# Minimal distance-based generalisation operators for first-order objects [⋆]

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

DSIC, Univ. Politècnica de València , Camí de Vera s/n, 46020 València, Spain.
{vestruch,cferri,jorallo,mramirez}@dsic.upv.es

**Abstract.** Distance-based methods have been 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 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* operators) for first-order data. We show that Plotkin's *lgg* is a *mg* operator for atoms under the distance introduced by J. Ramon, M. Bruynooghe and W. Van Laer. We also show that this is not the case for clauses with the distance introduced by J. Ramon and M. Bruynooghe. 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 [4]). Nevertheless, learning from complex data while preserving comprehensibility is even more challenging and has mainly been addressed in the area of ILP [7]. Despite the fact that distance-based methods are quite intuitive and have successfully been tested in several domains, a model that explains why a new example belongs to one class or another does not exist. This is due to the fact that the information about the matches between two objects (e.g. two molecules) is lost when these matches are encoded by a number (their distance). Unfortunately, this lack of explanatory patterns is incompatible for many application contexts.

[⋆] This work has been partially supported by the EU (FEDER) and the Spanish MEC under grant TIN 2004-7943-C04-02, ICT for EU-India Cross-Cultural Dissemination Project under grant ALA/95/23/2003/077-054, Generalitat Valenciana under grant GV06/301 and UPV under grant TAMAT.

For example, in molecule classification it would be very interesting 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 descriptions of this kind for distance-based algorithms in [2], where the concept of distance-based binary generalisation operator was introduced. Basically, the term 'distance-based' means that the operator computes patterns that are "consistent" with the distance employed. For instance, let $(\Sigma^*, d)$ be the word space defined over the alphabet $\Sigma = \{a, b, c\}$, and let $d$ be 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, to use these generalisation operators in a real context, we need to be able to generalise more than two elements. In [1], we introduced this idea for $n$-ary operators and we also studied the idea of minimality. Minimality is important to prevent underfitting in the search of patterns that 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 studied in depth when data is represented by means of first-order logic [11], the same does not happen for other kinds of data, and especially when data is in a metric space. Therefore, in [1] we proposed a general way to define minimal distance-based generalisation operators ($mg$ operators). We have applied this framework to several data sorts: sets, lists, graphs,... (see [3, 1]).

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 data which has been clustered employing this distance. Then, we try to extend this result to Horn clauses (more precisely, sets of literals), 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 in ILP (such as the one introduced in this paper).

The paper is organised as follows. Section 2 introduces the framework for distance-based generalisation operators and the notion of $mg$. Section 3 analyses Plotkin's $lgg$ as a $mg$ for atoms. Section 4 extends the result to sets of literals (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 our conclusions, possibilities for applications, some open problems, and future work.

## 2 Distance-based generalisation framework

We present the main notions related to our framework for defining a concept of generalisation based on distances. For more details see [1].

Our approach aims to define generalisation operators for data embedded in a metric space $(X, d)$. These operators are denoted as $\Delta(E)$, where $E$ is a finite set of elements ($|E| \geq 2$) of $X$ to be generalised. A generalisation computed by $\Delta(E)$ will be expressed by a pattern $p$ belonging to a pattern language $\mathcal{L}$. In fact, every pattern $p$ represents a set of elements of $X$ and is denoted by $Set(p)$. Thus, we can say that an element $x \in X$ is covered by a pattern $p$, if $x \in Set(p)$. In the same way, $p$ is a generalisation of $E$ iff $E \subset Set(p)$. For instance, given the strings *abb* and *abc*, and the regular pattern $ab*$, then $Set(ab*) = \{ab, abc, aba, abb, abaa, ...\}$, and we say that $ab*$ covers the elements $abb \in Set(ab*)$ and $abc \in Set(ab*)$.

The following definition establishes the relationship between a generalisation operator defined in a metric space and the underlying distance.

**Definition 1.** ***(Distance-based generalisation operator)*** *Let $(X, d)$ be a metric space and let $\mathcal{L}$ be a pattern language. We say that $\Delta : 2^X \to \mathcal{L}$ is a distance-based generalisation operator, if for every finite set $E \subset X$, $p \in \mathcal{L}$, $\Delta(E) = p$, there exists a nerve $N(E)$*[1] *such that, for every pair of elements $x, y$ in $E$ which are directly linked in $N(E)$, $Set(\Delta(E))$ includes all the elements $z$ such that $d(x, z) + d(z, y) = d(x, y)$.*

Given a metric space $(X, d)$, we say that, for every $x, y, z \in X$, $z$ is between $x$ and $y$ if $d(x, y) = d(x, z) + d(z, y)$.

Another issue related to the generalisation operator is to determine when it performs the least general generalisation (*lgg*, in short). This is an important issue if we want the generalisations to “fit” a group of elements as closely as possible. Although the *lgg* is a widely studied concept in the field of ILP [8], it has not been studied when data is not described by means of atoms. Thus, the following constructions are an alternative and a more general notion of minimal (least general) generalisation for different sorts of data, when data is in a metric space.

First, we establish a criterion to determine which pattern is less general, given two patterns computed by two distance-based generalisation operators $\Delta(E)$ and $\Delta'(E)$, respectively. The least general generalisation operator $\Delta$ might not be unique, so we call it minimal. Then, the minimal distance-based generalisation operator $\Delta$ (*mg* operator, in short) is the one where for every set $E$ and for every distance-based operator $\Delta'$, the pattern $\Delta(E)$ is less general than $\Delta'(E)$.

[1] Given a metric space $(X, d)$ and a set of undirected connected graphs $S_G$, a nerve function $N : 2^X \to S_G$ maps every finite set $E \subset X$ 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.

In order to formalise our proposal, we could use the inclusion operation between sets ($\subset$) as a "mechanism" to compare how general two generalisations are. In other words, a generalisation $\Delta(E)$ is less general than a generalisation $\Delta'(E)$, if $Set(\Delta(E)) \subset Set(\Delta'(E))$. However, this leads to several problems (see [1] for details):

1. Most generalisations are not comparable, since neither $Set(\Delta(E)) \subset Set(\Delta'(E))$ nor vice versa.
2. The inclusion operator between sets ($\subset$) ignores the underlying distance.
3. The minimal generalisation may not exist for some pattern languages.

Therefore, these drawbacks lead us to introduce a more abstract generality criterion. It is more interesting to find some kind of 'function' that assigns a generality (cost or optimality) value to every pattern, making every pair of patterns comparable. For this purpose, we introduce a special function, called the cost function.

**Definition 2. (cost function)** *Let $(X, d)$ and $\mathcal{L}$ be a metric space and a pattern language, respectively. We say that the mapping $k : 2^X \times \mathcal{L} \to \mathcal{R}$ is a cost function, if for every pattern $p \in \mathcal{L}$ and $E \subset X$, such that $E \subset Set(p) \subset X$ and $Set(p) \neq X$, then $k(E, p) < \infty$.*

Logically, this definition gives almost complete freedom for how to choose $k$. For the metric spaces considered in this paper, if we are looking for minimal patterns, the idea is that the cost function must depend on the fit (i.e. minimality). For this reason, we define the cost function as $k(E, p) = c(E|p)$, where $c(E|p)$ measures how well the pattern $p$ fits the data $E$. However, for other metric spaces (sets, graphs, lists, ...), more complex cost functions can be defined by also considering how complicated the pattern is [1], following the $MDL/MML$ principle.

**Definition 3. (inclusion-preserving cost function)** *Let $(X, d)$ and $\mathcal{L}$ be a metric space and a pattern language, respectively. We say that the cost function $c(E|p)$ is inclusion-preserving if for every $E \subset X$ and pair of patterns $p$ and $p'$ such that $Set(p) \subset Set(p')$ then $c(E|p) \leq c(E|p')$.*

One interesting point in our approach is that $c(E|p)$ is expressed in terms of the distance employed. One possible way of defining some instances for $c(E|p)$ is by using the well-known concept of border of a set[2]. Intuitively, if a pattern $p_1$ fits $E$ better than a pattern $p_2$, $\partial Set(p_1)$ will somehow be closer to $E$ than $\partial Set(p_2)$.

As the border of a set exists in every metric space, several definitions of $c(E|p)$ can be employed for different sorts of data, as we show in Table 1. It is easy to show that all of them are inclusion-preserving.

Now, we can introduce the definition of $mg$ operator.

[2] We will say that an element $e$ belonging to set $A \subseteq X$ is a border point, if for every $\epsilon > 0$, $B(e, \epsilon)$ (where $B(e, r)$ is the closed ball with centre on $e$ and radius $r$ ) is not totally included in $A$. In the standard notation, the border of a set $A$ will be denoted by $\partial A$.

| Sort of data | $\mathcal{L}$ | $c(E\|p)$ |
|---|---|---|
| Any | Any | $\sum_{\forall e \in E} inf_{r \in R} B(e, r_e) \not\subset Set(p)$ |
| Any | Any | $\sum_{\forall e \in E} sup_{r \in R} B(e, r_e)$ |
| Any | Any | $\sum_{\forall e \in E} min_{e' \in \partial Set(p)} d(e, e')$ |
| Any | $Set(p)$ represents a bound set | $\sum_{\forall e \in E} (min_{e' \in \partial Set(p)} d(e, e')$ $+ max_{e'' \in \partial Set(p)} d(e, e''))$ |

**Table 1.** Some definitions of the function $c(E|p)$.

**Definition 4. (Minimal distance-based generalisation operator)** *Let $(X, d)$ be a metric space, and let $\Delta$ be a distance-based generalisation operator defined in $X$ using the pattern language $\mathcal{L}$. Given a cost function $k(\cdot, \cdot)$, we say that $\Delta$ is a mg operator for $k(\cdot, \cdot)$ in $\mathcal{L}$, if for every distance-based generalisation operator $\Delta'$, then $k(E, \Delta(E)) \leq k(E, \Delta'(E))$, for every finite set $E \subset X$.*

In general, deriving the *mg* operator is complicated because of the high variety of nerve functions $N(\cdot)$ that can be defined. In some problems, it does not make sense to explore all the nerve functions (e.g. in clustering), and we might be interested in computing *mg* operators relative to one specific nerve function, namely:

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

Next, we analyse and/or derive *mg* operators for the specific case of first-order logic data (atoms and clauses) embedded in a metric space.

## 3 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 do this, a distance function, a pattern language and a cost function are defined. In what follows, $\mathcal{L}$, denotes a first-order language defined over the signature $\langle \mathcal{C}, \mathcal{F}, \Pi, \mathcal{X} \rangle$, where $\mathcal{C}$ is a set of constants, $\mathcal{F}$ (and respectively $\Pi$) is a family that is indexed on $N$ (non negative integers) with $\mathcal{F}_n$ ($\Pi_n$) being a set of $n-$adic function (predicate) symbols and $\mathcal{X}$ is a (infinite) denumerable set of variable symbols. In the case of no ambiguity, both predicate and function symbols are referred to as symbols, and variable symbols are referred to as variables. $f/n$ (and respectively $p/n$) denotes a function symbol $f \in \mathcal{F}_n$ (and respectively $p \in \Pi_n$). Finally, the reader may refer to [5, 6] for any concept about logic programming and inductive logic programming which is not explicitly defined.

### 3.1 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 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 that counts 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 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 $lgg(e_1, e_2) = q(Y, f(Z))$, $size(e_1) = (3, 0)$, $size(e_2) = (2, 1)$, $size(lgg(e_1, e_2)) = (1, 2)$, 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 [3] 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 from each other. 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 if 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 $d_a$ denotes the distance described above.

### 3.2 The pattern language and the cost function

The pattern language is the Herbrand base with variables induced by the signature, that is, $\mathcal{L}$ coincides with $X_a$. For example, let $\mathcal{C} = \{a, b\}$ be a set of constants, $\mathcal{F} = \{f/1\}$ a set of function symbols, $\mathcal{X} = \{X_1, X_2, \ldots\}$ a denumerable set of variables and $\Pi = \{p/1, q/1\}$ a set of predicate symbols. Then, $\mathcal{L}_1 = \{p(a, X_1), p(X_1, a), p(X_1, X_2), p(f(a), b), \ldots q(a, X_1), q(X_1, a), \ldots\}$. Given a pattern $p$, $Set(p)$ denotes all the atoms in $X_a$ which are instances of $p$. For example, $p(a) \in Set(p(X))$.

Regarding the cost function, $c(E|p)$ is the first function in Table 1. 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 $E$, $c(E|p) = \infty$ iff $Set(p) = X_a$.

### 3.3 Defining a *mg* operator

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, where $E$ is a finite set of two or more atoms, $lgg(E)$ is the $mg$ for this metric space and this cost function.

**Proposition 1.** *Given the metric space $(X_a, d_a)$. If $\mathcal{L}$ is the Herbrand base with variables induced by the signature and $c(E|p) = \sum_{\forall e \in E} r_e$ (being $r_e$ =*

[3] By *general* we mean the well-known concept from logic programming.

*$inf_{r \in \mathcal{R}} B(e, r) \not\subset Set(p)$), then $\Delta(E) = lgg(E)$ is a mg operator for $d_a$, $\mathcal{L}$ and $c(E|p)$.*

*Proof.* First, let us show that $lgg(E)$ performs minimal patterns according to the cost function. For every generalisation $p$ of $E$, $p \in \mathcal{L}$, such that $E \subset Set(p)$, by definition of $lgg$, $Set(lgg(E)) \subset Set(p)$, and by Definition 3, $c(E|lgg(E)) \leq c(E|p)$.

Secondly, note that $lgg(E)$ is distance-based. Clearly, for every two elements $e_i, e_j \in E$, $Set(lgg(e_i, e_j)) \subset Set(lgg(E))$. According to Proposition 6 in [2], $lgg(e_i, e_j)$ is distance-based in $(X_a, d_a)$ and so $Set(E)$ contains all the elements between $e_i$ and $e_j$, for every $e_i$ and $e_j$. Then, simply defining, for instance, $N(E)$ as a complete graph, $lgg(E)$ is distance-based.

This result does not necessarily hold when the cost function or even the pattern language is changed. In [1], we explore the combination of different cost functions and pattern languages in further detail.

## 4 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, which is usually expressed as a logic implication with a disjunction of all the positive literals in the consequent (head) and a conjunction of all the negative literals in the antecedent (body). For instance,

$$C = \{class(X, c_1), \neg molec(X), \neg atom(X, h)\}$$

can be interpreted and represented as: $class(X, c1) : -molec(X) \wedge atom(X, h)$[4].

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.

### 4.1 The metric space

The distance we are going to use is defined in [12]. This distance is based on minimal matchings[5] 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 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, $D(\alpha_{A,B}) = \{a \in A : \exists (a, b) \in \alpha_{A,B}\}$. 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.

[4] As in Prolog notation, the symbol $: -$ denotes the logic implication symbol $\leftarrow$.

[5] A matching from set $A$ to set $B$ is an injective mapping which is not necessarily defined over all the elements in $A$.

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 $B$ that is not included in the matching. More formally,

$$d(\alpha_{A,B}, A, B) = \sum_{\forall (a_i,b_j)\in\alpha_{A,B}} d(a_i, b_j) + \frac{M}{2}(|B - \alpha_{A,B}(A)| + |A - D(\alpha_{A,B})|) \quad (1)$$

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

$$d_m(A, B) = min_{\forall \alpha_{A,B}} d(\alpha_{A,B}, A, B) \quad (2)$$

Unless we say otherwise, $(2^X, d_m)$ denotes the metric space of sets, and $(X, d)$ denotes the metric space of the elements of the sets. In our case, the metric space $(X, d)$ to be considered, which is denoted as $(X_l, d_l)$, is obtained by extending the space $X_a$ and the distance $d_a$ (defined for atoms in the previous section) to both positive and negative atoms, i.e. literals. This extension is trivial, since $p(\ldots)$ and $\neg p(\ldots)$ are considered incompatible literals. Hence it is like treating them as being built by different predicates[6]. According to [12], the constant $M$ must be greater or equal to the maximal distance between two elements in $X$ in order to $d_m(\cdot, \cdot)$ satisfies all the axioms of a metric. This restriction forces us to bound (restrict) the space $X$. Then, the restriction over $X_l$ 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. These are called bounded literals. Thus, let $\bar{X}_l$ and $M = (R, R)$ be the bounded space and the penalty, respectively.

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

The above restriction is finally neither a real nor theoretical problem. A representation of a real-life object always requires a finite number of symbols and all the results concerning $(X_l, d_l)$ hold for $(\bar{X}_l, d_l)$, as the following Proposition 2 shows.

**Proposition 2.** *Proposition 1 (i.e. lgg for bounded literals is a mg) holds for the metric space $(\bar{X}_l, d_l)$, where $\bar{X}_l = \{x \in X_l :$ number of non-variable symbols in $x \leq k\}$with $k$ being a constant.*

*Proof.* Trivially, for every $e_i, e_j \in E \subset \bar{X}_1$, if $e_k$ is an element between $e_i$ and $e_j$ in $\bar{X}_l$, it is also between them in the space $X_l$ since the distance function is the

[6] The $lgg$ of two incompatible literals is undefined [5].

same; thus, $e_k \in Set(lgg(E))$ and $lgg(E)$ is distance-based in $\bar{X}_l$. Also, for every generalisation $p$ of $E$, such that $E \subset Set(p)$ we have that $Set(lgg(E)) \subset Set(p)$ by definition of $lgg$ and $c(E|lgg(E)) \leq c(E|p)$ by Definition 3. Thus, $lgg$ for literals is a $mg$ operator.

Therefore, the metric space for clauses is $(2^{\bar{X}_l}, d_m)$.

## 4.2 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. Some examples of patterns could be,

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

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, $p_1 = \{C_{11} \equiv \{class(X, c_1), \neg molec(X), \neg atom(X, , h)\}$, $C_{12} \equiv \{class(X, c_1), \neg molec(X), \neg atom(X, , o)\}\}$ and $p_2 = \{C_{21} \equiv \{class(X, c_2), \neg molec(Y), \neg atom(Y, , c))\}\}$. 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 $p \in \mathcal{L}$, $Set(p)$ represents all those clauses in the metric space $2^{\bar{X}_l}$ which are $\theta$-subsumed by $p$. Thus, the clause $\{class(m_1, c_1), \neg molec(m_1), \neg atom(m_1, h)\}$ belongs to $Set(p_1)$.

As for the cost function, $c(E|p)$ is the first function in Table 1. Clearly, it is a cost function for $(2^{\bar{X}_l}, d_m)$ and $\mathcal{L}$ since for a finite set of elements $E$ and a pattern $p$ covering $E$, $c(E|p) = \infty$ iff $Set(p) = 2^{\bar{X}_l}$.

## 4.3 Defining *mg* operators

Unlike $(X_a, d_a)$, let us first see that $\Delta(E) = lgg(E)$ (where $lgg$ is the *least general generalisation* for clauses [11]) is not a $mg$ operator in $(2^{\bar{X}_l}, d_m)$. Although it can easily be shown that $lgg(E)$ is a minimal pattern in our framework, $lgg(E)$ 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 $lgg(A, B)$ (see Example 2).

*Example 2.* 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.
We can easily see that $d_m(A, B) = (8, -8) = d_m(A, C) + d_m(C, B)$. However,

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

but $C \equiv: -p(f(b), f(b)), p(g(a), e)$ is not $\theta$-subsumed by $lgg(A, B)$.

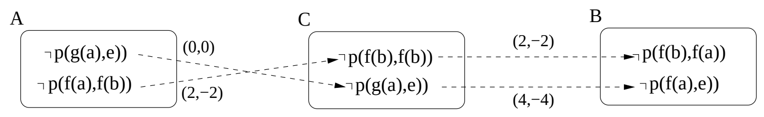

**Fig. 1.** The arrows and the labels indicate the optimal mappings between the different pairs of sets and the distance between the matched elements, respectively.

Unfortunately, defining $mg$ operators in this space is not as intuitive as in the previous section. We tackle the problem in a different way. First, we focus on determining binary $mg$ operators. Then we study if we can obtain $n$-ary $mg$ operators by combining these binary $mg$ operators.

**Proposition 3.** *Let $A$, $B$ and $C$ be 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 $c_k \in C$ that satisfies $d_l(a_i, b_j) = d_l(a_i, c_k) + d_l(c_k, b_j)$.*

*Proof.* 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,

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

where $k_{\alpha_{A,C}}$ (respectively, $k_{\alpha_{C,B}}$) denotes the number of elements of $A$ and $C$ (respectively, $C$ and $B$) 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, $\alpha'_{A,B}(A) = \alpha_{C,B}(\alpha_{A,C}(A))$. Keeping $\alpha'_{A,B}$ in mind, the sum $d_m(A,C) + d_m(C,B)$ can be written as,

$$\begin{aligned} d_m(A,C) + d_m(C,B) = &\textstyle\sum_{\forall (a_i,b_j)\in\alpha'_{A,B}} (d_l(a_i,\alpha_{A,C}(a_i)) + d_l(\alpha_{A,C}(a_i),b_j)) \\ &+ \textstyle\sum_{\forall a_i \in D(\alpha_{A,C}) - D(\alpha'_{A,B})} d_l(a_i,\alpha_{A,C}(a_i)) \\ &+ \textstyle\sum_{\forall c_i \in D(\alpha_{C,B}) - \alpha_{A,C}(A)} d_l(c_i,\alpha_{C,B}(c_i)) \\ &+ \frac{M}{2}\cdot k_{\alpha_{A,C}} + \frac{M}{2}\cdot k_{\alpha_{C,B}} \end{aligned} \tag{3}$$

The first term on the right-hand side of Equation (3) considers all the ordered pairs belonging to the matchings that share an element $c_i \in C$. The second and third terms concern those ordered pairs in $\alpha_{A,C}$ and $\alpha_{C,B}$ (respectively), which were not taken into account by the first term. Finally, the two last terms come from those unmatched elements.

Next, the chain of inequalities shown in Equation (4) can be derived as follows. First, we apply the triangle inequality over the first term on the right-hand side of Expression (3). Second, we remove the second and the third terms. Third, we apply $k_{\alpha_{A,C}} + k_{\alpha_{C,B}} \geq k_{\alpha'_{A,B}}$. And, finally, the last inequality is a direct consequence of the $d_m(\cdot,\cdot)$ definition (see Equation (2)).

$$\begin{aligned}
d_m(A,C)+d_m(C,B) \geq & \textstyle\sum_{\forall(a_i,b_j)\in\alpha'_{A,B}} d_l(a_i,b_j) \\
& +\textstyle\sum_{\forall a_i\in D(\alpha_{A,C})-D(\alpha'_{A,B})} d_l(a_i,\alpha_{A,C}(a_i)) \\
& +\textstyle\sum_{\forall c_i\in D(\alpha_{C,B})-\alpha_{A,C}(A)} d_l(c_i,\alpha_{C,B}(c_i)) \\
& +\tfrac{M}{2}\cdot k_{\alpha_{A,C}}+\tfrac{M}{2}\cdot k_{\alpha_{C,B}} \\
\geq & \textstyle\sum_{\forall(a_i,b_j)\in\alpha'_{A,B}} d_l(a_i,b_j) \\
& +\tfrac{M}{2}\cdot(k_{\alpha_{A,C}}+k_{\alpha_{C,B}}) \\
\geq & \textstyle\sum_{\forall(a_i,b_j)\in\alpha'_{A,B}} d_l(a_i,b_j)+\tfrac{M}{2}\cdot(k_{\alpha'_{A,B}})=d(\alpha'_{A,B},A,B) \\
\geq & d_m(A,B)
\end{aligned} \tag{4}$$

The equality $d(A,C)+d(C,B)=d(A,B)$ holds only if all the inequalities ($\geq$) on the right-hand-side of Equation (4) becomes an equality. Among all these transformations, only the first and the last one are necessary to prove the proposition. The first inequality turns into an equality if the element $c_k=\alpha_{A,C}(a_i)$ in the first term in the right-hand-side of (3) 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 if $\alpha'_{A,B}$ is an optimal matching, and if this occurs then the last inequality is transformed into an equality.

The fact that the $lgg(\cdot)$ for atoms is distance-based suggests 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 $B$. However, 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 $\alpha_{A,B}$. Since the $lgg$ 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 glance, this definition of $\Delta$ 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 corresponding pattern might not be distance-based (for further details see [1]). We deal with 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 do not belong to $Set(p_1)$ or to $Set(p_2)$. 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, Propositions 4 and 5 characterise the family of all the distance-based binary generalisation operators. They show 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.

**Proposition 4.** *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 $\Delta^*(A,B) = p$ defined as*

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

*is distance-based, where the repeated variables occurring in different $lgg(a_i, b_j)$ are independent.*

*Proof.* From Proposition 3, if a set $D$ is between $A$ and $B$ (i.e. $d_m(A,B) = d_m(A,D) + d_m(D,B)$) then there exists an optimal mapping $\alpha_{A,B}$ such that for every $(a_i, b_j) \in \alpha_{A,B}$ there exists $d_k \in D$ with $d_k$ being between $a_i$ and $b_j$. As the $lgg$ for literals is distance-based, necessarily $d_k \in Set(lgg(a_i, b_j))$ and $D \in Set(\{lgg(a_i, b_j) : \forall (a_i, b_j) \in \alpha_{A,B}\})$. Since all the optimal mappings are taken into consideration, for every set $D$ between $A$ and $B$, $D \in Set(p)$, and therefore, $\Delta^*(A,B)$ is distance-based.

**Proposition 5.** *Given the metric space $(2^{\bar{X}_l}, d_m)$ 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 pair of clauses $A$ and $B$, $Set(\Delta^*(A,B)) \subset Set(\Delta(A,B))$, with $\Delta^*$ being the distance-based operator defined in Proposition 4.*

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

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

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

$(\leftarrow)$ Thus, $Set(\Delta^*(A,B))$ is a subset of $Set(\Delta(A,B))$. Since $\Delta^*(A,B)$ is distance-based, automatically $\Delta(A,B)$ is distance-based.

Now, we must determine the $mg$ binary operator. It is direct from Proposition 5 and Definition 3, since, for every distance-based operator $\Delta(A,B)$ and for every pair of elements $A$ and $B$, we know that $Set(\Delta^*(A,B)) \subset Set(\Delta(A,B))$ and therefore, $c(\{A,B\}|\Delta^*(A,B)) \leq c(\{A,B\}|\Delta(A,B))$. 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 $\Delta'$, and fixing a nerve-function $N(\cdot)$. Given the set $E = \{e_1, \ldots, e_n\}$, $\Delta_{N(E)}(E) = \bigcup_{\forall (e_i, e_j) \in N(E)} \Delta'(e_i, e_j)$.

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 6 states that if $\Delta' = \Delta^*$ in the expression above, then $\Delta_{N(E)}(E)$ is a $mg$ that is related to a nerve function $N(E)$.

**Proposition 6.** *Let $\Delta^*(A,B)$ be the binary generalisation operator introduced in Proposition 4 and let $c(E|p) = \sum_{\forall e \in E} r_e$ (with $r_e = inf_{r\in\mathcal{R}} B(e,r) \not\subset Set(p)$) be the cost function. Then*

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

*is a mg operator that is related to the nerve function $N(E)$.*

*Proof.* 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 $c(E|\Delta'_{N(E)}) < c(E|\Delta_{N(E)})$. We define a binary distance-based operator $\Delta''$ 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 5,

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

As occurs for every $(e_i,e_j) \in N(E)$, $Set(\Delta_{N(E)}(E)) \subset Set(\Delta'_{N(E)}(E))$, and, consequently, $\Delta'_{N(E)}$ cannot be $mg$.

Before concluding, note that a pattern computed by a $mg$ cannot contain repeated variables in the different $lgg(a_i,b_j)$ from the same clause. On the one hand, this makes sense since the metric does not capture the semantic of repeated variables occurring in different atoms. Hence, for the sets $A = \{p(a),q(a)\}$, $B = \{p(b),q(b)\}$, and $C = \{p(c),q(d)\}$, $d_l(A,B) = d_l(A,C)$ when, intuitively, $B$ should be more similar to $A$. This is a strong constraint to express some real-world properties. However, this concerns only the $mg$ operator. It does not mean that distance-based operators cannot contain repeated variables in different atoms in general. For instance, Proposition 5 suggests that we could adapt a bottom-up ILP inference algorithm to take the pattern (clauses) computed by the $mg$ as input. The output of the algorithm is a more general pattern than the input pattern. It is also distance-based and contains repeated variables among different atoms. Furthermore, this fact indicates that some adaptations of the ILP algorithms can be viewed as distance-based operators.

## 5 Conclusions and future work

This work develops the notion of $mg$ operator for every sort of data that is embedded in a metric space. Here we include a definition of the framework, following the main ideas explained in [1] in order to address the minimal generalisation operators for some first-order objects.

We have shown that Plotkin's $lgg$ can be seen as a particular case of this setting because the classical $lgg$ for atoms is a $mg$ operator w.r.t. the metric space defined in [13] and a specific (but simple) cost function. We showed this result in [2], but only as a binary operator and without the notion of minimality. The notion of cost function, which is exclusively defined in terms of distances, completes the connection between the concepts of distance, pattern and generalisation that we established in previous works. Furthermore, in this work, we have suggested that different *mg* operators for atoms can be obtained by changing the cost function, which can be an alternative to *lgg* for redesigning existing ILP methods or for deriving new ones. As for clauses, Plotkin's *lgg* has been shown not to be a *mg* operator for the particular metric space derived from the distance introduced by [12]. Due to the complexity of this metric space, a new *mg* operator relative to one specific nerve function has been introduced. Other distances, cost functions and pattern languages have also been studied (see [1]). For instance, Plotkin's *lgg* for atoms is not distance-based w.r.t. the distance introduced in [9]. This is an example that some distances are more appropriate than others in a logic context.

The applicability of the framework and the new lines of research are numerous. First, we think that the new *mg* operators (and the new understanding of the *lgg* as a distance-based operator) can be useful to redefine, reunderstand, and cross different methods and ideas within ILP. For instance, some bottom-up ILP methods (some of which have almost been forgotten since the early nineties) can be adapted to work with newly derived *mg* operators, as we outlined at the end of the section above. Furthermore, the adapted ILP methods would be distance-based operators. For instance, some size measures for atoms and clauses (see Section 14.9 in [10]) could be used for the cost function in a similar way as they were used in the context of refinement. Second, distance-based *mg* would be a good link to extend ILP techniques outside ILP, since we have defined them for many other data types: lists, trees, graphs, sets (see [1] [3]). Specifically, bottom-up ILP methods could by adapted to other kinds of complex objects (not necessarily first-order). For instance, we are currently investigating the possibility of applying ILP bottom-up methods of this kind to lists or graphs. Third, we think that provided that we have an adequate *mg* operator (as some of the ones studied or derived in this work), we could easily adapt traditional distance-based techniques to ILP such as clustering techniques (k-means, minimum-spanning tree, etc.) or classification techniques (k-nn) in a more sophisticated way than has been done to date. In other words, we can turn these techniques from instance-based techniques to model-based techniques.

One of the specific issues that must be addressed for any new *mg* is, logically, its efficiency. In some cases, if the *mg* is distance-based, but computationally expensive to find, we might need to find heuristics or approximations. Some of these approximations (as we mentioned in the specific case of the *mg* in Section 4) consist of making one optimal matching instead of all the possible optimal matchings between two elements. We have explored this possibility for lists (see [1]) by introducing the notion of pseudo distance-based operator. However, it is important to highlight that, in our framework, the *mg* operators are based on a cost function. This is more flexible than when the *mg* operator is solely based on the notion of generalisation or inclusion. If all the distances are pre-computed between elements, the computation of the *mg* can be speeded up.

We are currently adapting classical (and, for the moment, simple) machine learning techniques to our framework, such as a nearest-neighbour classifier based on *mg* for several data sorts or a distance-based decision tree.

## Acknowledgement

We thank the anonymous reviewers for their valuable and insightful comments.

## References

1. V. Estruch. *A distance-based generalisation framework for model-based learning from structured data.* PhD thesis, Technical University of Valencia, 2007. http://www.dsic.upv.es/∼flip/#Papers.
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 Mining and Learning with Graphs, MLG06*, 2006.
4. T. Gaertner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.
5. N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, New York, 1994.
6. J. W. Lloyd. *Foundations of logic programming; (2nd extended ed.).* Springer-Verlag New York, Inc., New York, NY, USA, 1987.
7. S. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.
8. S. H. Muggleton. Inductive logic programming: Issues, results, and the challenge of learning language in logic. *Artificial Intelligence*, 114(1–2):283–296, 1999.
9. S-H. Nienhuys-Cheng. Distance between Herbrand interpretations: A measure for approximations to a target concept. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *LNCS*, pages 213–226. Springer-Verlag, 1997.
10. S-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming.* Springer-Verlag New York, Inc., 1997.
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 *CompulogNet Area Meeting on Computational Logic and Machine Learning*, pages 35–41. University of Manchester, UK, 1998.