Fundamenta Informaticae 90 (2009) 1–24
IOS Press

A General Similarity Framework for Horn Clause Logic

S. Ferilli *, **T.M.A. Basile, M. Biba, N. Di Mauro F. Esposito**
Dipartimento di Informatica
Università di Bari
via E. Orabona, 4 - 70125 Bari - Italia
{*ferilli, basile, biba, ndm, esposito*}*@di.uniba.it*

Abstract. First-Order Logic formulæ are a powerful representation formalism characterized by the use of relations, that cause serious computational problems due to the phenomenon of indeterminacy (various portions of one description are possibly mapped in different ways onto another description). Being able to identify the correct corresponding parts of two descriptions would help to tackle the problem: hence, the need for a framework for the comparison and similarity assessment. This could have many applications in Artificial Intelligence: guiding subsumption procedures and theory revision systems, implementing flexible matching, supporting instance-based learning and conceptual clustering. Unfortunately, few works on this subject are available in the literature. This paper focuses on Horn clauses, which are the basis for the Logic Programming paradigm, and proposes a novel similarity formula and evaluation criteria for identifying the descriptions components that are more similar and hence more likely to correspond to each other, based only on their syntactic structure. Experiments on real-world datasets prove the effectiveness of the proposal, and the efficiency of the corresponding implementation in the above tasks.

Keywords: First-Order Logic, Logic Programming, Similarity/Distance Measures

1. Introduction

First-order logic (*FOL* for short) is a powerful formalism, that is able to express relations between objects and hence can overcome the typical limitations shown by propositional or attribute-value representations. As a consequence and tradeoff for their expressive power, the presence of relations causes various portions of one description to be possibly mapped in (often many) different ways onto another description,

*Address for correspondence: Dipartimento di Informatica, Università di Bari, via E. Orabona, 4 - 70125 Bari - Italia

a problem known as *indeterminacy*, which poses serious problems of computational effort when two descriptions have to be compared to each other. Hence, the need for a set of general criteria that are able to support the comparison between formulæ. Specifically, our objective is developing a similarity framework for FOL descriptions, based on a measure and a number of criteria that must be simple (in order to ensure efficient computation), sensible (they have to reflect as much as possible the intuitive way in which humans compare two descriptions), effective (must capture as much information as possible about the items to be compared), flexible (allow to weight differently the two items) and general (are **not** devised *ad-hoc* for some problem).

This could have many applications, particularly in the Artificial Intelligence and Machine Learning community. For instance, the similarity criteria could be exploited by a subsumption procedure to converge quickly towards the correct associations. More generally, even when two formulæ do not correspond exactly, one could nevertheless be interested in assessing a degree of similarity between them, which is typically obtained by means of functions that could take advantage from the same criteria. This would help to overcome the strict distinction operated by FOL between true and false assertions, that sometimes is too rigid to deal with real-world cases (e.g., when a definition does not completely account for an observation, even if this is due to just a few characteristics), so that a *flexible matching* procedure would be more appropriate than an exact one. Instance-based techniques (e.g. Case-Based Reasoning and k-Nearest Neighbor) strongly rely on similarity measures for classifying unseen observations according to the closest known prototypes. From the opposite perspective, a similarity measure can be interpreted as a means for estimating the distance between the formulæ, to be exploited by unsupervised learning techniques for grouping observations into homogeneous concepts (called *conceptual clustering* in FOL).

As regards supervised learning techniques, it often happens that a concept is expressed by many alternative definitions each of which, at least in principle, is intended to represent a particular form of the concept, according to the intuition that the concept itself is polymorfous, and hence cannot be captured by a single definition. As a consequence, such definitions are expected to show a high degree of orthogonality, referring to distinct portions (ideally, defining a partition of the whole set) of concept instances. In practice, when the definitions are learned empirically and do not undergo a systematization process, this is not always the case, and the scope of definitions often overlap. Hence, the need for some tool that could help Machine Learning systems in choosing the best definition to be refined when the current theory proves unable to account for a new given observation. More operationally, there is a need for a function that, given an observation and various definitions for the concept it belongs to, assigns to each of the latter a value expressing its degree of syntactic similarity with the former. Additionally, when two descriptions (e.g., a definition and an observation) must be generalized, the similarity criteria could help the procedure in focussing on the components that are more similar and hence more likely to correspond to each other. Clearly, this concerns the semantic aspects of the domain, and hence there is no precise (i.e., algorithmic) way for recognizing the correct (sub-)formulæ. Thus, the problem must be attacked heuristically, by developing some method that can hypothesize which (part of a) description refers to which (part of the) other, based only on their syntactic structure. To these aims, partial similarities among subparts of the descriptions must be searched for.

A particular kind of FOL formulæ are *Horn clauses*, i.e. expressions of the form $l_1 \wedge \cdots \wedge l_n \Rightarrow l_0$ (usually represented in Prolog style as l_0 :- $l_1, \ldots, l_n$) to be interpreted as "l_0 (called *head* of the clause) is true, provided that l_1 and ... and l_n (called *body* of the clause) are all true". The l_i's are *atoms*, where an atom is a predicate applied to a proper number of terms (called its *arity*). A predicate p of arity k is usually denoted as p/k. Given a clause C, $head(C)$ denotes the head atom of C and $body(C)$ the

set of atoms in the body of C. An interesting perspective on FOL is that, by restricting to sets of Horn clauses (called Logic programs), it defines Logic Programming [19], an important paradigm in Artificial Intelligence (many first-order Machine Learning systems infer theories in the form of logic programs). $terms(E)$ denotes the set of terms that appear in a FOL expression E. Informally, a clause C is linked if any two of its atoms can be connected by a chain of atoms in C such that adjacent atoms in the chain have some term in common (see [19] for a formal definition): linked sub-parts of non-linked clauses, having no connection between each other, can be dealt with separately. Datalog [6] is (at least, syntactically) a restriction of Logic Programming that allows only variables and constants as terms (and hence avoids the use of function symbols): the *flattening/unflattening* procedures [26] can translate generic Horn clauses into Datalog ones and vice-versa. Thus, we can focus on the case of linked Datalog clauses without loss of generality.

In the next sections, the criteria and a corresponding formula and on which basing similarity considerations between descriptions will be presented, followed by a pool of techniques to assess the similarity between clause components, that are intended to represent a good tradeoff between significance, effectiveness and expressiveness on one side, and computational efficiency on the other. Then, Section 4 will discuss computational complexity issue, present related work and summarize experimental results on different tasks. Lastly, Section 5 will conclude the paper and outline future work directions.

2. Similarity Parameters and Formula

Intuitively, the evaluation of similarity between two expressions i' and i'' might be based both on the presence of common features[1], which should concur in a positive way to the similarity evaluation, and on the features of each expression that are not owned by the other (let us define this as the *residual* of the former with respect to the latter), which should concur negatively to the whole similarity value assigned to them [18]. More precisely, two distinct residuals exist, each expressing the features of one of the two expressions that the other does not own[2]. Thus, plausible parameters on which basing similarity between i' and i'' are:

n , the number of features owned by i' but not by i'' (*residual* of i' wrt i'');

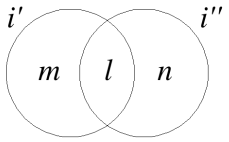
l , the number of features owned both by i' and by i'';

m , the number of features owned by i'' but not by i' (*residual* of i'' wrt i').

Indeed, other classical and state-of-the-art distance measures in the current literature, mostly developed in the propositional setting, are based on the same parameters: e.g., the one developed by Tverski [30],

[1]The exact definition of *feature* depends on the kind of expressions to be compared, and will be given in the following for terms, atoms and sequences of atoms.

[2]The situation is immediately visualized through a Venn diagram, in which each of the two sets represents the features of an expression, so that their intersection represents the common features, while the two symmetrical differences represent the residuals.



by Dice ($S = 2l/(n+2l+m)$) or Jaccard's index ($J(A,B) = l/(n+l+m)$) and distance ($J_\delta(A,B) = 1 - J(A,B) = (n+m)/(n+l+m)$). However, all of them show a behaviour that is not desirable in specific cases. For instance, Dice's measure and Jaccard index evaluate to 0 when the two expressions have no feature in common, independently of how many different features they have, and hence cannot distinguish between expressions that have no features in common but a different amount of extra features each. Such considerations led us to the definition of a similarity function that could overcome these limitations.

We developed a novel similarity function, $sf(\alpha, i', i'')$, that intuitively expresses the degree of similarity between two expressions i' and i'' based on the above parameters, also including an additional parameter α, $0 \leq \alpha \leq 1$, that weights the importance of either expression. Thus, $\mathrm{sf}(\alpha, i', i'')$ can be written as $\mathrm{sf}(\alpha, n, l, m)$, where n, l and m are the similarity parameters referred to i' and i''. In the following we will use both forms indifferently, as needed.

$$\mathrm{sf}(\alpha, i', i'') = \mathrm{sf}(\alpha, n, l, m) = \alpha \frac{l+1}{l+n+2} + (1-\alpha)\frac{l+1}{l+m+2} \tag{1}$$

First of all, note that the left-hand-side ratio refers to expression i', while the right-hand-side ratio refers to expression i''. However, for $\alpha = 0.5$, the function is symmetric with respect to the two expressions to be compared. It takes values ranging in the classical spectrum $]0, 1[$, to be interpreted as the degree of similarity between the two expressions. This can represent an aid for human interpretation, since we are used to it from the theory of probability. Thus, one can consider the returned value as the level of likelihood/confidence that the two expressions under comparison are actually similar. A complete overlapping of the two ($n = m = 0$) tends to the limit of 1 as long as the number of common features grows. The full-similarity value 1 is never reached, consistently with the intuition that the only case in which this should happen is $i' = i''$ (i.e. the two expressions are exactly the same formula, which can be immediately checked before applying *sf*). Indeed, in the following, we assume that $i' \neq i''$, thus excluding such an obvious case (that rarely occurs in the real world). Conversely, in case of no overlapping ($l = 0$) the function will tend to 0 as long as the number of non-shared features grows. This is consistent with the intuition that there is no limit to the number of different features owned by the two descriptions, which contribute to make them ever different. One last case is worth being considered: when there are no features at all associated to two descriptions under comparison (i.e., $n = l = m = 0$) the function evaluates to $1/2$, which can be considered intuitively correct for a case of maximum uncertainty. Although it might seem a meaningless case, it could happen, for instance, when comparing a model against an observation, both of which descriptions might include an object with no features: when comparing such objects, one cannot know whether the overlapping is actually total because in fact both of them have indeed no relevant feature at all, or the observed object actually has no relevant features but it just happened that previous generalizations have dropped from the model all the features that it previously owned.

2.1. Weight Definition

Some considerations are worth concerning weight assignment. Parameter α was introduced to give different importance to either of the two structures being compared (this might be typically needed when the comparison concerns a model against an observation); being associated to the left-hand-side ratio, it actually weights the relative importance of expression i' with respect to expression i'' ($\alpha = 0.5$ would yield

the basic function). Although α can, of course, be manually set in order to reflect the intended relative importance of the model and the observation, it would be useful to automatically set it, both for freeing the experimenters from further parameters estimation (which is a typically heuristic and experience-based task), and for making the function itself more flexible in adapting to the specific model-observation pair at hand.

Intuitively, the importance of any of the two expressions should increase as long as the number of overlapping features grows, and as long as the number of non-overlapping features decreases, which leads (as a first approximation) to $\alpha' = l/n$ for i' and $\alpha'' = l/m$ for i''. This suffers of problems of infiniteness or indefiniteness as soon as any of the three parameters evaluates to zero, hence a better setting is $\alpha' = (l+1)/(n+1); \alpha'' = (l+1)/(m+1)$. Now, their sum:

$$\alpha' + \alpha'' = \frac{l+1}{n+1} + \frac{l+1}{m+1} = \frac{(l+1)(m+1+n+1)}{(n+1)(m+1)}$$

must be normalized to 1, so that α can be obtained by switching to the portions that α' and α'' represent of this sum:

$$\alpha = \frac{\frac{l+1}{n+1}}{\frac{(l+1)(m+n+2)}{(n+1)(m+1)}} = \frac{m+1}{n+m+2}$$

$$(1-\alpha) = \frac{\frac{l+1}{m+1}}{\frac{(l+1)(m+n+2)}{(n+1)(m+1)}} = \frac{n+1}{n+m+2}$$

Notice that, if i is the model, n is expected to be generally less than m, since the model has probably lost some properties due to previous generalizations. Thus, the above weights actually give more credit to the model than to the observation, as desired. Moreover, interestingly, the weights do not depend on the overlapping.

Being an evaluation of the automatic weighting technique outside the scope of this paper, in the following $\alpha = 0.5$ will always be assumed, in which case we will write $\mathrm{sf}(i', i'') = \mathrm{sf}(n, l, m) = \mathrm{sf}(0.5, n, l, m)$.

3. Similarity Criteria for Horn Clause Components

Although the new similarity function definition is determinant in improving performance over the previous ones, as we will see in the experiments, the main contribution of this paper lays in the repeated and pervasive exploitation of the formula in various combinations that can assign a similarity degree to the different clause constituents, up to the whole clauses. In FOL formulæ, terms represent specific objects, that are related to each other by means of predicates. Accordingly, two levels of similarity can be defined for pairs of first-order descriptions: the *object* level, concerning similarities between the terms referred to in the descriptions, and the *structure* one, referring to how the nets of relationships in the descriptions overlap.

In the case of clauses, since the head is unique (and hence can be uniquely matched), we can use it as a starting point for the comparison. Specifically, we will consider as comparable only clauses having atoms of the same arity n in the head, and the comparison outcome will be interpreted as the degree of similarity between the two n-tuples of terms in the heads. In case the predicates in the heads of

the two clauses to be compared are different, for the sake of uniformity, we will assume that they are replaced by a new (not already present in the clauses) dummy predicate during the comparison. Thus, comparison of two clauses $p'(t'_1, \ldots, t'_n)$:- $l'_1, \ldots, l'_{n'}$ and $p''(t''_1, \ldots, t''_n)$:- $l''_1, \ldots, l''_{n''}$ will provide a degree of similarity between the n-tuples $< t'_1, \ldots, t'_n >$ and $< t''_1, \ldots, t''_n >$, and will be actually carried out on two corresponding clauses $dummy(t'_1, \ldots, t'_n)$:- $l'_1, \ldots, l'_{n'}$ and $dummy(t''_1, \ldots, t''_n)$:- $l''_1, \ldots, l''_{n''}$.

Example 3.1. Comparison of two clauses $father(carl, mary)$:- $l'_1, \ldots, l'_{n'}$ and $mother(ann, john)$:- $l''_1, \ldots, l''_{n''}$ will provide a degree of similarity between the couples $< carl, mary >$ and $< ann, john >$, which will be actually evaluated on clauses $dummy(carl, mary)$:- $l'_1, \ldots, l'_{n'}$ and $dummy(ann, john)$:- $l''_1, \ldots, l''_{n''}$.

Let us consider, as a running example throughout the paper, the following two clauses (in this case, a rule C and a classified observation E, although all considerations will apply to the general case as well):

$C : h(X)$:- $p(X,Y), p(X,Z), p(W,X), r(Y,U), o(Y,Z), q(W,W), s(U,V),$
$\pi(X), \phi(X), \rho(X), \pi(Y), \sigma(Y), \tau(Y), \phi(Z), \sigma(W), \tau(W), \pi(U), \phi(U).$

$E : h(a)$:- $p(a,b), p(a,c), p(d,a), r(b,f), o(b,c), q(d,e), t(f,g),$
$\pi(a), \phi(a), \sigma(a), \tau(a), \sigma(b), \tau(b), \phi(b), \tau(d), \rho(d), \pi(f), \phi(f), \sigma(f).$

3.1. Term Similarity

Consider in the following two clauses C' and C''. Call $T' = terms(C') = \{t'_1, \ldots, t'_n\}$, and $T'' = terms(C'') = \{t''_1, \ldots, t''_m\}$. Intuitively, for evaluating the similarity between a pair of terms $(t', t'') \in T' \times T''$, two kinds of term features can be distinguished: the properties they own and the roles they play in relation to other terms.

Definition 3.1. (Term features)
Consider a clause C, and an atom in C built on predicate p and having a term t among its arguments. If p is a unary predicate, then it is a *characteristic feature* (or *property*) of t in C; otherwise, let n be the arity of p, and i the position of t among its arguments, then $p/n.i$ is a *relational feature* (or *role*) of t in C.

Two corresponding similarity values can be associated to t' and t'': a *characteristic similarity*, where (1) is applied to values related to the properties, and a *relational similarity*, based on how many times the two terms play the same or different roles in the n-ary predicates.

Definition 3.2. (Characteristic Similarity)
Let P' be the set of characteristic features of t' in C' and P'' be the set of characteristic features of t'' in C''. The *characteristic similarity* between t' and t'' is computed as

$$\mathrm{sf}_c(t', t'') = \mathrm{sf}(n_c, l_c, m_c)$$

where $n_c = |P' \setminus P''|$ (*characteristic residual* of t' wrt t''), $l_c = |P' \cap P''|$ and $m_c = |P'' \setminus P'|$ (*characteristic residual* of t'' wrt t')[3].

[3]Intuitively, $P' \setminus P''$ are the properties of t' in C' but not of t'' in C''; $P' \cap P''$ are the common properties between t' in C' and t'' in C''; $P'' \setminus P'$ are the properties of t'' in C'' but not of t' in C'.

Relational features represent roles played by terms, since different argument positions in a predicate actually refer to different roles played by its arguments. For instance, in the binary predicate `on_top/2`, the first argument position (*on_top*/2.1) identifies the role of the upper object, and the second one (*on_top*/2.2) represents the role of the lower one. Due to the possibility that the same term plays multiple times the same role in different relations (e.g., an object laying on many others), we have to take into account the number of occurrences, and hence consider *multi*sets instead of sets.

Definition 3.3. (Relational Similarity)
Let R' be the multiset of relational features of t' in C' and R'' be the multiset of relational features of t'' in C''. Then, the *relational similarity* between t' and t'' can be computed as

$$\mathrm{sf}_r(t', t'') = \mathrm{sf}(n_r, l_r, m_r)$$

where $n_r = |R' \setminus R''|$ (*relational residual* of t' wrt t''), $l_r = |R' \cap R''|$ and $m_r = |R'' \setminus R'|$ (*relational residual* of t'' wrt t')[4].

Overall similarity between two terms, called *object similarity*, is defined as

$$\mathrm{sf}_o(t', t'') = \mathrm{sf}_c(t', t'') + \mathrm{sf}_r(t', t'') \tag{2}$$

that ranges in $]0, 2[$ (but can obviously be normalized to $]0, 1[$ if needed).

Example 3.2. The sets of properties and roles for each term in C and E, and the comparison for some of the possible pairs, are reported in Table 1.

3.2. Structural Similarity

When checking for the structural similarity of two formulæ, many terms can be involved, and hence their mutual relationships represent a constraint on how each of them in the former formula can be mapped onto another in the latter. The structure of a formula is defined by the set of n-ary predicates, and specifically the way in which they are applied to the various terms to relate them (i.e., *atoms*, according to the terminology introduced above), for which reason in the following we will deal only with predicates and atoms with arity greater than 1 in the rest of this section. This is the most difficult part, since relations are specific to the first-order setting and are the cause of indeterminacy in mapping (parts of) a formula into (parts of) another one.

Definition 3.4. Given two clauses C' and C'', a *term association* θ is a subset of $terms(C') \times terms(C'')$. A pair $(t', t'') \in \theta$ is written t'/t'' and called a *binding*. A term association is said to be *consistent* if it is a bijection, otherwise it is said to be *inconsistent*.

Note that the union of term associations is a term association itself.

Definition 3.5. (Matching association)
The *matching association* between two atoms $l' = p'(t'_1, \ldots, t'_n)$ and $l'' = p''(t''_1, \ldots, t''_n)$ is defined as $\theta_{l'/l''} = \{t'_1/t''_1, \ldots, t'_n/t''_n\}$; it is undefined for atoms built on predicates having different arity. The

[4] Intuitively, $R' \setminus R''$ are the occurrences of roles that t' plays in C' but t'' does not play in C''; $R' \cap R''$ are the occurrences of roles that both t' in C' and t'' in C'' play; $R'' \setminus R'$ are the occurrences of roles that t'' plays in C'' but t' does not play in C'.

Table 1. Object Similarity

C			E		
t'	P'	R'	t''	P''	R''
X	$\{\pi, \phi, \rho\}$	$\{p/2.1, p/2.1, p/2.2\}$	a	$\{\pi, \phi, \sigma, \tau\}$	$\{p/2.1, p/2.1, p/2.2\}$
Y	$\{\pi, \sigma, \tau\}$	$\{p/2.2, r/2.1, o/2.1\}$	b	$\{\sigma, \tau\}$	$\{p/2.2, r/2.1, o/2.1\}$
Z	$\{\phi\}$	$\{p/2.2, o/2.2\}$	c	$\{\phi\}$	$\{p/2.2, o/2.2\}$
W	$\{\sigma, \tau\}$	$\{p/2.1, p/2.1, p/2.2\}$	d	$\{\tau, \rho\}$	$\{p/2.1, p/2.1\}$
U	$\{\pi, \phi\}$	$\{r/2.2, s/2.1\}$	f	$\{\pi, \phi, \sigma\}$	$\{r/2.2, t/2.1\}$

Some comparisons:

t'/t''	$(P' \setminus P''), (P' \cap P''), (P'' \setminus P')$	$(R' \setminus R''), (R' \cap R''), (R'' \setminus R')$
X/a	$\{\rho\}, \{\pi, \phi\}, \{\sigma, \tau\}$	$\emptyset, \{p/2.1, p/2.1, p/2.2\}, \emptyset$
Y/b	$\{\pi\}, \{\sigma, \tau\}, \emptyset$	$\emptyset, \{p/2.2, r/2.1, o/2.1\}, \emptyset$
Y/c	$\{\pi, \sigma, \tau\}, \emptyset, \{\phi\}$	$\{r/2.1, o/2.1\}, \{p/2.2\}, \{o/2.2\}$
Z/b	$\{\phi\}, \emptyset, \{\sigma, \tau\}$	$\{o/2.2\}, \{p/2.2\}, \{r/2.1, o/2.1\}$
Z/c	$\emptyset, \{\phi\}, \emptyset$	$\emptyset, \{p/2.2, o/2.2\}, \emptyset$
W/d	$\{\sigma\}, \{\tau\}, \{\rho\}$	$\{p/2.2\}, \{p/2.1, p/2.1\}, \emptyset$
U/f	$\emptyset, \{\pi, \phi\}, \{\sigma\}$	$\{s/2.1\}, \{r/2.2\}, \{t/2.1\}$

Corresponding similarity evaluation:

t'/t''	(n_c, l_c, m_c)	$\mathrm{sf}_c(t', t'')$	(n_r, l_r, m_r)	$\mathrm{sf}_r(t', t'')$	$\mathrm{sf}_o(t', t'')$
X/a	$(1, 2, 2)$	0.55	$(0, 3, 0)$	0.80	1.35
Y/b	$(1, 2, 0)$	0.68	$(0, 4, 0)$	0.83	1.51
Y/c	$(3, 0, 1)$	0.27	$(2, 1, 1)$	0.45	0.72
Z/b	$(1, 0, 2)$	0.29	$(1, 1, 2)$	0.45	0.74
Z/c	$(0, 1, 0)$	0.67	$(0, 2, 0)$	0.75	1.42
W/d	$(1, 1, 1)$	0.50	$(1, 2, 0)$	0.68	1.18
U/f	$(0, 2, 1)$	0.68	$(1, 1, 1)$	0.50	1.18

matching association between two sequences of atoms $< l'_1, \ldots, l'_k >$ and $< l''_1, \ldots, l''_k >$ is defined as $\theta_{<l'_1,\ldots,l'_k>/<l''_1,\ldots,l''_k>} = \bigcup_{i=1,\ldots,k} \theta_{l'_i/l''_i}$ iff $\forall i = 1, \ldots, k : l'_i$ and l''_i are built on the same predicate and $\theta_{l'_i/l''_i}$ is defined; it is undefined otherwise.

Two expressions are *compatible* if they can be mapped onto each other without yielding inconsistent term associations (i.e., a term in one of them cannot be associated to different terms in the other).

Definition 3.6. (Compatibility)
Two term associations θ' and θ'' are compatible iff $\theta' \cup \theta''$ is consistent. Two atoms, or two sequences of atoms, are *compatible* iff their matching association is defined and consistent. Two clauses are *compatible* iff their heads are compatible.

3.2.1. Star Similarity

Intuitively, the star of an atom depicts 'in breadth' how it relates to the rest of the formula.

Definition 3.7. (Star)
The *star* of an atom $l_{\bar{i}}$ in the body of a clause $C = l_0$:- $l_1, \ldots, l_n$ is the multiset

$$\{p/k \quad | \quad \exists i, 1 \leq i \leq n, i \neq \bar{i} : l_i = p(t_1, \ldots, t_k), k > 1, terms(l_i) \cap terms(l_{\bar{i}}) \neq \emptyset\}$$

It includes the predicates on which are built other atoms in the body of the clause that have some term in common with the given atom. Multisets are needed since a predicate can have multiple instantiations among the considered atoms.

Definition 3.8. (Star similarity)
The *star similarity* $\mathrm{sf}_s(l', l'')$ between two compatible atoms l' and l'' having stars S' and S'', respectively, is computed as

$$\mathrm{sf}_s(l', l'') = \mathrm{sf}(n_s, l_s, m_s) + C^s(\{\mathrm{sf}_o(t', t'')\}_{t'/t'' \in \theta_{l'/l''}}) \tag{3}$$

where $n_s = |S' \setminus S''|$ (*star residual* of l' wrt l''), $l_s = |S' \cap S''|$, $m_s = |S'' \setminus S'|$ (*star residual* of l'' wrt l') and C^s is a composition function (e.g., the average).

It is computed based on the number of common and different elements in each of the two stars[5]. Since this value refers only to atom-related information, and does not take into account the similarity of the involved terms, an overall more adequate evaluation of similarity between l' and l'' can be obtained by taking into account also the object similarity values for all pairs of terms included in the matching association $\theta_{l'/l''}$. In case sf_o ranges in $]0, 2[$ it ranges in $]0, 3[$, but can obviously be normalized to $]0, 1[$ if needed.

Example 3.3. Table 2 reports the stars for a sample of atoms in C and E and some comparisons between them.

3.2.2. Clauses as Stratified Graphs

Then, we can note that any first-order logic formula can be represented as a graph in which atoms are the nodes, and edges connect two nodes *iff* they are related in some way[6]. From such an equivalence between formulæ structures and graphs, it follows that a comparison between two formulæ to assess their structural similarity corresponds to the computation of (sub-)graph homomorphisms, a problem known to be NP-hard due to the possibility of mapping a (sub-)graph onto another in many different ways. As a consequence, we are interested in heuristics that can give significant hints on the structure overlapping between two formulæ with little computational effort. Indeed, leveraging on the fact that clauses are made up by just a single atom in the head and a conjunction of atoms in the body, we can exploit a graph representation that is easier than that for general formulæ, as described in the following. In particular, we will deal with *linked* clauses only (i.e. clauses whose associated graph is connected), and will build the graph based on a simple (as to the details it expresses about the fomula), yet powerful (as regards the information it conveys) feature, that is term sharing between couples of atoms.

[5]Intuitively, $S' \setminus S''$ are the occurrences of relations l' is involved in C' but l'' is not in C''; $S' \cap S''$ are the relations in which both l' in C' and l'' in C'' are involved; $S'' \setminus S'$ are the occurrences of relations l'' is involved in C'' but l' is not in C'.
[6]The actual definition of the graph can be made more or less complex, adding or deleting edges or introducing labels for them, in order to represent different details according to its intended use, up to a complete translation of all information conveyed by the formula.

Table 2. Star Similarity

C		E	
l'	S'	l''	S''
$p(X,Y)$	$\{p/2, p/2, r/2, o/2\}$	$p(a,b)$	$\{p/2, p/2, r/2, o/2\}$
$p(X,Z)$	$\{p/2, p/2, o/2\}$	$p(a,c)$	$\{p/2, p/2, o/2\}$
$p(W,X)$	$\{p/2, p/2, q/2\}$	$p(d,a)$	$\{p/2, p/2, q/2\}$
$r(Y,U)$	$\{p/2, o/2, s/2\}$	$r(b,f)$	$\{p/2, o/2, t/2\}$
$o(Y,Z)$	$\{p/2, p/2, r/2\}$	$o(b,c)$	$\{p/2, p/2, r/2\}$
$q(W,W)$	$\{p/2\}$	$q(d,e)$	$\{p/2\}$
$s(U,V)$	$\{r/2\}$	$t(f,g)$	$\{r/2\}$

Some comparisons:

l'	l''	$(S' \setminus S''), (S' \cap S''), (S'' \setminus S')$	(n_s, l_s, m_s)	$\mathrm{sf}(n_s, l_s, m_s)$
$p(X,Y)$	$p(a,b)$	$\emptyset, \{p/2, p/2, r/2, o/2\}, \emptyset$	$(0,4,0)$	0.83
$p(X,Y)$	$p(a,c)$	$\{r/2\}, \{p/2, p/2, o/2\}, \emptyset$	$(1,3,0)$	0.73
$p(X,Z)$	$p(a,c)$	$\emptyset, \{p/2, p/2, o/2\}, \emptyset$	$(0,3,0)$	0.80
$p(X,Z)$	$p(a,b)$	$\emptyset, \{p/2, p/2, o/2\}, \{r/2\}$	$(0,3,1)$	0.73
$p(W,X)$	$p(d,a)$	$\emptyset, \{p/2, p/2, q/2\}, \emptyset$	$(0,3,0)$	0.80
$r(Y,U)$	$r(b,f)$	$\{s/2\}, \{p/2, o/2\}, \{t/2\}$	$(1,2,1)$	0.60
$o(Y,Z)$	$o(b,c)$	$\emptyset, \{p/2, p/2, r/2\}, \emptyset$	$(0,3,0)$	0.80

Corresponding similarity evaluation ($C^s = avg$):

l'	l''	sf_o		$\mathrm{sf}_s(l', l'')$
$p(X,Y)$	$p(a,b)$	$\mathrm{sf}_o(X,a) = 1.35$	$\mathrm{sf}_o(Y,b) = 1.51$	2.26
$p(X,Y)$	$p(a,c)$	$\mathrm{sf}_o(X,a) = 1.35$	$\mathrm{sf}_o(Y,c) = 0.72$	1.77
$p(X,Z)$	$p(a,c)$	$\mathrm{sf}_o(X,a) = 1.35$	$\mathrm{sf}_o(Z,c) = 1.42$	2.19
$p(X,Z)$	$p(a,b)$	$\mathrm{sf}_o(X,a) = 1.35$	$\mathrm{sf}_o(Z,b) = 0.74$	1.78
$p(W,X)$	$p(d,a)$	$\mathrm{sf}_o(W,d) = 1.18$	$\mathrm{sf}_o(X,a) = 1.35$	2.07
$r(Y,U)$	$r(b,f)$	$\mathrm{sf}_o(Y,b) = 1.51$	$\mathrm{sf}_o(U,f) = 1.18$	1.95
$o(Y,Z)$	$o(b,c)$	$\mathrm{sf}_o(Y,b) = 1.51$	$\mathrm{sf}_o(Z,c) = 1.42$	2.27

Definition 3.9. Given a clause C, its *associated graph* $G_C = (V, E)$ is defined as

- $V = \{l_0\} \cup \{l_i | i \in \{1, \ldots, n\}, l_i$ built on k-ary predicate, $k > 1\}$ and
- $E \subseteq \{(a_1, a_2) \in V \times V \mid terms(a_1) \cap terms(a_2) \neq \emptyset\}$

where the edges to be included in E are chosen according to Algorithm 1.

Algorithm 1 builds a Directed Acyclic Graph (DAG), *stratified* (i.e., with the set of nodes partitioned) in such a way that, for any fixed stratum (element of the partition, also called *level*), all the incoming edges come from nodes in a single (different) stratum, and all the outcoming edges reach nodes in a single stratum (different from both the previous ones), as follows. The head is the only node at level 0 (first element of the partition). Then, each successive level (element of the partition) includes new nodes

Algorithm 1 Construction of the graph associated to C

Require: $C = l_0$:- $l_1, \ldots, l_n$: Clause
  $i \leftarrow 0$; $Level_0 \leftarrow \{l_0\}$; $E \leftarrow \emptyset$; $Atoms \leftarrow \{l_1, \ldots, l_n\}$
  while $Atoms \neq \emptyset$ **do**
    $i \leftarrow i + 1$
    $Level_i \leftarrow \{l \in Atoms \mid \exists l' \in Level_{i-1} : terms(l) \cap terms(l') \neq \emptyset\}$
    $E \leftarrow E \cup \{(l', l'') \mid l' \in Level_{i-1}, l'' \in Level_i, terms(l') \cap terms(l'') \neq \emptyset\}$
    $Atoms \leftarrow Atoms \setminus Level_i$
  end while
  return $G = (\bigcup_i Level_i, E)$: graph associated to C

(not present in previous levels) that have at least one term in common with nodes in the previous level. In particular, each node (atom) in the new level has an incoming edge from each node (atom) in the previous level having some argument (term) in common with it.

Example 3.4. Figure 1 shows the graphs associated to clauses C and D. Let us build the graph $G_C = (V, E)$. We have
$V = \{h(X)\} \cup \{p(X,Y), p(X,Z), p(W,X), r(Y,U), o(Y,Z), q(W,W), s(U,V)\}$
$E = Level_0 \cup Level_1 \cup Level_2 \cup Level_3$, where:

- The head represents the 0-level of the stratification:
  $Level_0 = \{h(X)\}$.
- Then directed edges may be introduced from $h(X)$ to $p(X,Y)$, $p(X,Z)$ and $p(W,X)$, that are the only atoms having X as an argument, which yields level 1 of the term stratification:
  $Level_1 = \{p(X,Y), p(X,Z), p(W,X)\}$.
- Now the next level can be built, adding directed edges from atoms in level 1 to the atoms not yet considered that share a variable with them:
  - $r(Y,U)$: end of an edge starting from $p(X,Y)$;
  - $o(Y,Z)$: end of edges starting from $p(X,Y)$ and $p(X,Z)$; and
  - $q(W,W)$: end of an edge starting from $p(W,X)$.

  $Level_2 = \{r(Y,U), o(Y,Z), q(W,W)\}$.
- The third and last level of the graph includes the only remaining atom, $s(U,V)$, having an incoming edge from $r(Y,U)$:
  $Level_3 = \{s(U,V)\}$.

Similarly for the graph G_D.

Theorem 3.1. Given a clause C and its associated graph G, any term appearing in C appears in at most two adjacent levels of G.

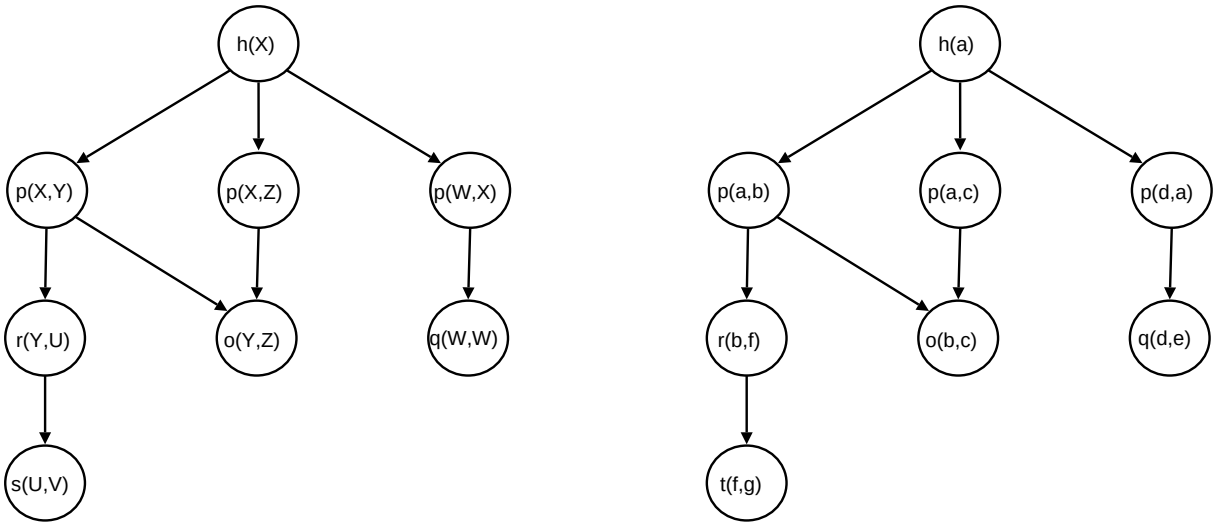


Figure 1. Graphs associated to clauses C (on the left) and D (on the right)

Proof:
Any pair of adjacent levels in G obviously share terms, being by construction (according to Algorithm 1) term sharing the condition for adding a directed edge between two nodes (atoms) in order to build a new level. Now, suppose (*ad absurdum*) that there are two atoms l' and l'' sharing a term t, such that l' appears in G at level i and l'' appears in G at level $i + j$ with $j > 1$. Then, by construction, l'' should have been included in level $i + 1$ according to Algorithm 1, and thus it should not be considered when building next levels, which contradicts the hypothesis. □

Each level can be divided into two subsets: the nodes introducing new terms with respect to the previous levels and those having as arguments only terms already introduced (in this latter case, Theorem 3.1 ensures that all such terms appear only in the previous level). Note that, as a further consequence, the stratification of the atoms straightforwardly induces a corresponding stratification of the terms (i.e., objects) that appear in the clause, based on their minimum distance from the head (intended as the number of atoms one has to traverse before encountering that term as an argument). Let us call this *stratification signature* of the clause: it allows to limit the possible binding between termss in two formulæ, since only terms in the same level can be mapped onto each other.

Example 3.5. The stratification signatures of C and E are, respectively:

$$< \{X\}, \{Y, Z, W\}, \{U\}, \{V\} > \qquad \text{and} \qquad < \{a\}, \{b, c, d\}, \{e, f\}, \{g\} >.$$

This means that: X *must* be associated to a; Y, Z, W can be associated only to b, c, d; U can be associated only to either e or f; V should be associated to g, but since the predicates they appear in are different in the two clauses (s in C, t in D) such a binding is not valid and hence they can be ignored in the comparison.

3.2.3. Path Similarity

The presence of a single atom in the clause head is the leverage to have a unique starting point for building the associated graph, which also gives a unique access point for traversing it according to precise directions represented the directed edges. As a consequence, when comparing two compatible clauses, since their heads must necessarily match to each other (being the only head atom in the clause and being compatible by hypothesis), having such a well-defined perspective among the many possible on their structure allows to significantly reduce indeterminacy. Specifically, all possible paths starting from the head and reaching *leaf* nodes (those with no outcoming edges) are univoquely determined. Thus, they can be interpreted as the basic components of the overall structure of the clause, and be exploited instead of single atoms when checking similarity between clauses. This gives a leverage for significantly reducing the amount of indeterminacy in the comparison. Intuitively, a path in a clause depicts 'in depth' a given portion of the relations it describes.

Definition 3.10. (Path similarity)
Given two compatible clauses C' and C'' with associated graphs $G_{C'}$ and $G_{C''}$ respectively, and two paths $p' =< l'_0, l'_1, \ldots, l'_{n'} >$ in $G_{C'}$ and $p'' =< l''_0, l''_1, \ldots, l''_{n''} >$ in $G_{C''}$. The *intersection* between p' and p'' is defined as

$$p' \cap p'' = (p_1, p_2) = (< l'_1, \ldots, l'_k >, < l''_1, \ldots, l''_k >) \text{ s.t.}$$
$$0 \leq k \leq \min(n', n'') \wedge \theta_{<l'_0,\ldots,l'_k>/<l''_0,\ldots,l''_k>} \text{ is defined and consistent} \wedge$$
$$(k = n' \vee k = n'' \vee \theta_{<l'_0,\ldots,l'_{k+1}>/<l''_0,\ldots,l''_{k+1}>} \text{ is undefined or inconsistent})$$

and their *differences* are defined as the trailing parts:

$$p' \setminus p'' =< l'_{k+1}, \ldots, l'_{n'} > \qquad p'' \setminus p' =< l''_{k+1}, \ldots, l''_{n''} >$$

Then, the *path similarity* between p' and p'' is computed as

$$\mathrm{sf}_p(p', p'') = \mathrm{sf}(n_p, l_p, m_p) + C^p(\{\mathrm{sf}_s(l'_i, l''_i)\}_{i=1,\ldots,k}) \tag{4}$$

where $n_p = |p' \setminus p''| = n' - k$, $l_p = |p_1| = |p_2| = k$, $m_p = |p'' \setminus p'| = n'' - k$ and C^p is a composition function (e.g., the average)[7].

It is based on the pair of longest compatible initial subsequences of the two paths (excluding the heads), and takes into account also the star similarity values for all pairs of atoms associated by those subsequences. Note that the intersection could be empty (in case $\theta_{<l'_0,l'_1>/<l''_0,l''_1>}$ is undefined or inconsistent), and that either difference could be empty as well. In case sf_s ranges in $]0, 3[$ it ranges in $]0, 4[$, but can obviously be normalized to $]0, 1[$ if needed.

Example 3.6. Let us now find all paths in C and E:

Path No.	C	E
1.	$< p(X,Y), r(Y,U), s(U,V) >$	$< p(a,b), r(b,f), t(f,g) >$
2.	$< p(X,Y), o(Y,Z) >$	$< p(a,b), o(b,c) >$
3.	$< p(X,Z), o(Y,Z) >$	$< p(a,c), o(b,c) >$
4.	$< p(W,X), q(W,W) >$	$< p(d,a), q(d,e) >$

[7] Intuitively, n_p is the length of the trail incompatible sequence of p' wrt p'' (*path residual* of p' wrt p''); l_p is the length of the maximum compatible initial sequence of p' and p''; m_p] is the length of the trail incompatible sequence of p'' wrt p' (*path residual* of p'' wrt p').

Table 3. Path Similarity ($C^p = avg$)

p' / p''	$p' \cap p''$	$p' \setminus p''$ / $p'' \setminus p'$	$\theta_{p' \cap p''}$	(n_p, l_p, m_p) / $\mathrm{sf}_p(p', p'')$
C.1	$< p(X,Y), r(Y,U) >$	$< s(U,V) >$	$\{X/a, Y/b, U/f\}$	$(1,2,1)$
E.1	$< p(a,b), r(b,f) >$	$< t(f,g) >$		2.71
C.1	$< p(X,Y) >$	$< r(Y,U), s(U,V) >$	$\{X/a, Y/b\}$	$(2,1,1)$
E.2	$< p(a,b) >$	$< o(b,c) >$		2.71
C.1	$< p(X,Y) >$	$< r(Y,U), s(U,V) >$	$\{X/a, Y/c\}$	$(2,1,1)$
E.3	$< p(a,c) >$	$< o(b,c) >$		2.22
C.2	$< p(X,Y) >$	$< o(Y,Z) >$	$\{X/a, Y/b\}$	$(1,1,2)$
E.1	$< p(a,b) >$	$< r(b,f), t(f,g) >$		2.71
C.2	$< p(X,Y), o(Y,Z) >$	$<>$	$\{X/a, Y/b, Z/c\}$	$(0,2,0)$
E.2	$< p(a,b), o(b,c) >$	$<>$		3.02
C.2	$< p(X,Y) >$	$< o(Y,Z) >$	$\{X/a, Y/c\}$	$(1,1,1)$
E.3	$< p(a,c) >$	$< o(b,c) >$		2.27
C.3	$< p(X,Z) >$	$< o(Y,Z) >$	$\{X/a, Z/b\}$	$(1,1,2)$
E.1	$< p(a,b) >$	$< r(b,f), t(f,g) >$		2.23
C.3	$< p(X,Z) >$	$< o(Y,Z) >$	$\{X/a, Z/b\}$	$(1,1,1)$
E.2	$< p(a,b) >$	$< o(b,c) >$		2.28
C.3	$< p(X,Z), o(Y,Z) >$	$<>$	$\{X/a, Z/c, Y/b\}$	$(0,2,0)$
E.3	$< p(a,c), o(b,c) >$	$<>$		2.98
C.4	$< p(W,X) >$	$< q(W,W) >$	$\{W/d, X/a\}$	$(1,1,1)$
E.4	$< p(d,a) >$	$< q(d,e) >$		2.57

All other intersections are empty, and hence ignored.

As a sample comparison, consider $C.1$ and $E.1$: the longest compatible initial subsequence compatible both for predicates and as to overall term associations is made up by their first two atoms; then, being the third atoms different by predicate, the rests of the sequences (actually, in this case just the third atom itself) belong to the residuals. The list of path similarities is reported in Table 3 (note that similarities C.1-E.4 and C.2-E.4 are empty since the head forces variable X to be bound to constant a).

3.3. Clause Similarity

The overall similarity between two (tuples of) terms reported in the head atoms of two compatible clauses, according to their description reported in the respective bodies, can be computed based on their generalization, that informally represents what the two clauses have in common, where the notion of generality depends on the generalization model adopted. In particular, one would like to exploit their *least general generalization*, in order to preserve in the generalization as much information as possible of the original clauses. Unfortunately, such a generalization is not easy to find: either classical θ-subsumption is used as a generalization model, and then one can compute Plotkin's least general generalization [23], at the expenses of some undesirable side-effects concerning the need of computing its reduced equivalent

(and also of some counter-intuitive aspects of the result), or, as most ILP learners do, one requires the generalization to be a subset of the clauses to be generalized. In the latter option, that we choose for the rest of the work, the θ_{OI} generalization model [9], based on the Object Identity assumption, represents a supporting framework with solid theoretical foundations to be exploited.

Definition 3.11. (formulæ similitudo)
Given two clauses C' and C'' with heads l'_0 and l''_0 respectively, call $C = l_0$:- $l_1, \ldots, l_k$ their least general generalization, and consider the substitutions θ' and θ'' such that $l_0\theta' = l'_0$, $l_0\theta'' = l''_0$ and $\forall i = 1, \ldots, k : l_i\theta' = l'_i \in body(C')$ and $l_i\theta'' = l''_i \in body(C'')$, respectively. The formula for assessing the overall similarity between C' and C'', called *formulæ similitudo* and denoted *fs*, is the following:

$$\mathrm{fs}(C', C'') = \mathrm{sf}(n, l, m) \cdot \mathrm{sf}(n_o, l_o, m_o) + C^c(\{\mathrm{sf}_s(l'_i, l''_i)\}_{i=1,\ldots,k})$$

where $n = |body(C')| - |body(C)|$, $l = |body(C)| = k$, $m = |body(C'')| - |body(C)|$; $n_o = |terms(C')| - |terms(C)|$, $l_o = |terms(C)|$, $m_o = |terms(C'')| - |terms(C))|$; and C^c is a composition function (e.g., the average).

It is based on the amounts of common and different atoms[8] and terms[9], and takes into account also the star similarity values for all pairs of atoms associated by the least general generalization. In case sf_s ranges in $]0, 2[$ it ranges in $]0, 3[$, but can obviously be normalized to $]0, 1[$ if needed. This function evaluates the similarity of two clauses according to the composite similarity of a maximal subset of their atoms that can be put in correspondence (which includes both structural and object similarity), smoothed by adding the overall similarity in the number of overlapping and different atoms and terms between the two (whose weight in the final evaluation should not overwhelm the similarity coming from the detailed comparisons, hence the multiplication).

Example 3.7. The least general generalization between C and D is

$$H(X) : -p(X, Y), p(X, Z), p(W, X), o(Y, Z), r(Y, U)$$

with associations $\{X/a, Y/b, Z/c, W/d, U/f\}$. The corresponding similarity, for $C^c = avg$ is

$$\mathrm{fs}(C, D) = \mathrm{sf}(2, 5, 2) \cdot \mathrm{sf}(1, 5, 2) + avg(\{2.26, 2.19, 2.07, 2.27, 1.95\}) = 0.67 \cdot 0.71 + 10.74/5 = 2.62$$

or, normalized to $]0, 1[$, 0.87.

[8] Intuitively, n represents how many atoms in C' are not covered by its least general generalization with respect to C'' (*clause residual* of C' wrt C''); l is the maximal number of atoms that can be put in correspondence between C' and C'' according to their least general generalization; m represents how many atoms in C'' are not covered by its least general generalization with respect to C' (*clause residual* of C'' wrt C').

[9] Intuitively, n_o represents how many terms in C' are not associated by its least general generalization to terms in C'' (*object residual* of C' wrt C''); l_o is the maximal number of terms that can be put in correspondence in C' and C'' as associated by their least general generalization; m_o represents how many terms in C'' are not associated by its least general generalization to terms in C' (*object residual* of C'' wrt C').

Algorithm 2 Similarity-based generalization

Require: $C' = l'_0$:- $l'_1, \ldots, l'_{n'}$, $C'' = l''_0$:- $l''_1, \ldots, l''_{n''}$: Clauses
  $P_{C'} \leftarrow paths(C')$; $P_{C''} \leftarrow paths(C'')$;
  $P \leftarrow \{(p_{C'}, p_{C''}) \in P_{C'} \times P_{C''} \mid p_{C'} \cap p_{C''} \neq (<>, <>)\}$;
  $G \leftarrow \emptyset$; $\theta \leftarrow \theta_{l'_0/l''_0}$
  while $P \neq \emptyset$ **do**
    $(\overline{p}_{C'}, \overline{p}_{C''}) \leftarrow \mathrm{argmax}_{(p_{C'}, p_{C''}) \in P}(\mathrm{sf}_p(p_{C'}, p_{C''}))$
    $P \leftarrow P \setminus \{(\overline{p}_{C'}, \overline{p}_{C''})\}$
    $(q_{C'}, q_{C''}) \leftarrow \overline{p}_{C'} \cap \overline{p}_{C''}$
    if $\theta_{q_{C'}/q_{C''}}$ compatible with θ **then**
      $G \leftarrow G \cup q_{C'}$; $\theta \leftarrow \theta \cup \theta_{q_{C'}/q_{C''}}$
    end if
  end while
  return l_0 :- G: generalization between C' and C''; θ: matching association

3.4. Exploitation in Clause Generalization

Given a Horn clause, instead of working on its single atoms, the paths univoquely determined according to the technique proposed in Section 2 can be safely (i.e., without loss of generality) considered as its basic components. Indeed, considering paths instead of single atoms allows the generalizer to have more information about the pieces to be composed for building up the generalization [12], and hence gives it the opportunity of taking better choices, that will hopefully lead more quickly towards the proper sub-formulæ associations.

Once the set of paths for the two clauses C' and C'' to be generalized are obtained, each pair of paths, in which the former is taken from C' and the latter from C'', can be compared for assessing their similarity. Of course, pairs that show a higher similarity are more likely to correspond to each other, and hence their common part (i.e., their *intersection*) should be taken as part of the generalization. Thus, a generalization can be computed according to Algorithm 2. Starting from the empty generalization, the path couples are considered in turn by decreasing similarity, starting from the top and going down the ranking, and their intersection is added to the partial generalization generated thus far whenever it is compatible with it (intersections that are not compatible with the current partial generalization are just skipped). Further generalizations can then be obtained through backtracking. Since this procedure takes into account the most similar sub-parts of the clauses first, it can optionally be stopped when some threshold is reached, even before complete analysis of the path list, in order to cut the generalization and prevent it from becoming too specific, still ensuring that only the less significant similarities are dropped.

Example 3.8. The path intersection with highest similarity value is C.2/E.2, and hence the first partial generalization becomes $\{p(X,Y), o(Y,Z)\}$, with associations $\{X/a, Y/b, Z/c\}$. Then C.3/E.3 is considered, whose associations are compatible with the current ones, so it contributes with $\{p(X,Z)\}$ to the generalization (there are no new associations). Then comes C.1/E.1, that being compatible extends the generalization by adding $\{r(Y,U)\}$ and the association with $\{U/f\}$. It is the turn of C.1/E.2 and then of C.2/E.1, that are compatible but redundand, and hence do not add anything to the current generalization

(nor to the associations)[10]. Then C.4/E.4 is considered, that is compatible and extends with $\{p(W, X)\}$ and $\{W/d\}$ the current generalization and associations, respectively. Lastly C.3/E.2, C.2/E.3, C.3/E.1 and C.1/E.3 are considered, but discarded because of their associations being incompatible. The final generalization is $H(X) : -p(X, Y), p(X, Z), p(W, X), o(Y, Z), r(Y, U)$, which is also the least general one (as previously stated), with overall associations $\{X/a, Y/b, Z/c, W/d, U/f\}$.

4. Discussion

The similarity framework for Horn Clauses presented in this paper is made up of a set of similarity parameters and a similarity function based on them, plus a set of criteria that focus on particular clause components to tackle the problem of indeterminacy while still preserving a considerable amount of information about the description structures. It is important to stress the point that the similarity framework is syntax-based, and hence totally general, since it does not assume domain-related knowledge for assessing the similarity degree between two descriptions.

As to the similarity parameters, they are standard and widely accepted in the past literature of the field. As to the similarity function, a novel one was developed to avoid cases in which other standard functions proposed and accepted in the literature are not able to distinguish between different cases (e.g., when $l = 0$ for different n and m values), or might have definition problems (e.g., when $n = l = m = 0$ a division by zero would be raised) or could assign the full similarity value 1 just because the syntactic structure of two expressions is the same (whereas we believe that equality should be reserved to exact identification between the two).

As to the criteria, they refer to increasingly complex description components, from terms to atoms to groups of atoms to whole clauses. More precisely, the similarity of each component type is based on the similarity of simpler components (only), so that no recursion nor indeterminacy can be present. It should be noted that no single criterion is by itself neatly discriminant, but their cooperation succeeds in assigning sensible similarity values to the various kinds of components, and in distributing on each kind of component a proper portion of the overall similarity, so that the difference becomes ever clearer as long as they are composed one ontop the previous ones. This makes the proposed approach robust to lacks of information due to some of the criteria. For instance, in cases in which the clause head does not provide helpful information on the structure (e.g., when it includes all, or almost all the terms as arguments, since the whole description in the body must be evaluated without focussing on any particular object), only the last step is missing contribution, and the similarity can still benefit from the information coming from all the other parameters in order to identify good matching sub-formula candidates.

4.1. Computational Complexity Issues

As to the computational complexity, the cost for computing the formula is clearly constant. Then, note that each term (respectively atom, path) in one clause must be compared with each term (respectively atom, path) in the other by means of the formula; assuming that the maximum number of terms (respectively atoms, paths) in either of the two clauses is n, we have less than n^2 applications of the formula, and hence a quadratic complexity in the number of terms (respectively atoms, paths) to be compared.

[10] Actually, C.1/E.1, C.1/E.2 and C.2/E.1 have the same value, but being all compatible among each other the actual ordering, which depends on the sort algorithm used, is not significant to the final outcome.

Now, since the number of terms and atoms is fixed in the clauses, the only thing still to evaluate is the number of paths. Note that the DAG resulting from the linkedness-based stratification of the atoms in a clause is not necessarily a hierarchy (a tree), but more generally a heterarchy (i.e., each node may have many parents, although loops are not allowed in the structure), where each edge can connect nodes only at adjacent strata (see discussion above). In the worst case, every node at each level is connected to every node at the levels just above and below, in which case the overall number of paths is equal to the product of the number of nodes at each level. For such a product to be largest, the nodes should be equally distributed among levels (each atom appears just once in the DAG). Let us call m the number of nodes at each level: hence, since each path is made up of one node from each level, its maximum length is the DAG depth, equal to n/m, and the associated number of paths is $m^{n/m}$ (number of nodes at each level, multiplied by itself so many times as the number of levels), i.e. it is exponential in the DAG depth. Indeed, in the worst case of the atoms being absolutely equally distributed both *in* and *among* levels $m = \sqrt{n} \Rightarrow m^{n/m} = \sqrt{n}^{n/\sqrt{n}} = \sqrt{n}^{\sqrt{n}}$. However, in more realistic cases, in which atoms are irregularly distributed in breadth and/or depth, and adjacent levels are not completely connected, the complexity moves towards the two extremes $m = 1 \Rightarrow m^{n/m} = 1^n = 1$ and $m = n \Rightarrow m^{n/m} = n^1 = n$.

4.2. Related Works

Despite of a large number of possible distance measures developed for attribute-value representations [17], few works are present in the literature that face the problem of defining similarity or distance measures for first-order descriptions. A work in this direction was carried out in [11], where a distance measure between structural symbolic descriptions is proposed, based on probability theory applied to the formula components. Compared to that proposal, our function preserves the probabilistic $]0, 1[$ range. However, rather than resorting to probabilistic models that might be hard to understand and justify, it is based on heuristic and intuitive premises, which yields a number of advantages: the underlying principles are easier to understand, it is far easier to compute and does not require the assumptions and simplifying hypotheses (statistical independence, mutual exclusion) made in [11] to ease the probability handling, no *a-priori* knowledge of the representation language is required (such as the type domains that in [11] allow the distance computation between references). It does not require the user to set weights on the single predicates' importance, which is a difficult task even for trained people, but limits the human intervention to a setting of the relative importance of the two descriptions involved, that can be easily understood. It is not based on the presence of 'mandatory' relations, like for the $G1$ subclause in [11]. Last but not least, its application range is not limited to a Model-Observation matching comparison, but can be properly weighted so that both descriptions have the same dignity in the similarity evaluation.

Many systems, in the field of supervised learning, prove the importance of a distance measure. For instance, *KGB* [3] uses a similarity function specifically designed for FOL languages and parameterized by the user, in order to guide generalization; our ideas of characteristic and relational similarity are very close to those in [3], but then the actual similarity computation is much more straightforward. While *KGB* cannot handle negative information possibly present in the clauses, our approach can be easily extended to deal with them, by applying Algorithm 1 only to positive atoms and just ignoring in Algorithm 2 the path pairs whose corresponding associations are inconsistent with the negative atoms.

The k-Nearest Neighbor classifier *RIBL* [8] is based on a FOL similarity function that is a modified version of that proposed in [3]. The basic idea of the measure used in *RIBL* is that objects are described by values (e.g., size and position) and their relation to other objects. Their similarity depends on the

similarity of their attributes' values (the similarity of their size) and the similarity of the objects related to them. The similarity of the related objects in turn depends on the attribute values of these objects and their relation to other objects and so on. Such a propagation poses the problem of indeterminacy in associations, that our technique avoids thanks to the different structural approach. Although the indeterminacy problem could be handled by using some CSP technique (e.g., [32]), there is no reason for not exploiting the information provided by the clause heads, since (as already pointed out) they are purposely exploited for highlighting the focus-of-attention, and hence *must* match. The basic idea is to compute the similarity between two objects by considering the immediate neighborhood of the objects. For instance, the similarity between two identifiers is determined by the similarity between the set of facts where these identifiers occur. Also, the similarity between two facts is determined by the similarity between their arguments. If some of these arguments are identifiers again, one can get a cycle. Therefore, a depth parameter is introduced and similarity is only computed up to this depth. *RISE* [7] combines a rule classifier with a k-NN: when the instance to classify is not covered by any rule, the distance of the instance to the different rules is evaluated and the instance is classified according to the majority vote of its "neighbor" rules .

While the distances underlying *KBG* and *RIBL* or *RISE* are basically syntax-driven, [27] presents a discrimination-based approach for the induction of a distance on FOL examples, that mainly depends on the pattern discriminating the target concepts. k clauses are choosen and the truth value of the clause on the examples (whether the clause covers the example or not) are used as k features. In a second step, one uses a distance on the space $\{0,1\}^k$ of these features as a distance between the examples.

In [22] a distance between terms is proposed that treats terms as a hierarchy where the top of high-level structure (the main functor) is most important and the deeper nested subterms are less important. Then a distance is proposed between interpretations based on a level mapping, a function that maps every simple expression on a natural number. [24] presents a distance function between two atoms A and B based on the difference with their lgg. Such a distance consists of a pair whose first component is based on the differences between the functors on both terms. It is an extension of the notion of distance used in [22]: it is also defined for non-ground atoms and it introduces weights. The second component is based on the differences in occurrences of variables, it allows to differentiate distances in cases where the first component cannot. Then, this distance between atoms is used to compute distances between clauses.

More related to the generalization task, the approach proposed by Kodratoff [16] is based on just an evaluation of similarity between objects, whose overall most likely associations are then set so to maximize the global fitness; our proposal in some way extends those ideas by exploiting the structural features to progressively exclude impossible or less suitable terms associations.

Cluster analysis concerns the organization of a collection of unlabeled patterns into groups (clusters) of homogeneous elements based on their similarity. The similarity measure exploited to evaluate the distance between elements is responsible for the effectiveness of the clustering algorithms. Many research efforts on data representation, elements' similarity and grouping strategies have produced several successful clustering methods (see [15] for a survey). The classical strategies can be divided in bottom-up and top-down. In the former, each element of the dataset is considered as a cluster. Successively, the algorithm tries to group the clusters that are more similar according to the similarity measure. This step is performed until the number of clusters the user requires as a final result is reached, or the minimal similarity value among clusters is greater than a given threshold. In the latter approach, known as hierarchical clustering, at the beginning all the elements of the dataset form a unique cluster. Successively, the cluster is partitioned into clusters made up of elements that are more similar according to the similarity

measure. This step is performed until the number of clusters required by the user as a final result is reached.

A further classification is based on whether an element can be assigned (NotExclusive or Fuzzy Clustering) or not (Exclusive or Hard Clustering) to more than one cluster. Also the strategy exploited to partition the space is a criterion used to classify the clustering techniques: in Partitive Clustering a representative point (centroid, medoid, etc..) of the cluster in the space is chosen; Hierarchical Clustering produces a nested series of partitions by merging (Hierarchical Agglomerative) or splitting (Hierarchical Divisive) clusters, Density-based Clustering considers the density of the elements around a fixed point.

Closely related to data clustering is Conceptual Clustering, a Machine Learning paradigm for unsupervised classification which aims at generating a concept description for each generated class. In conceptual clustering both the inherent structure of the data and the description language, available to the learner, drive cluster formation. Thus, a concept (regularity) in the data could not be learned by the system if the description language is not powerful enough to describe that particular concept (regularity). This problem arises when the elements simultaneously describe several objects whose relational structures change from one element to the other. First-Order Logic representations allow to overcome these problems. However, most of the clustering algorithms and systems work on attribute-value representation (e.g., `CLUSTER/2` [20], `CLASSIT` [14], `COBWEB` [13]). Other systems such as `LABYRINTH` [29] can deal with structured objects exploiting a representation that is not powerful enough to express the dataset in a lot of domains. There are few systems that cluster examples represented in FOL (e.g., `AUTOCLASS-like` [25], `KBG` [2]), some of which still rely on propositional distance measures (e.g., `TIC` [4]).

4.3. Experimental evaluation

Some experiments were designed to check whether the proposed framework is actually able to give significant similarity hints when comparing two structures. All of them were run under WindowsXP Professional on a PC endowed with a 2.13 GHz Intel processor and 2GB RAM. For supervised learning tasks, 10-fold cross-validation was exploited to assess predictive accuracy. The 10 folds were created so that the distribution of examples from the 4 classes was uniform in each fold (each training and test set contained approximately 90% and 10%, respectively, of examples from each class).

Two real-world datasets, requiring first-order logic descriptions for capturing the complexity of the domain and concerning very different domains from each other, were identified for performing the experiments and ensure general applicability and performance of the proposed approach. Mutagenesis is a classical ILP dataset [28] representing 188 chemical compounds that are to be distinguished, according to their behavior, between those that are Active and those that are NonActive with respect to mutagenicity, for which regression-based techniques are not able to learn useful theories. The dataset is described with a total of 25917 atoms, for an average of nearly 138 atoms per description. In particular, we exploited an automatic discretization procedure [1] for turning numeric descriptors into symbolic ones, each corresponding to an interval of the original ranges. The other dataset concerns automatically generated layout descriptions of first pages of scientific papers, according to which identifying the papers' series and significant components. It contains 353 descriptions, belonging to 4 different classes: Elsevier journals, Springer-Verlag Lecture Notes series (SVLN), Journal of Machine Learning Research (JMLR) and Machine Learning Journal (MLJ). For each class, some layout components of interest are selected: Title, Abstract and Author for all classes, plus Keywords for JMLR and MLJ, and Logo and Keywords for Elsevier. The complexity of such a dataset is considerable, due to several aspects: the journals layout

styles are quite similar, so that it is not easy to grasp the difference when trying to group them in distinct classes; moreover, the 353 documents are described with a total of 67920 atoms, for an average of more than 192 atoms per description (some descriptions are made up of more than 400 atoms); last, the description is heavily based on a membership relation (*frame*) that increases indeterminacy.

The first question concerns whether the proposed similarity function is actually able to lead towards the identification of the proper sub-parts to be put in correspondence in the two descriptions under comparison. Since the 'correct' association is not known, this can be evaluated only indirectly. A way for doing this is evaluating the **compression factor** of the guided generalization, i.e. the portion of atoms in the clauses to be generalized that is preserved by the generalization, defined as the ratio between the length of the generalization over that of the shortest clause to be generalized: the higher such a value, the more confident one can be that the correct associations were provided by the similarity criteria and formula. Indeed, since it is usual in ILP systems that a generalization must be a subset of either clause to be generalized[11], the more atoms the generalization preserves from these clause, the less general it is. Of course, the more the difference in length between the two clauses to be generalized, the more indeterminacy is present, and hence the more difficult it is to identify the proper corresponding parts between them. Interestingly, on the document dataset the similarity-driven generalization preserved on average more than 90% atoms of the shortest clause, with a maximum of 99,48% (193 atoms out of 194, against an example of 247) and just 0,006 variance. As a consequence, one woud expect that the produced generalizations are least general ones or nearly so. Application of Tverski's similarity formula to the same setting always returned shorter generalizations than those obtained by using our formula.

The similarity-driven **generalization procedure** was compared to a previous non-guided procedure, embedded in the learning system INTHELEX [10]. Whenever the first generalization was not consistent with all past negative examples, the system was allowed to search for other ones on backtracking. On the Mutagenesis dataset, the proposed generalization technique reached a slightly better predictive accuracy (87%) than the non-guided version (86%) exploiting only 30% runtime. No backtracking was ever needed, against the 5815 of the non-guided version: this was a good hint that the similarity criteria, strategy and formula are actually able to lead the correct identification of corresponding sub-parts of the compounds descriptions. On the document dataset also F-measure (with parameter 1 in order to equally weight Precision and Recall) was evaluated, to ensure that the performance was balanced between positive and negative examples. Classification outcomes show that the similarity-driven version outperformed the classical one in all considered parameters: 70% runtime savings, higher quality theory (less clauses per definition, less exceptions), better learning behaviour (less generalizations/specializations needed), +1% average accuracy (98%) and +2% average F-measure (96%). For the understanding task, overall averages are 95% for accuracy (-1%), and 89% for F-measure (-2%), obtained in 32.14% (1 day 3 hours cumulative) time savings. It must be pointed out that the old procedure *failed* on fold 4 of label Keywords in class MLJ, running out of memory after various hours, so the averages are actually computed on the remaining 159 folds only.

The **clustering** task aims at grouping a set of items into homogeneous classes according to the similarity between their descriptions; if an intensional (e.g., first-order logic) description of each cluster is provided, it is defined *conceptual* clustering. We embedded the proposed similarity assessment technique into a classical K-means algorithm, exploiting *medoid* prototypes[12] instead of centroids, since

[11] Note that this is an odd assumption in general, but not in INTHELEX because of its enforcing the Object Identity assumption.
[12] The medoid of a cluster is defined as the observation in the cluster that has the minimum average distance from all the other members of the cluster.

first-order logic formulæ do not induce an euclidean space. The stop criterion was set as the moment in which a new iteration outputs a partition already seen in previous iterations. Since the class of each observation in the datasets is known, we provided the clustering procedure with the number of clusters to be obtained (4 in the case of documents and 2 for Mutagenesis), and then compared the results with the correct classes to assess their quality (*supervised clustering*). Each cluster was compared to its best-matching class, and their overlapping evaluated according to precision, recall and purity. In fact, for each cluster the precision-recall values were neatly high for one class and considerably low for all the others; moreover, and each cluster had a different best-matching class, so that the association and consequent evaluation became straightforward. Results for the document dataset revealed a 91.60% precision, 93.28% recall and 92.35% purity, indicating that the proposed method is highly effective since it is able to autonomously recognize the original classes. This is very encouraging, especially in the perspective of the representation-related difficulties. A comparison to other measures in the literature reports an improvement with respect to both Jaccard's, Tverski's and Dice's measures up to +5,48% for precision, up to + 8,05% for recall and up to + 2,83% for purity. Also on the Mutagenesis dataset the precision, recall and purity figures are very high (81.56%, 81.30% and 83.51%, respectively), considering that the state-of-the-art performance on accuracy (comparable to clustering purity) of supervised systems on this dataset is between 83% and 87%.

As to the exploitation of the proposed similarity technique to the **k-Nearest Neighbour** technique, k was set to the square root of the number of learning instances, i.e. 17. Note that the classification was a multi-class one, so (altough k is odd) the nearest neighbours are not bound to a binary classification and ties are possible. The results of the classification performance show an overall accuracy of 94.37%, which means that few documents were associated to the wrong class, and hence the distance technique is very good in identifying the proper similarity features within the given descriptions. Actually, very often in the correct cases not just the majority, but almost all of the nearest neighbors to the description to be classified were from the same class. Specifically, classes Elsevier and MLJ always have 100% accuracy, which means that the corresponding examples are quite significant and sharply distinguishable. Errors were concentrated in the SVLN and JMLR classes (where, however, high accuracy rates were reached: 89.70% and 90.03%, respectively), and reveal that MLJ is quite distinct from the other classes, while Elsevier, although well-recognizable in itself, is somehow in between JMLR and SVLN, which are also close to each other. Interestingly, mismatchings concerning Elsevier are unidirectional: some JMLR and SVLN are classified as Elsevier, but the *vice-versa* never happened; on the other hand, in the case of JMLR and SVLN it is bidirectional, suggesting a higher resemblance between the two.

5. Conclusions

The problem of indeterminacy in mapping a First-Order Logic formula onto another, due to the presence of relations, causes serious computational problems. Hence, many Artificial Intelligence tasks that are based on FOL would take advantage from techniques for the comparison and similarity assessment among (parts of) descriptions. This paper proposed a novel similarity framework for Horn clauses, on which the Logic Programming paradigm is founded, and comparison strategies for their progressively complex components (terms, atoms, sets of atoms). Experiments on real-world datasets prove the effectiveness of the proposal, and the efficiency of the corresponding implementation in many tasks: it helps inductive generalization to preserve common features of the descriptions to be generalized, leads super-

vised learning systems towards cleaner and more accurate theories dramatically reducing the runtime needed for building them, effectively groups unknown descriptions in consistent and homogeneous clusters, improves classification performance of instance-based techniques. The proposed similarity formula also outperforms state-of-the-art alternatives.

References

[1] Biba, M., Esposito, F., Ferilli, S., Mauro, N. D., Basile, T. M. A.: Unsupervised Discretization Using Kernel Density Estimation., in: Veloso [31], 696–701.

[2] Bisson, G.: Conceptual clustering in a first order logic representation, *ECAI '92: Proceedings of the 10th European conference on Artificial intelligence*, John Wiley & Sons, Inc., 1992.

[3] Bisson, G.: Learning in FOL with a similarity measure, *Proc. of AAAI-92* (W. Swartout, Ed.), 1992.

[4] Blockeel, H., Raedt, L. D., Ramon, J.: Top-down induction of clustering trees, *Proceedings of the 15th International Conference on Machine Learning* (J. Shavlik, Ed.), Morgan Kaufmann, 1998.

[5] Bock, H.: *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*, Springer-Verlag, 1999.

[6] Ceri, S., Gottlob, G., Tanca, L.: *Logic Programming and Databases*, Springer, 1990.

[7] Domingos, P.: Rule induction and instance-based learning: a unified approach, *Proc. of IJCAI-95*, Morgan Kaufmann, 1995.

[8] Emde, W., Wettschereck, D.: Relational instance based learning, *Proc. of ICML-96* (L. Saitta, Ed.), 1996.

[9] Esposito, F., Fanizzi, N., Ferilli, S., Semeraro, G.: A Generalization Model Based on OI-implication for Ideal Theory Refinement., *Fundam. Inform.*, **47**(1-2), 2001, 15–33.

[10] Esposito, F., Ferilli, S., Fanizzi, N., Basile, T., Mauro, N. D.: Incremental Multistrategy Learning for Document Processing, *Applied Artificial Intelligence Journal*, **17**(8/9), 2003, 859–883.

[11] Esposito, F., Malerba, D., Semeraro, G.: Classification in Noisy Environments Using a Distance Measure Between Structural Symbolic Descriptions, *IEEE Transactions on PAMI*, **14**(3), 1992, 390–402.

[12] Ferilli, S., Basile, T., Mauro, N. D., Biba, M., Esposito, F.: Similarity-Guided Clause Generalization, *AI*IA-2007: Artificial Intelligence and Human-Oriented Computing* (R. Basili, M. Pazienza, Eds.), 4733, Springer, 2007.

[13] Fisher, D. H.: Knowledge Acquisition Via Incremental Conceptual Clustering, *Machine Learning*, **2**(2), 1987, 139–172.

[14] Gennari, J. H., Langley, P., Fisher, D.: Models of incremental concept formation, *Artificial Intelligence*, **40**(1-3), 1989, 11–61.

[15] Jain, A. K., Murty, M. N., Flynn, P. J.: Data clustering: a review, *ACM Computing Surveys*, **31**(3), 1999, 264–323.

[16] Kodratoff, Y., Ganascia, J.-G.: Improving the Generalization Step in Learning, in: *Machine Learning: An Artificial Intelligence Approach: Volume II* (R. S. Michalski, J. G. Carbonell, T. M. Mitchell, Eds.), Kaufmann, Los Altos, CA, 1986, 215–244.

[17] Li, M., Chen, X., Li, X., Ma, B., Vitanyi, P.: The similarity metric, 2003.

[18] Lin, D.: An information-theoretic definition of similarity, *Proc. 15th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1998.

[19] Lloyd, J. W.: *Foundations of logic programming; (2nd extended ed.)*, Springer-Verlag New York, Inc., New York, NY, USA, 1987, ISBN 3-540-18199-7.

[20] Michalski, R. S., Stepp, R. E.: Learning from Observation: Conceptual Clustering, in: *Machine Learning: An Artificial Intelligence Approach* (R. S. Michalski, J. G. Carbonell, T. M. Mitchell, Eds.), Springer: Berlin, 1984, 331–363.

[21] Muggleton, S.: Inductive Logic Programming, *New Generation Computing*, **8**(4), 1991, 295–318.

[22] Nienhuys-Cheng, S.: Distances and limits on herbrand interpretations, *Proc. of ILP-98* (D. Page, Ed.), 1446, Springer, 1998.

[23] Plotkin, G. D.: A Note on Inductive Generalization, *Machine Intelligence*, **5**, 1970, 153–163.

[24] Ramon, J.: *Clustering and instance based learning in first order logic*, Ph.D. Thesis, Dept. of Computer Science, K.U.Leuven, Belgium, 2002.

[25] Ramon, J., Dehaspe, L.: Upgrading Bayesian Clustering to First Order Logic, *Proceedings of the 9th Belgian-Dutch Conference on Machine Learning*, Department of Computer Science, K.U.Leuven, 1999.

[26] Rouveirol, C.: Extensions of Inversion of Resolution Applied to Theory Completion, in: *Inductive Logic Programming*, Academic Press, 1992, 64–90.

[27] Sebag, M.: Distance Induction in first order logic, *Proc. of ILP-97* (N. Lavrač, S. Džeroski, Eds.), 1297, Springer, 1997.

[28] Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain, 1994.

[29] Thompson, K., Langley, P.: Incremental concept formation with composite objects, *Proceedings of the sixth international workshop on Machine learning*, Morgan Kaufmann Publishers Inc., 1989.

[30] Tversky, A.: Features of Similarity, *Psychological Review*, **84**(4), 1977, 327–352.

[31] Veloso, M. M., Ed.: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2007.

[32] Wieczorek, S., Bisson, G., Gordon, M. B.: Guiding the Search in the NO Region of the Phase Transition Problem with a Partial Subsumption Test, *Machine Learning: ECML 2006*, 4212, Springer, 2006.