Set}(E)$ contains all the elements between e_i and e_j , for every e_i and e_j . Then, simply defining $N(E)$ as a complete graph, we have that $\text{lgg}(E)$ is distance-based.

Proposition 9 *The Proposition 8 hold for the metric space (\bar{X}_a, d_a) , where $\bar{X}_a = \{x \in X_a : \text{number of non-variable symbols in } x \leq k\}$.*

Proof 9 *Trivially, for every $e_i, e_j \in E \subset \bar{X}_a$, if e_k is an element between e_i and e_j in \bar{X}_a , it is also in between in the space X_a since the distance function is the same and then $e_k \in \text{Set}(\text{lgg}(E))$ and i.e. $\text{lgg}(E)$ is distance-based in \bar{X}_a .*

Proposition 10 *The metric space $(2^{\bar{X}_a}, d_m)$ is $(2, 0)$ -connected.*

Proof 10 Let A and B be two sets in $2^{\bar{X}_a}$. For the sake of simplicity, let us suppose that $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2\}$ and the optimal matching is $\alpha_{A,B} = \{(a_1, b_1)\}$. Since there is no confusion, the subscripts of the matching can be omitted. Note that there is no loss of generality in our assumption because all the possible scenarios between A and B are covered by this concrete case. That is, $|A| \geq |B|$ and there are elements in A and B not included by the optimal matching and others which also are included.

Next, we introduce a new matching α' defined as $\alpha'(a_1) = \alpha(a_1) = b_1$ and $\alpha'(b_2) = a_2$ **no deberia ser a reves de a2 a b2?** and let us see that A and B are $(2,0)$ -path connected in $(2^{\bar{X}_a}, d_m)$. As a_i and b_j are elements in (\bar{X}_a, d) , from Proposition 9 we can affirm that a_1 and b_j are $(2,0)$ -path connected. More concretely, there exist three $(2,0)$ -paths P_1, P_2 and P_3 where

$$\begin{aligned} P_1 &\equiv a_1 \leftrightarrow a_{11} \leftrightarrow \dots \leftrightarrow a_{1k_1} \leftrightarrow b_1 \\ P_2 &\equiv a_2 \leftrightarrow a_{21} \leftrightarrow \dots \leftrightarrow a_{1k_2} \leftrightarrow b_2 \\ P_3 &\equiv a_3 \leftrightarrow a_{31} \leftrightarrow \dots \leftrightarrow a_{3k_3} \leftrightarrow b_3 \end{aligned} \quad (5)$$

Then, the following paths in $2^{\bar{X}_a}$ can be defined:

$$\begin{aligned} P'_1 &\equiv A \leftrightarrow (A - \{a_1\}) \cup \{a_{11}\} \leftrightarrow \dots \leftrightarrow C_1 = (A - a_1) \cup \{b_1\} \\ P'_2 &\equiv C_1 \leftrightarrow (C_1 - \{a_2\}) \cup \{a_{21}\} \leftrightarrow \dots \leftrightarrow C_2 = (C_1 - \{a_2\}) \cup \{b_2\} \\ P'_3 &\equiv C_2 \leftrightarrow (C_2 - \{a_3\}) \cup \{a_{31}\} \leftrightarrow \dots \leftrightarrow C_3 = (C_2 - \{a_3\}) \cup \{b_2\} = B \end{aligned} \quad (6)$$

As $d_m(\cdot, \cdot)$ is defined, for any two consecutive elements in P'_i , its distance is less or equal to $(2,0)$ and clearly P'_i is a $(2,0)$ -path. Finally, concatenating the paths P'_i , we have that $P = P'_1 \cdot P'_2 \cdot P'_3$ is a $(2,0)$ path connecting A and B .

Proposition 11 Let A, B and C three finite sets of elements. If the equality $d_m(A, B) = d_m(A, C) + d_m(C, B)$ holds, then there exists an optimal mapping $\alpha'_{A,B}$ such that for every pair of elements (a_i, b_j) in $\alpha'_{A,B}$ there exists an element $c_k \in C$ satisfying $d_l(a_i, b_j) = d_l(a_i, c_k) + d_l(c_k, b_j)$.

Proof 11 Let $\alpha_{A,C}$, $\alpha_{C,B}$ be the optimal matchings used for the computation of $d_m(A, C)$ and $d_m(C, B)$, respectively. We can write,

$$\begin{aligned} d_m(A, C) &= \sum_{\forall (a_i, c_j) \in \alpha_{A,C}} d_l(a_i, c_j) + \frac{M}{2} \cdot k_{\alpha_{A,C}} \\ d_m(C, B) &= \sum_{\forall (c_i, b_j) \in \alpha_{C,B}} d_l(c_i, b_j) + \frac{M}{2} \cdot k_{\alpha_{C,B}} \end{aligned}$$

where $k_{\alpha_{A,C}}$ (respectively, $k_{\alpha_{C,B}}$) denotes the number of elements which do not belong to $\alpha_{A,C}$ (respectively, $\alpha_{C,B}$).

Next, we define the matching $\alpha'_{A,B}$ as the composition of the mappings $\alpha_{A,C}$ and $\alpha_{C,B}$. That is, $\alpha'_{A,B} : \alpha_{C,B}(\alpha_{A,C}(a))$. Keeping $\alpha'_{A,B}$ in mind, the sum $d_m(A, C) + d_m(C, B)$ can be written as,

$$\begin{aligned} d_m(A, C) + d_m(C, B) &= \sum_{\forall (a_i, b_j) \in \alpha'_{A,B}} d_l(a_i, \alpha_{A,C}(a_i)) + d_l(\alpha'_{A,C}(a_i), b_j) \\ &\quad + \sum_{\forall a_i \in D(\alpha_{A,C}) - D(\alpha'_{A,B})} d_l(a_i, \alpha_{A,C}(a_i)) \\ &\quad + \sum_{\forall c_i \in D(\alpha_{C,B}) - \alpha_{A,C}(A)} d_l(c_i, \alpha_{C,B}(c_i)) \\ &\quad + \frac{M}{2} \cdot k_{\alpha_{A,C}} + \frac{M}{2} \cdot k_{\alpha_{C,B}} \end{aligned} \quad (7)$$

The first term at the right-hand side of 7 groups all the ordered pairs such that the element $c_i \in C$ is shared. The second term concerns those ordered pairs in $\alpha_{A,C}$ not taken into account by the first term. The last terms deal with those unmatched elements.

Next, the following chain of inequalities can be derived.

$$\begin{aligned}
d_m(A, C) + d_m(C, B) &\geq \sum_{\forall (a_i, b_j) \in \alpha'_{A, B}} d_l(a_i, b_j) \\
&\quad + \sum_{\forall a_i \in D(\alpha_{A, C}) - D(\alpha'_{A, B})} d_l(a_i, \alpha_{A, C}(a_i)) \\
&\quad + \sum_{\forall c_i \in D(\alpha_{C, B}) - \alpha_{A, C}(A)} d_l(c_i, \alpha_{C, B}(c_i)) \\
&\quad + \frac{M}{2} \cdot k_{\alpha_{A, C}} + \frac{M}{2} \cdot k_{\alpha_{C, B}} \\
&\geq \sum_{\forall (a_i, b_j) \in \alpha'_{A, B}} d_l(a_i, b_j) \\
&\quad + \frac{M}{2} \cdot (k_{\alpha_{A, C}} + k_{\alpha_{C, B}}) \\
&\geq \sum_{\forall (a_i, b_j) \in \alpha'_{A, B}} d_l(a_i, b_j) + \frac{M}{2} \cdot (k_{\alpha'_{A, B}}) = d(\alpha'_{A, B}, A, B) \\
&\geq d_m(A, B)
\end{aligned} \tag{8}$$

An explanation is as follows. The first inequality is obtained by applying the triangle inequality over the first term in the Expression 7. The second one comes from removing the second and the third terms from the Expression 8. The third inequality results because $k_{\alpha_{A, C}} + k_{\alpha_{C, B}} \geq k_{\alpha'_{A, B}}$. The last one is a direct consequence of the $d_m(\cdot, \cdot)$ definition.

The equality $d(A, C) + d(C, B) = d(A, B)$ holds only if all the inequalities (\geq) in the right-hand-side of the expression (8) become an equality. Among all of these transformations, only the first and the last one are fundamental in order to prove the proposition. Let us analyse when this happens.

The first inequality turns into an equality if the element $c_k = \alpha'_{A, C}(a_i)$ in the first term of the right-hand-side of (7) satisfies $d(a_i, b_j) = d(a_i, c_k) + d(c_k, b_j)$, for every pair (a_i, b_j) in $\alpha'_{A, B}$. The proposition is automatically proved in case $\alpha'_{A, B}$ is an optimal matching. But this necessarily occurs so that the last inequality must be transformed into an equality.

Proposition 12 Given two clauses A and B in $(2^{\bar{X}_l}, d_l)$ and the pattern language \mathcal{L} consisting of all logic programs defined over a signature. The binary generalisation operator $\Delta^*(A, B) = p$ such that

$$p = \bigcup_{\forall \text{ optimal } \alpha_{A, B}} \{\text{lgg}(a_i, b_j) : \forall (a_i, b_j) \in \alpha_{A, B}\},$$

being the repeated variables occurring in different $\text{lgg}(a_i, b_j)$ independent, then Δ^* is distance-based.

Proof 12 From Proposition 11, we know that if a set D is between A and B , then there exists an optimal mapping α such that for every $(a_i, b_j) \in \alpha$ there exists a $d_k \in D$ being d_k between a_i and b_j . As lgg for atoms is distance-based, necessarily $d_k \in \text{Set}(\text{lgg}(a_i, b_j))$ and i.e. $D \in \text{Set}(\{\text{lgg}(a_i, b_j) : \forall (a_i, b_j) \in \alpha\})$. As all the optimal mappings are considered, for every set D between A and B , $D \in \text{Set}(p)$ and i.e., $\Delta^*(A, B)$ is distance-based.

Proposition 13 Given the metric space $(2^{\bar{X}_l}, d_l)$ and the pattern language \mathcal{L} consisting of all the logic programs defined over a signature. A mapping $\Delta : 2^{\bar{X}_l} \times 2^{\bar{X}_l} \rightarrow \mathcal{L}$ is distance-based iff for every clauses A and B $\text{Set}(\Delta^*(A, B)) \subset \text{Set}(\Delta(A, B))$, being Δ^* the distance-based operator defined in Proposition 12.

Proof 13 (\rightarrow) If $\Delta(A, B)$ is distance-based, then it means that for every D between A and B , $D \in \text{Set}(\Delta(A, B))$. Then, for every optimal mapping $\alpha_{A, B}$, we define $D_{\alpha_{A, B}}$ as

$$D_{\alpha_{A, B}} = \{\text{lgg}(a_i, b_j) : \forall (a_i, b_j) \in \alpha_{A, B}\}$$

which is clearly between A and B . Then, we have that, for every optimal mapping $\alpha_{A,B}$, $D_{\alpha_{A,B}} \in \text{Set}(\Delta(A,B))$, i.e. $\Delta^*(A,B) \subset \text{Set}(\Delta(A,B))$ and by definition of $\text{Set}(\cdot)$, $\text{Set}(\Delta^*(A,B)) \subset \text{Set}(\Delta(A,B))$.

(\leftarrow) We have that $\text{Set}(\Delta^*(A,B))$ is a subset of $\text{Set}(\Delta(A,B))$. As $\Delta^*(A,B)$ is distance-based, automatically $\Delta(A,B)$ is distance-based.

Proposition 14 Let $\Delta^*(A,B)$ the binary generalisation operator introduced in the Proposition 12 and let $k(E,p) = c(p) + c(E|p)$ the cost function $k(E,p) = c(p) + c(E|p)$ where $c(p)$ is the constant function and $c(E|p) = \sum_{\forall e \in E} r_e$ (being $r_e = \inf_{r \in \mathcal{R}} B(e, r_e) \notin \text{Set}(p)$), then

$$\Delta_{N(E)}(E) = \bigcup_{\forall (e_i, e_j) \in N(E)} \Delta^*(e_i, e_j)$$

is a mg operator relative to the nerve function $N(E)$.

Proof 14 We will proceed by contradiction. Let us suppose that $\Delta_{N(E)}$ is not mg. Then there exists a distance-based $\Delta'_{N(E)}$ such that $k(E, \Delta'_{N(E)}) \leq k(E, \Delta_{N(E)})$ for every E , and there necessarily exists a set $|E| > 2$ such that $k(E, \Delta'_{N(E)}) < k(E, \Delta_{N(E)})$. We define a binary distance-based operator Δ'' restricted to all the pairs $(e_i, e_j) \in N(E)$, such that $\Delta''(e_i, e_j) = \Delta'_{N(E)}(E) = p$. But according to Proposition 13, we have

$$\text{Set}(\Delta^*_{N(E)}(e_i, e_j)) \subseteq \text{Set}(\Delta''(e_i, e_j)) = \text{Set}(\Delta'_{N(E)}(E))$$

As it happens for every $(e_i, e_j) \in N(E)$, then we have that $\text{Set}(\Delta_{N(E)}(E)) \subseteq \text{Set}(\Delta'_{N(E)}(E))$, and consequently $\Delta'_{N(E)}$ cannot be mg.