Similarity Measures between Order-Sorted Logical Arguments

Victor David\textsuperscript{1}  Jérôme Delobelle\textsuperscript{2}  Jean-Guy Mailly\textsuperscript{2}

\textsuperscript{1} Department of Mathematics and Computer Science University of Perugia, Italy \textsuperscript{2} Université Paris Cité, LIPADE, F-75006 Paris, France
victor.david@unipg.it  jerome.delobelle@u-paris.fr  jean-guy.mailly@u-paris.fr

Abstract

Similarity in formal argumentation has received some attention recently, since one can argue that, in some context, using similar arguments to reach a conclusion is not the same as using dissimilar ones. While existing work considers arguments built from propositional logic, in this work we adapt the notion of similarity measures to arguments built from Order-Sorted First Order Logic, an extension of First Order Logic which allows to represent complex information, considering the type of the data. We study and evaluate our approach with respect to an adaptation of axioms from the literature. This paves the way to new reasoning modes taking into account similarity between arguments in complex settings like ontologies.

1 Introduction

Formal argumentation has become a major topic in Knowledge Representation and Reasoning (KRR), with various applications like decision making [29], defeasible reasoning [16], dealing with inconsistent knowledge bases [12], as well as in multi-agent systems [23]. So, when agents use logic-based information for reasoning, it is possible to build arguments from this information, where typically an argument is a pair made of a set of formulae (called support) and a single formula (called conclusion). The conclusion should be a logical consequence of the support. Examples of arguments are $A = \langle \{p \land q \land r\}, p \land q \rangle$, $B = \langle \{p \land q\}, p \land q \rangle$ and $C = \langle \{p, q\}, p \land q \rangle$. From the definition of arguments, one can identify attacks between them, and then use a semantics to evaluate the arguments. Finally, conclusions of the “strong” arguments are inferred from the base. In the literature, there exist several families of semantics (e.g. extension-based, ranking-based or gradual semantics) to determine which arguments are “strong”. We refer the reader to [1] for a recent overview of the existing families of semantics in abstract argumentation and the differences between these approaches (e.g., definition, outcome, application). Among the existing gradual semantics, like $h$-Categorizer [12], some of them satisfy the Counting (or Strict Monotony) principle defined in [2]. This principle states that each attacker of an argument contributes to weakening the argument. For instance, if the argument $D = \langle \{\neg p \lor \neg q\}, \neg p \lor \neg q \rangle$ is attacked by $A, B, C$, then each of the three arguments will decrease the strength of $D$. However, the three attackers are somehow similar, thus $D$ will lose more than necessary. Consequently, the authors in [4] have motivated the need for investigating the notion of similarity between pairs of such logical arguments. They introduced a set of principles that a reasonable similarity measure should satisfy, and provided several measures that satisfy them. In [3, 5, 6] several extensions of $h$-Categorizer that take into account similarities between arguments have been proposed. All these works consider propositional logic. In this paper, we suggest to adapt the principles behind similarity measures for logical arguments to a much more expressive framework, namely Order-Sorted First Order Logic (OS – FOL) [24], a formalism which generalizes (standard) First Order Logic (FOL). Fragments of OS – FOL have been used for reasoning in various domains (e.g. [17] uses FOL for reasoning about policies, and [22] proposes an architecture for building cognitive agents capable of deduction on facts and rules inferred directly from natural language). More generally, many KRR formalisms can be captured through OS – FOL, like Description Logics [11]. While FOL has already interesting modelling capabilities, OS – FOL allows to naturally model situations where variables belong to a given domain, and there can be relations between the domains of the variables (e.g., the domains made of all the penguins is a subset of the domain contai-
ning all the birds). So, studying logical arguments built from OS – FOL, we are able to apply our work to existing argumentation frameworks based on FOL [13, 10], but also other rich frameworks like Description Logics [11]. This paves the way to applications of argumentation (and similarity measures) to inconsistent knowledge expressed in these rich structured frameworks.

2 Background

2.1 Logic and Arguments

We assume that the reader is familiar with propositionnal logic and First Order Logic (FOL). First Order Logic is a rich framework that develops information about the objects and can also express the relationship between them (using predicates). An example is “Tweety is a penguin, all penguins are birds and all birds have wings, so Tweety has wings” which can be expressed as

\[ \text{penguin(Tweety)} \land (\forall x, \text{penguin}(x) \rightarrow \text{bird}(x)) \land (\forall x, \text{bird}(x) \rightarrow \text{haveWings}(x)) \]

for the premises, and

\[ \text{haveWings(Tweety)} \]

as the consequence. However, this framework does not allow to distinguish between various types of objects. This means that it would be possible to write a FOL formula like \( \text{hasRoots(Tweety)} \), which does not make sense since Tweety is a bird, not a plant. Since we want to apply our method to contexts where data can have a specific type, we use Order-Sorted FOL [24], a generalization of (standard) FOL where all the variables are associated with a sort (as well as the parameters of the predicates).

Definition 1 (Order-Sorted FOL) Let \( S_0 = \{ s_1, \ldots, s_n \} \) be a set of sorts, and \( < \subseteq S_0 \times S_0 \) a partial order over \( S_0 \).

An Order-Sorted First Order Language \( OS – FOL \) is a set of formulae built up by induction from:

– a set \( C \) of constants (\( C = \{ c_1, \ldots, c_k \} \)),
– a set \( V \) of variables (\( V = \{ x^1, y^2, z^3, \ldots \mid s \in S_0 \} \)),
– a set \( P \) of predicates (\( P = \{ P_1, \ldots, P_m \} \)),
– a function \( \text{ar} : P \rightarrow \mathbb{N} \) which gives the arity of predicates,
– a function \( \text{sort} \) s.t. for \( P \in \mathbb{S}_0 \), \( \text{sort}(P) \in \mathbb{S}_0^{\text{ar}(P)} \), and for \( c \in \mathbb{C} \), \( \text{sort}(c) \in S_0 \),
– the usual connectives \( (\neg, \lor, \land, \rightarrow, \leftrightarrow) \), Boolean constants \( T \) (true) and \( \bot \) (false) and quantifier symbols \( (\forall, \exists) \).

A grounded formula is a formula without any variable.

We use lowercase greek letters (e.g. \( \phi, \psi \)) to denote formulae, and uppercase ones (e.g. \( \Phi, \Psi \)) to denote sets of formulae. The set of all Order-Sorted FOL formulae is denoted by \( OS – FOL \). We assume formulae to be prenex, i.e. written as \( Q_1 x_1, \ldots, Q_k x_k \phi \) where \( Q_i \) is a quantifier (for each \( i \in \{ 1, \ldots, k \} \)) and \( \phi \) is a non-quantified formula. A formula \( \phi \) is in negation normal form (NNF) if and only if it does not contain implication or equivalence symbols, and every negation symbol occurs directly in front of an atom. Following [21], we slightly abuse words and denote by \( \text{NNF}(\phi) \) the formula in NNF obtained from \( \phi \) by “pushing down” every occurrence of \( \neg \) (using De Morgan’s law) and eliminating double negations. For instance, \( \text{NNF}(\neg(P(a) \rightarrow Q(a)) \lor \neg Q(b)) = P(a) \land \neg Q(a) \land Q(b) \).

In that case, we call literal either an atom (i.e. a predicate with its parameters) or the negation of an atom. The set of grounded atoms is denoted by \( A \). We denote by \( L \) the set of literals occurring in \( \text{NNF}(\phi) \), hence \( L = \{ (\neg(P(a) \rightarrow Q(a)) \lor \neg Q(b)) \} = \{ (P(a), \neg Q(a), Q(b)) \} \). For a given set of predicates \( P \), we define \( L = \{ P(x^1_1, \ldots, x^1_k), \neg P(x^2_1, \ldots, x^2_k) \mid P \in \mathbb{P}, \text{sort}(P) = \{ s_1, \ldots, s_j \} \} \) the set of literals. We say that a literal \( L \) is negative when it starts with a negation, denoted by \( \text{Pol}(L) = 1 \). Otherwise we say that it is positive, denoted by \( \text{Pol}(L) = + \). And we say that two literals have the same polarity if they are either both positive or both negative. Finally, given a grounded literal \( L = \pm P(a_1, \ldots, a_k) \) where \( \pm \) indicates the polarity of \( L \), \( \text{Pred}(L) \) corresponds to the name of the predicate underlying \( L \), and \( \text{Para}(L) = \{ a_1, \ldots, a_k \} \).

Consider \( \phi \in OS – FOL \). \( \phi \) is in a conjunctive normal form (CNF) if it is a conjunction of clauses \( \{ l_i \} \) where each clause \( c_i \) is a disjunction of literals \( \lor \). For instance \( P(a) \land (Q(a) \lor Q(b)) \) is in CNF while \( P(a) \lor Q(a) \lor Q(b) \) is not. CNF formulae are particular NNF formulae. Clauses are also usually represented as sets of literals, and CNF formulae as sets of clauses.

In OS – FOL, the partial order \( < \) represents “sub-type” relations between groups of entities. For instance, the fact that dogs are a special type of mammals can be represented by such a sub-type relation. In the case where \( s_1 < s_2 \), a predicate which expects a parameter of type \( s_2 \) can be applied to a constant or variable of type \( s_1 \) (for instance, a predicate about mammals can be applied to dogs).

\[ d \rightarrow m \rightarrow a \rightarrow l \rightarrow p \]

\[ \text{Figure 1} – \text{Hierarchy of sorts from Example 1. An arrow from } s_1 \text{ to } s_2 \text{ means } s_1 < s_2. \]

Example 1 OS – FOL formulae can be used to reason about ontological information. Assume that we have the following information: mammals and birds are animals, dogs and cats are mammals, penguins and chickens are birds. Moreover, Zazu is a bird, Tweety is a penguin, and Dogmatix is a dog. Finally, animals are living beings, as well as...
plants. This can be represented by the following sorts and constants:

- \( S_0 = \{ m, b, a, d, c, p, ch, l, pl \} \) with \( m < a, b < a, d < m, c < m, p < b, c < b, a < l, pl < l \) (see Figure 1).
- \( Z \in C \) with \( \text{sort}(Z) = b \) is a constant for Zazu.
- \( T \in C \) with \( \text{sort}(T) = p \) is a constant for Tweety.
- \( D \in C \) with \( \text{sort}(D) = d \) is a constant for Dogmatix.

We know that all birds have wings, and both mammals and birds are warm-blooded. Also, some birds and some mammals fly, but not all of them. If a bird is wounded, then it cannot fly. If a bird is a penguin, then it cannot fly. Some birds are wounded. Finally, Tweety is a penguin.

This information can be represented by the predicates

\[ P = \{ hW, wB, f, w, p \} \]

standing respectively for “have-Wings”, “warmBlooded”, “fly”, “wounded” and “penguin” s.t. \( \text{ar}(P_i) = 1 \) and \( \text{sort}(P_i) = a \) for each \( P_i \in P \).

We can build, e.g. the formula \( \forall b \cdot hW(x^b) \) meaning that all birds have wings (because the variable \( x^b \) has the sort \( b \)).

The other pieces of information are represented by

\[ \forall x^b \cdot wB(x^b) \]

\[ \exists x^{m_b} f(x^b) \land \neg f(x^b) \]

\[ \forall x^b \cdot wB(x^b) \to \neg f(x^b) \]

\[ \forall x^p \cdot p(x^b) \to \neg f(x^b) \]

\[ \forall x^w \cdot wB(x^b) \]

\[ p(T) \]

However formulae like \( \exists x^b, f(x^b) \) or \( \forall x^p \cdot wB(x^p) \) are not well-formed, since the predicates \( f \) and \( wB \) cannot be applied to living beings or plants.

OS – FOL formulae are evaluated via a notion of structure:

**Definition 2 (Structure)** Given \( n \in \mathbb{N} \), an \( n \)-sorted structure is \( \text{St} \) = (Dom, Rel, Cons) where:

- Dom = \( \{ D_1, \ldots, D_n \} \) are the (non-empty) domains,
- Rel = \( \{ R_1, \ldots, R_m \} \) are relations over the domains,
- Cons = \( \{ c_1, \ldots, c_l \} \) are constants in the domains.

**Example 2** A structure associated with the OS – FOL from Example 1 is \( \text{St} \) = (Dom, Rel, Cons) where:

- Dom = \( \{ D_1 \ldots D_3 \} \) are the sets of all individuals of the various types (e.g. \( D_1 \) is the set of mammals, corresponding to the sort symbol \( m \); \( D_2 \) is the set of birds, corresponding to the sort symbol \( b \); etc.),
- Rel = \( \{ R_1, \ldots, R_3 \} \) are the relations corresponding to the predicate symbols (e.g. \( R_1 \) identifies winged animals, . . . )
- Cons = \( \{ \text{Zazu, Tweety, Dogmatix} \} \) are respectively a particular bird (an element of the domain \( D_2 \) associated with the sort \( b \)), a particular penguin (an element of \( D_6 \) associated with the sort \( p \)) and a particular dog (an element of \( D_4 \) associated with the sort \( d \)).

Classical first order logic formulae can be evaluated via 1-sorted structures. For this reason, any fragment of first order logic is captured by OS – FOL. Now, we show how OS – FOL formulae are interpreted.

**Definition 3 (Interpretation)** An interpretation \( \text{IS} \) over a structure \( \text{St} \) assigns to elements of the OS – FOL vocabulary some values in the structure \( \text{St} \). Formally,

\[ \text{IS}(s_i) = D_i, \text{for } i \in \{ 1, \ldots, n \} \text{ s.t. for each } s_i, s_j \in \text{IS}, \text{if } s_i \leq s_j \text{ then } \text{IS}(s_i) \subseteq \text{IS}(s_j) \] (each sort symbol is assigned to a domain s.t. the sub-type relations are respected),

\[ \text{IS}(P_i) = R_i, \text{for } i \in \{ 1, \ldots, m \} \] (each predicate symbol is assigned to a relation),

\[ \text{IS}(a_i) = c_i, \text{for } i \in \{ 1, \ldots, l \} \] (each constant symbol is assigned to a constant value). As a shorthand, we write \( \text{IS}(s_1, \ldots, s_k) = \text{IS}(s_1) \times \cdots \times \text{IS}(s_k) \). Then satisfaction of formulæ is recursively defined by:

\[ \text{IS} \models P_i(x_1, \ldots, x_k), \text{where } (x_1, \ldots, x_k) \in \text{IS}(s_1, \ldots, s_k) \text{ with sort}(x_i) = s_i \text{ for each } i \in \{ 1, \ldots, k \}, \text{iff } (x_1, \ldots, x_k) \in R_i, \]

\[ \text{IS} \models \exists x^b \phi \text{ iff } \text{IS}(x^b \mapsto v) \models \phi \text{ for some } v \in D_i, \]

\[ \text{IS} \models \forall x^b \phi \text{ iff } \text{IS}(x^b \mapsto v) \models \phi \text{ for each } v \in D_i, \]

\[ \text{IS} \models \phi \land \psi \text{ iff } \text{IS} \models \phi \text{ and } \text{IS} \models \psi, \]

\[ \text{IS} \models \phi \lor \psi \text{ iff } \text{IS} \models \phi \text{ or } \text{IS} \models \psi, \]

\[ \text{IS} \models \neg \phi \text{ iff } \text{IS} \not\models \phi, \]

where \( \text{IS}(x^b \mapsto v) \) is a modified version of \( \text{IS} \) s.t. the variable \( x^b \) is replaced by a value \( v \) in the domain \( D_i \), corresponding to the sort symbol \( s_i \). Finally, if \( \Phi \) is a set of formulæ, then \( \text{IS} \models \Phi \text{ iff } \text{IS} \models \phi \text{ for each } \phi \in \Phi \).

Observe that Definition 3 does not specify the satisfaction of implications and equivalences, but they can be defined as usual by \( (\phi \land \psi) \equiv (\neg \phi \lor \psi) \) and \( (\phi \lor \psi) \equiv (\neg \phi \land \psi) \).

We use \( \text{Mod}(\Phi) \) to denote the set of interpretations satisfying a set of formulæ \( \Phi \), and we call \( \Phi \) consistent if \( \text{Mod}(\Phi) \neq \emptyset \).

**Example 3** Continuing Example 1, we define \( \text{IS} \) by:

\[ \text{IS}(m) = D_1, \text{IS}(b) = D_2, \ldots, \text{IS}(pl) = D_5, \]

\[ \text{IS}(hW) = R_1, \ldots, \text{IS}(p) = R_5, \]

\[ \text{IS}(Z) = \text{Zazu}, \text{IS}(T) = \text{Tweety}, \text{IS}(D) = \text{Dogmatix}. \]

The formula \( \phi = \forall x^b hW(x^b) \) is satisfied by \( \text{IS} \) since all elements of the domain \( D_2 \) associated with the sort \( b \) actually have wings. On the contrary, consider the set of formulæ \( \Phi = \{ \forall x^p f(x^p), \forall x^p \neg f(x^p) \} \). This set of formulæ is not satisfied, because \( p < b \) and so the domains satisfy \( D_6 \subset D_2 \), meaning that all penguins are birds. Then, from FO we can deduce that any penguin can fly (because of the first formula) and cannot fly (because of the second formula) at the same time. So, this formula is not satisfied by \( \text{IS} \). Notice that we could not define an interpretation \( \text{IS}' \) s.t. \( \text{IS}'(Z) = \text{Tweety} \) and \( \text{IS}'(T) = \text{Zazu} \), since \( \text{Zazu} \) is a bird, and \( T \) has the sort \( p \) (i.e. it can only be a penguin, not any kind of bird).

Now we introduce the concept of instantiation, i.e. grounded formulæ which are compatible with a given OS – FOL formulæ.

**Definition 4 (Instantiation)** Given \( \Phi \) a set of OS – FOL formulæ and \( \text{IS} \) an interpretation over a structure \( \text{St} \), the set of instantiations of \( \Phi \) is defined recursively by:
\textbf{Definition 5 (Consequence Relation)} Consider the set of formulae \( \Phi = \{ \phi_1, \ldots, \phi_n \} \) where \( \phi_1 \) is a grounded formula s.t. \( I_{\Phi} \models \phi_1 \). We conclude that \( \Phi \subset \top \) iff \( \phi_1 \land \cdots \land \phi_n \models \phi_1 \). An example of logic consists of \( (L, \models) \) where \( L \) is an OS – FOL language following Definition 1 and \( \models \) is the consequence relation from Definition 5. Classical logic can be used to define arguments, i.e., logic-based representation of reasons supporting a specific conclusion. Logical arguments usually need to satisfy some constraints [12]:

**Definition 6 (Logical Argument)** An argument built under a logic \( (L, \models) \) is a pair \( \langle \Phi, \phi \rangle \), where \( \Phi \subset \top \) and \( \phi \in L \). The set of all arguments \( \text{Arg}(L) \) is defined as follows:

Example 5 Let \( A_1 \) and \( A_2 \) be two arguments:

\[ A_1 = \{ (\exists x)^b w(x^b), \forall x^b w(x^b) \rightarrow \neg f(x^b) \} \]

\[ A_2 = \{ (p(Tweety), \forall x^b p(x^b) \rightarrow \neg f(x^b), \neg f(Tweety)) \} \]

Note that two sets of formulae \( \Phi, \Psi \subset \top \) are equivalent, denoted by \( \Phi \equiv \Psi \), iff there is a bijection \( f : \Phi \rightarrow \Psi \) s.t. \( \forall \phi \in \Phi, \phi \equiv f(\phi) \). We use this restricted equivalence notion to avoid equivalences that could be false due to incorrect information. For example, the sets \( \{ \text{Square}(a), \text{Square}(a) \rightarrow \text{Rectangle}(a) \} \) and \( \{ \text{Rectangle}(a), \text{Rectangle}(a) \rightarrow \text{Square}(a) \} \) should not be equivalent. However, we may want to consider that a set of formulae is equivalent with the conjunction of its elements (e.g. \( \{ P(a), Q(a) \} \) and \( \{ P(a) \land Q(a) \} \) are equivalent). To make them equivalent, we borrow the method used in [7]. We transform every formula into a CNF, then we split it into a set containing its clauses. In our approach, we consider one CNF per formula. For that purpose, we will use a finite sub-language \( F \) that contains one formula per equivalent class and the formula should be in CNF.

**Definition 7 (Finite CNF over Language \( F \))** Let \( F \subset \top \) and \( \forall \phi \in L \), there is a unique \( \psi \in F \) s.t. \( \phi \equiv \psi \). We define \( \text{CNF}(\Phi) = \psi \).

While we do not specify the elements of \( F \), we use concrete formulae in the examples, and they are assumed to belong to \( F \).

We can lift the consequence relation to sets of formulae by \( \{ \psi_1, \ldots, \psi_n \} \models \phi \) if \( \psi_1 \land \cdots \land \psi_n \models \phi \). An example of logic consists of \( (L, \models) \) where \( L \) is an OS – FOL language following Definition 1 and \( \models \) is the consequence relation from Definition 5. Classical logic can be used to define arguments, i.e., logic-based representation of reasons supporting a specific conclusion. Logical arguments usually need to satisfy some constraints [12]:

**Example 4** Consider the set of formulae \( \Phi = \{ \phi_1, \phi_2 \} \). We assume here that the domain associated with the sort \( b \) is the set \( \{ \text{Tweety, Zazu} \} \). Applying Definition 4, \( I_{\Phi} = \{ I_1 \cup I_2, I_1 \in \text{Inst}_b(\{ \exists x^b w(x^b) \}), I_2 \in \text{Inst}_b(\{ \forall x^b w(x^b) \rightarrow \neg f(x^b) \}) \} \). We start with the first formula, i.e. \( \phi_1 = \exists x^b w(x^b) \).

**Example 5** Let \( A_1 \) and \( A_2 \) be two examples of arguments:

\[ A_1 = \{ (\exists x)^b w(x^b), \forall x^b w(x^b) \rightarrow \neg f(x^b) \} \]

\[ A_2 = \{ (p(Tweety), \forall x^b p(x^b) \rightarrow \neg f(x^b), \neg f(Tweety)) \} \]

Note that two sets of formulae \( \Phi, \Psi \subset \top \) are equivalent, denoted by \( \Phi \equiv \Psi \), iff there is a bijection \( f : \Phi \rightarrow \Psi \) s.t. \( \forall \phi \in \Phi, \phi \equiv f(\phi) \). We use this restricted equivalence notion to avoid equivalences that could be false due to incorrect information. For example, the sets \( \{ \text{Square}(a), \text{Square}(a) \rightarrow \text{Rectangle}(a) \} \) and \( \{ \text{Rectangle}(a), \text{Rectangle}(a) \rightarrow \text{Square}(a) \} \) should not be equivalent. However, we may want to consider that a set of formulae is equivalent with the conjunction of its elements (e.g. \( \{ P(a), Q(a) \} \) and \( \{ P(a) \land Q(a) \} \) are equivalent). To make them equivalent, we borrow the method used in [7]. We transform every formula into a CNF, then we split it into a set containing its clauses. In our approach, we consider one CNF per formula. For that purpose, we will use a finite sub-language \( F \) that contains one formula per equivalent class and the formula should be in CNF.

**Definition 7 (Finite CNF over Language \( F \))** Let \( F \subset \top \) and \( \forall \phi \in L \), there is a unique \( \psi \in F \) s.t. \( \phi \equiv \psi \). We define \( \text{CNF}(\Phi) = \psi \).

While we do not specify the elements of \( F \), we use concrete formulae in the examples, and they are assumed to belong to \( F \).

Now we introduce \( \text{UC}(\Phi) \) as the representation of the formulae in \( \Phi \) as one set of clauses. Intuitively, recall that any formula can be seen as a set of clauses, associated with a sequence of quantifiers. A set of formulae can then be seen
as set of clauses and a sequence of quantifiers, such that variables are renamed to avoid ambiguities. As an example, assume $\phi_1 = \exists x P(x) \land Q(x)$ and $\phi_2 = \exists x R(x) \vee \phi_2'. We have $UC(\{\phi_1, \phi_2\}) = \exists x, x' (P(x), Q(x)) \vee R(x')$. Formally, for $\Phi = \{Q_{\phi_i} \mid i \in \mathbb{N}\} \subseteq \mathcal{F}$, where $\phi_i$ is a non-quantified CNF formula (i.e. a set of clauses for some $\psi \in \mathcal{F}$), and $Q_{\phi_i}$ is the sequence of quantifiers associated with $\phi_i$, we define $UC(\Phi) = \{Q_{\phi_1} \cdots Q_{\phi_n} \mid \bigcup_{\phi_i \in \Phi} Q_{\phi_i}(\bigcup_{\delta \in \Phi} \delta^*)\}$, where a renaming is applied to each clause ($\delta^*$) and each sequence of quantifiers ($Q_{\phi_i}$) in order to guarantee that no variable is shared between quantifiers $Q_{\phi_i}$ and $Q_{\phi_j}$ (with $i \neq j$) or between clauses coming from different formulae $\phi_i$ and $\phi_j$ (with $i \neq j$). We simply write $UC(\phi)$ instead of $UC(\{\phi\})$, for $\phi \in \mathcal{F}$.

Implicitly, in the rest of the paper, we consider $UC(\Phi)$ as the set made of a single formula such that the sequence of quantifiers is the concatenation of $Q_{\phi_1} \cdots Q_{\phi_n}$ and the non-quantified part is the CNF formula corresponding to the set of clauses $\bigcup_{\phi_i \in \Phi} Q_{\phi_i}(\bigcup_{\delta \in \Phi} \delta^*)$.

Note that $UC(\{P(a), Q(a)\}) = UC(\{P(a) \land Q(a)\}) = \{P(a), Q(a)\}$ or with some quantifiers $UC(\{\forall x \exists y P(x, y), \forall x Q_1(x) \lor Q_2(x)\}) = UC(\{\forall x \exists y_1 \exists y_2 P(x, y_1, x, y_2) \land \forall x \forall y \forall z Q_1(x, y, z) \lor Q_2(x, y, z)\} = \{\forall x \exists y_1 \exists y_2 P(x, y_1, x, y_2) \land \forall x \forall y \forall z Q_1(x, y, z) \lor Q_2(x, y, z)\}.

Let us now introduce the notion of compiled argument.

**Definition 8 (Compiled Argument)** The compilation of $A \in \mathcal{A}(\mathcal{L})$ is $A' = \{UC(Supp(A)), Conc(A)\}$.

**Example 6** Consider $A, B, C \in \mathcal{A}(\mathcal{L})$ such that
- $A = \{P(a) \land Q(a) \land Q(b), P(a) \land Q(a)\}$,
- $B = \{P(a) \land Q(a), P(a) \land Q(a)\}$, and
- $C = \{P(a), Q(a)\}$.

The compilations of the three arguments $A, B, C$ are:
- $A' = \{\{P(a), Q(a), Q(b)\}, P(a) \land Q(a)\}$,
- $B' = \{\{P(a), Q(a)\}, P(a) \land Q(a)\}$, and
- $C' = \{\{P(a), Q(a)\}, P(a) \land Q(a)\}$.

We can see in Example 6 that argument $A$ is not concise, meaning that it has irrelevant information ($Q(b)$) for implying its conclusion. As it was shown in [7], using clausal arguments ensure that the arguments are concise.

**Definition 9 (Equivalent Arguments)** Two arguments $A, B \in \mathcal{A}(\mathcal{L})$ are equivalent, denoted by $A \approx B$, iff $UC(Supp(A)) = UC(Supp(B))$ and $UC(Conc(A)) = UC(Conc(B))$. We denote by $A \neq B$ when $A$ and $B$ are not equivalent.

**Definition 10 (Sub-argument)** Given two arguments $A = \langle \Phi, \phi \rangle$ and $B = \langle \Psi, \phi \rangle$, we say that $A$ is a sub-argument of $B$ if $UC(\Phi) \subseteq UC(\Psi)$.

### 2.2 Binary Similarity Measure between OS – FOL Argument

A similarity measure is used to indicate whether two arguments are similar or not, i.e. whether they share some parts of the reasoning mechanism used to build the arguments.

**Definition 11 (Similarity Measure)** Let $\mathcal{X}$ be a set of objects. A similarity measure on $\mathcal{X}$, denoted by $sim^2$, is a function from $\mathcal{X} \times \mathcal{X}$ to $[0, 1]$.

In this section, we focus on similarity measures over arguments, i.e. $\mathcal{X} = \mathcal{A}(\mathcal{L})$. Intuitively, $sim^2(\mathcal{A}(\mathcal{L}))(A, B)$ is close to 0 if the difference between $A$ and $B$ is important, while it is close to 1 if the arguments are similar. Several principles that similarity measures should satisfy have been discussed in the literature [4, 8, 7]. Some of the principles (Maximality, Symmetry, Substitution, and Syntax Independence) can be stated exactly as in the literature [7], since they do not concern the internal structure of the arguments. Notice that some authors have argued against the fact that a similarity measure should absolutely satisfy symmetry[28, 19]. Some of the principles can be stated exactly as in the literature [7], since they do not concern the internal structure of the arguments. It is the case of these principles: Maximality states that the similarity between an argument and itself should be maximal; Symmetry states that the similarity measure should be symmetric; Substitution states that two fully similar arguments should be equally similar to any third argument; and Syntax Independence states that similarity between arguments should be independent from the syntax. For the other ones, we may need to adapt them to our OS – FOL-based arguments.

First, we adapt the Minimality principle. It states that, if two arguments do not have anything in common in their content, then their degree of similarity should be minimal. While, in propositional logic, determining the set of common propositional variables is enough, here we need to consider (domains of) predicates and constants. We do not consider variables here since they are used in the context of quantifiers: there is no reason to assume that there is something common between $\forall x, P(x)$ and $\forall x, Q(x)$.

Before presenting the Minimality principle, let us introduce some useful notations. Given a formula $\phi$, $Dom(\phi) = \bigcup_{P \in Pred(\phi)} \text{sort}(P)$ represents the domains of the predicates in $\phi$ (or, more precisely, the sort symbols associated with these domains). We extend the notation to $Dom(\Phi) = \bigcup_{\phi \in \Phi} Dom(\phi)$ for $\Phi$ a set of formulae.

**Principle 1 (Minimality)** A similarity measure $\text{sim}^2(\mathcal{L})$ satisfies Minimality iff for all $A, B \in \mathcal{A}(\mathcal{L})$, if

3. Notice that some authors have argued against the fact that a similarity measure should absolutely satisfy symmetry [28, 19].
1. one of $A, B$ is not trivial,
2. $\forall s_j \in \text{Dom}(\text{Supp}(A))$, $s_j \in \text{Dom}(\text{Supp}(B)) \; s.t. \; s_i < s_j$ or $s_j < s_i$ or $s_i = s_j$,
3. $\forall s_j \in \text{Dom}(\text{Conc}(A))$, $\exists s_j \in \text{Dom}(\text{Conc}(B)) \; s.t. \; s_i < s_j$ or $s_j < s_i$ or $s_i = s_j$, then $\text{sim}_{\text{FOL}}(A, B) = 0$.

The first condition excludes the case where the arguments have no formula in the support and therefore no sort to compare and the second and third conditions ensure that each argument has completely different information.

The second (resp. third) principle states that the more an argument shares formulae in its support (resp. conclusion) with another one, the higher is their similarity. For these principles, we need to introduce the notation $\subseteq$ which represents the set of all grounded clauses in OS → FOL.

Notice that we consider in the two next principles only arguments having no irrelevant information (i.e., $A^*, B^*, C^* \in \text{Arg}(L)$) allowing safe handling of their similarity. The first conditions allow us to isolate the specific behaviours on second and third conditions. For Principle 2 focusing on supports we ensure that we have identical or completely different conditions such that it does not contradict that $(A, B)$ is more similar than $(A, C)$. We cannot, as in Principle 3, use the fact that the conclusions of $B$ and $C$ are equivalent as this would prevent conditions 2 and 3 from being satisfied (due to the minimality of the supports of an argument, e.g. the case of one support included in another is not possible). Please note that the constraints $C_A \setminus B_A \subseteq C$ ensure that the distinct elements in $C$ cannot have similarity with $A$.

**Principle 2 (Monotony – Strict Monotony)**

A similarity measure $\text{sim}_{\text{FOL}}(L)$ satisfies Monotony iff for all $A, B, C, A^*, B^*, C^* \in \text{Arg}(L)$, if
1. $\text{UC}(\text{Conc}(A)) = \text{UC}(\text{Conc}(B))$ or $\forall s_i \in \text{Dom}(\text{Conc}(A))$, $\exists s_j \in \text{Dom}(\text{Conc}(C)) \; s.t. \; s_i < s_j$ or $s_j < s_i$ or $s_i = s_j$,
2. $\text{UC}(\text{Supp}(A)) \cap \text{UC}(\text{Supp}(C)) \subseteq \text{UC}(\text{Supp}(A)) \cap \text{UC}(\text{Supp}(B))$,
3. for $B_A = \text{UC}(\text{Supp}(B)) \setminus \text{UC}(\text{Supp}(A))$ and $C_A = \text{UC}(\text{Supp}(C)) \setminus \text{UC}(\text{Supp}(A))$, $B_A \subseteq C_A$, $C_A \setminus B_A \subseteq C$ and $\forall s_i \in \text{Dom}(\text{Supp}(A))$, $\exists s_j \in \text{Dom}(C_A \setminus B_A) \; s.t. \; s_i < s_j$ or $s_j < s_i$ or $s_i = s_j$, then $\text{sim}_{\text{FOL}}(A, B) \geq \text{sim}_{\text{FOL}}(A, C)$.

(Monotony)

– If the inclusion in condition 2. is strict or, $\text{UC}(\text{Supp}(A)) \cap \text{UC}(\text{Supp}(C)) \neq \emptyset$ and $B_A \subset C_A$, then $\text{sim}_{\text{FOL}}(A, B) > \text{sim}_{\text{FOL}}(A, C)$.

(Strict Monotony)

**Principle 3 (Dominance – Strict Dominance)**

A similarity measure $\text{sim}_{\text{FOL}}(L)$ satisfies Dominance iff for all $A, B, C, A^*, B^*, C^* \in \text{Arg}(L)$, if
1. $\text{UC}(\text{Supp}(B)) = \text{UC}(\text{Supp}(C))$,
2. $\text{UC}(\text{Conc}(A)) \cap \text{UC}(\text{Conc}(C)) \subseteq \text{UC}(\text{Conc}(A)) \cap \text{UC}(\text{Conc}(B))$.

$\text{UC}(\text{Conc}(B))$, $3. \; \text{for } B_A = \text{UC}(\text{Conc}(B)) \setminus \text{UC}(\text{Conc}(A))$ and $C_A = \text{UC}(\text{Conc}(C)) \setminus \text{UC}(\text{Conc}(A))$, $B_A \subseteq C_A$, $C_A \setminus B_A \subseteq C$ and $\forall s_i \in \text{Dom}(\text{Conc}(A))$, $\exists s_j \in \text{Dom}(C_A \setminus B_A) \; s.t. \; s_i < s_j$ or $s_j < s_i$ or $s_i = s_j$, then $\text{sim}_{\text{FOL}}(A, B) \geq \text{sim}_{\text{FOL}}(A, C)$.

(Dominance)

– If the inclusion in cond. 2. is strict or, $\text{UC}(\text{Conc}(A)) \cap \text{UC}(\text{Conc}(C)) \neq \emptyset$ and $B_A \subset C_A$, then $\text{sim}_{\text{FOL}}(A, B) > \text{sim}_{\text{FOL}}(A, C)$.

(Strict Dominance)

### 3 Similarity Models

To define the similarity between two arguments, we will split the reasoning in several steps, corresponding to the different levels used in the construction of the arguments. At each level, different similarity measures can be used to compare the objects, and various aggregation functions can be used to go from the comparison of objects to the comparison of sets of objects (leading to the next level). This level structure is based on the fact that our arguments are built from CNF formulæ. More precisely,

**Level 1:** compute the similarity between two literals, by combining the similarity between their polarity, the predicate involved, and the predicates parameters (Section 3.1);

**Level 2:** then we use the previous level and aggregate the result of comparing literals in order to compare grounded clauses (Section 3.2);

**Level 3:** next, we aggregate the similarity between grounded clauses to obtain the similarity between sets of grounded clauses (Section 3.3);

**Level 4:** finally, we can define the similarity between sets of instantiations, since each instantiation is a set of grounded clauses (Section 3.4).

The similarity between two arguments is obtained by computing the similarity between the instantiations of their supports and the similarity between their conclusions, so Level 4 is the last level of abstraction that we need.

#### 3.1 Similarity between literals

Recall that a literal is a predicate with or without a negation operator “¬”. To know how similar are two literals, we compute the similarity between two atoms (i.e. without the literals’ polarity) and combine these scores according to the polarity. At the level of atoms, we identify two parameters influencing the similarity : the value of the predicates and those of their vectors of parameters. Thus the similarity between two atoms can be seen as a combination of three functions : $c$ to compute the similarity between two vectors of constants, $p$ between two predicates and $g$ to aggregate these scores.
Definition 12 (Similarity between Atoms) Let \( c : \bigcup_{j,k=1}^{\infty} C^j \times C^k \to [0,1] \) be a similarity measure between a pair of vectors of constants, \( p : P \times P \to [0,1] \) be a similarity measure between a pair of predicates and \( g : [0,1] \times [0,1] \to [0,1] \) be an aggregation function. Given two atoms \( A_1 = P_1(a_1, \ldots, a_j) \) and \( A_2 = P_2(b_1, \ldots, b_k) \), to compute the similarity score between \( A_1 \) and \( A_2 \) we define \( \text{simA}(p, g)(A_1, A_2) = g(p(\text{Pred}(A_1), \text{Pred}(A_2)), c(\text{Para}(A_1), \text{Para}(A_2))) \).

A possible \( p \) is the function returning 1 if the predicates are the same, 0 otherwise.

Definition 13 (Function Equal) Let \( x, y \) be two arbitrary objects. The function \( \text{eq} : \mathcal{X} \times \mathcal{X} \to [0,1] \) is defined by \( \text{eq}(x, y) = 1 \) if \( x = y \); or \( \text{eq}(x, y) = 0 \) otherwise.

We propose an instance of \( c \) suited to vectors of objects. Other methods could be used and are kept for future work.

Definition 14 (Pointwise Similarity) Let \( X = (x_1, \ldots, x_j), Y = (y_1, \ldots, y_k) \) be arbitrary vectors of objects. The pointwise similarity between \( X \) and \( Y \) is:

\[
\text{pws}(X, Y) = \begin{cases} 1 & \text{if } X = Y = \emptyset \\ \frac{\sum_{i=1}^{j} \text{eq}(x_i, y_i)}{\max(1,k)} & \text{otherwise} \\ \end{cases}
\]

Having a similarity score between two atoms, we propose to use the polarities as binary factors of acceptance or not of the similarity between atoms.

Definition 15 (Similarity between Literals) Consider two literals \( l_1, l_2 \in \mathbb{L} \), such that the respective atoms are \( A_1 \) and \( A_2 \). We define \( \text{simL}(p, g, c) : \mathbb{L} \times \mathbb{L} \to [0,1] \), the similarity measure between two literals according to a similarity measure between atoms \( \text{simA}(p, g, c) \) s.t.: \( \text{simL}(p, g, c)(l_1, l_2) = \begin{cases} \text{simA}(p, g, c)(A_1, A_2) & \text{if } \text{Pol}(l_1) = \text{Pol}(l_2) \\ 0 & \text{otherwise} \end{cases} \)

Example 7 \( \text{simL}(\text{min}, \text{eq}, \text{pws})(P(A, B), \neg P(A, C)) = 0 \) because the polarity is not the same. Conversely, we have \( \text{simL}(\text{min}, \text{eq}, \text{pws})(P(A, B), P(A, C)) = \frac{1}{2} \) because: \( \text{simL}(\text{min}, \text{eq}, \text{pws})(P(A, B), P(A, C)) = \text{simL}(\text{min}, \text{eq}, \text{pws})(P(A, B), P(A, C)) = \text{min}(\text{eq}(P, P), \text{pws}((A, B), (A, C))) = \text{min}(1, \text{eq}(A, A) \oplus \text{eq}(B, C)) = \text{min}(1, \frac{1}{2}) = \frac{1}{2} \).

3.2 Similarity between grounded clauses

From the level two of the definition of our similarity measures on arguments, we will need several mathematical tools that can be defined in an abstract way. In this part, we apply these tools only for level 2 (the comparison of two CNF formulae), but they will be applicable also at the next levels. We start with the notion of aggregation function.

Definition 16 (Aggregation Function) Let \( \mathcal{X} \) be a set of objects and \( \{x_1, x_2, \ldots\} \subseteq \mathcal{X} \). We say that \( \oplus \) is an aggregation function if \( \forall k \in \mathbb{N}, \oplus \) is a mapping \( [0,1]^k \to [0,1] \) such that:

- if \( x_i \geq x_j \) then \( \oplus(x_1, \ldots, x_i, \ldots, x_k) \geq \oplus(x_1, \ldots, x_j, \ldots, x_k) \) (non-decreasingness)
- \( \oplus(0, \ldots, 0) = 0 \) (weak minimality)
- \( \forall i \in \{1, \ldots, k\}, \oplus(x_i) = x_i \) (identity)

These properties are satisfied by e.g. min, max and avg.

Now we introduce the notion of membership function which expresses how much an object is similar to the elements of a set.

Definition 17 (Membership Function) Given \( \mathcal{X} \) a set of objects, \( x \in \mathcal{X} \) an object, \( X \subseteq \mathcal{X} \) an aggregation function and \( \text{simL} \) a similarity measure the membership function of \( x \) in \( X \), \( \epsilon^X_{\oplus, \text{simL}} : \mathcal{X} \times \mathcal{X} \to [0,1] \) is defined by \( \epsilon^X_{\oplus, \text{simL}}(x, X) = \epsilon_{\oplus, \text{eq}}(\text{pws}(\text{p}(x), \text{p}(X))) \).

Let us note that classical set-membership can be captured by \( \epsilon_{\text{max}, \text{eq}} \) where \( \text{eq} \) is the equality function from Definition 13. Now we can evaluate how much a literal is similar to a clause, i.e. a set of literals: given \( l \in \mathbb{L} \) a literal, \( L \subseteq \mathbb{L} \) a set of literals and \( \oplus \) an aggregation function, we define the function \( s^L = \epsilon^L_{\oplus, \text{simL}(\text{p}, \text{c})} \). Then, the similarity between two grounded clauses is computed by \( \text{simC}^L_{\oplus, \text{c}} \).

Definition 18 (Membership of a literal in a set of literals) Let \( l \in \mathbb{L} \) be a literal, \( L \subseteq \mathbb{L} \) be a set of literals and \( \oplus \) an aggregation function. We define the membership of a literal in a set of literals by the function \( \epsilon^L_{\oplus, \text{eq}} : \mathbb{L} \times 2^\mathbb{L} \to [0,1] \) s.t.:

\[
\epsilon^L_{\oplus, \text{eq}}(l, L) = \bigoplus_{F \subseteq L}(\text{simL}(\text{p}, \text{c})(l, F))
\]

Definition 19 (Similarity measure between two clauses) Let \( \delta_1 = l_1 \lor \ldots \lor l_j, \delta_2 = l'_1 \lor \ldots \lor l'_k \in \text{OS} - \text{FOL} \) be two grounded clauses. The similarity measure between two grounded clauses is

\[
\text{simC}^L_{\oplus, \text{c}} : \text{OS} - \text{FOL} \times \text{OS} - \text{FOL} \to [0,1].
\]

Roughly speaking, what we mean in Definition 19 (and subsequent similar definitions) is that the similarity between two grounded clauses must be computed using a similarity measure (in the sense of Definition 11), and ideally this measure should use the membership function \( \epsilon^L_{\oplus, \text{eq}} \) to compare a given literal with a set of literals (i.e. with a grounded clause). But at this level of abstraction, we do not explicitly defined one function realizing this computation, Def. 19 characterizes the general meaning of what a similarity measure between clauses should be. In the rest of this paper, we will use one concrete approach to define similarity measures, namely Tversky’s ratio model [28], but other approaches could be used instead as soon as they satisfy the requirements of Def. 19 (and Def. 11).
Tversky’s ratio model [28] is a general similarity measure which encompasses different well known similarity measure as the Jaccard measure [18], Dice measure [15], Sorensen one [27], Symmetric Anderberg [9] and Sokal and Sneath 2 [26]. We propose to extend it in two different ways. Firstly, instead of using the usual operators of membership of an element to a set, we propose to use our parameterisable membership function $\varepsilon$ (see Definition 17). Then a new parameter $\gamma$ allows us to have a lower evaluation between a set of literals than a set of clauses (or instantiations), i.e. when sets of objects are interpreted disjunctively or conjunctively.

**Definition 20 (Extended Tversky Measure)** Let $X, Y \subseteq X$ be arbitrary sets of objects. Let $\varepsilon_{\oplus, \sim}^X$ be a membership function with $\oplus$ an aggregation function and $\sim$ a similarity measure. We denote by $\text{avg}$ the average function. Let us consider

- $a = \text{avg} \left\{ \sum_{x \in X} \varepsilon_{\oplus, \sim}^X(x, y), \sum_{y \in Y} \varepsilon_{\oplus, \sim}^X(y, x) \right\}$,
- $b = \sum_{x \in X} (1 - \varepsilon_{\oplus, \sim}^X(x, y))$,
- $c = \sum_{y \in Y} (1 - \varepsilon_{\oplus, \sim}^X(y, x))$,
- $\alpha, \beta \in [0, +\infty]$ and $\gamma \in ]0, +\infty[$.

The extended Tversky measure between $X$ and $Y$ is:

$$\text{Tve}^{\alpha, \beta, \gamma, \varepsilon_{\oplus, \sim}^X}(X, Y) = \begin{cases} 1 & \text{if } X = Y = \emptyset \\ \alpha a + \beta b + \gamma c & \text{otherwise} \end{cases}$$

Classical similarity measures (see Table 1 in [4] for the definitions) can be obtained with $\alpha = \beta = 2^{-\gamma}$ and $\gamma = 1$ and the classical set-membership. In particular, the Jaccard measure (i.e. $\text{jac}$) is obtained with $n = 0$, Dice (i.e. $\text{dic}$) with $n = 1$, Sorensen (i.e. $\text{soe}$) with $n = 2$, Anderberg (i.e. $\text{adb}$) with $n = 3$, and Sokal and Sneath 2 (i.e. $\text{ss}^2$) with $n = -1$. Under some reasonable assumptions, Tversky measure s.t. $\alpha = \beta = \gamma$.

**Proposition 1** For any $X, Y \subseteq X$, any $\gamma \in ]0, +\infty[$, any membership function $\varepsilon_{\oplus, \sim}^X$ s.t. $\sim$ is symmetric, we have $\text{Tve}^{\alpha, \beta, \gamma, \varepsilon_{\oplus, \sim}^X}(X, Y) = \text{Tve}^{\alpha, \beta, \gamma, \varepsilon_{\oplus, \sim}^X}(Y, X)$, where $\oplus = \oplus, \varepsilon_{\oplus, \sim}^X$.

In the rest of the paper we will focus our study on the membership function using the aggregator function $\max$. Table 1 denotes the set of parametric (non-)symmetric extended versions of the well known similarity extensions, where fixing $\alpha$ and $\beta$ corresponds to choosing among Jaccard, Dice, Sorensen, Anderberg, or Sokal and Sneath.

<table>
<thead>
<tr>
<th>Symmetric Measures</th>
<th>Non-Symmetric Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Tve}^{1/2, 1/2, \varepsilon}(X, Y) = \text{jac}(X, Y)$</td>
<td>$\text{Tve}^{1, 1, \varepsilon}(X, Y) = \text{ns-jac}(X, Y)$</td>
</tr>
<tr>
<td>$\text{Tve}^{1, 0, 1/2, \varepsilon}(X, Y) = \text{dic}(X, Y)$</td>
<td>$\text{Tve}^{1, 0, 1, \varepsilon}(X, Y) = \text{ns-dic}(X, Y)$</td>
</tr>
<tr>
<td>$\text{Tve}^{0, 1, 1, \varepsilon}(X, Y) = \text{ss}^2(X, Y)$</td>
<td>$\text{Tve}^{0, 0, 1, \varepsilon}(X, Y) = \text{ns-ss}^2(X, Y)$</td>
</tr>
</tbody>
</table>

3.3 Similarity between sets of grounded clauses.

Recall that $\mathbb{C}$ is the set of all grounded clauses in $\text{OS} - \text{FOL}$.

**Definition 21 (Grounded clause membership)** Let $\delta \in \mathbb{C}$ be a grounded clause and $\Delta \subseteq \mathbb{C}$ be a set of grounded clauses. Let $\otimes^\mathbb{C}$ and $\oplus^\mathbb{C}$ be two aggregation functions and $\varepsilon^\mathbb{C} = \text{sim}_C^{\delta, \oplus^\mathbb{C}}$ be a similarity measure between a pair of clauses with $\varepsilon^\mathbb{S} = \text{sim}_L^{\delta, \oplus^\mathbb{S}}$. The membership function of a grounded clause in a set of grounded clauses, denoted $\varepsilon_{\delta, \varepsilon^\mathbb{C}}^\Delta : \mathbb{C} \times \mathbb{C} \rightarrow [0, 1]$, is $\varepsilon_{\delta, \varepsilon^\mathbb{C}}^\Delta(\delta, \Delta) = \otimes^\mathbb{C}_{\delta, \varepsilon^\mathbb{C}}(\mathbb{C}(\delta, \Delta'))$.

**Definition 22 (Similarity between sets of grounded clauses)** Let $\varepsilon_{\delta, \varepsilon^\mathbb{S}}^\Delta$ be a membership function with $\varepsilon^\mathbb{S} = \text{sim}_L^{\delta, \oplus^\mathbb{S}}$, and $\varepsilon^\mathbb{S} = \text{sim}_L^{\delta, \oplus^\mathbb{S}}$. A similarity measure between two sets of grounded clauses is defined as $\text{sim}^\mathbb{S}_{\delta, \varepsilon^\mathbb{S}} : \mathbb{C} \times \mathbb{C} \rightarrow [0, 1]$.

**Example 9** Let $\Delta_1$ and $\Delta_2$ be two sets of grounded clauses.

$\Delta_1 = \{w(T), \neg w(T) \lor \neg f(T), \neg w(Z) \lor \neg f(Z)\}$

$\Delta_2 = \{p(T), \neg p(T) \lor \neg f(T), \neg p(Z) \lor \neg f(Z)\}$
Definition 23 (Instantiation membership) Consider an instantiation \( \Delta \in \mathbb{I} \) and a set of instantiations \( I \subseteq \mathbb{I} \). Let \( \oplus_1^{\Delta} \), \( \oplus_2^{\Delta} \) and \( \oplus_3^{\Delta} \) be three aggregation functions and \( s_1^{\Delta} = \text{sim}^{\Delta} \) be a similarity measure between a pair of set of clauses with \( s_2^{\Delta} = \text{sim}^{\Delta} \). The membership function of an instantiation in a set of instantiations, \( s_3^{\Delta} : I \times 2^I \rightarrow [0, 1] \), is \( s_3^{\Delta}(\Delta, I) = \oplus_1^{\Delta}(s_1^{\Delta}(\Delta, \Delta')) \).

Definition 24 (Similarity between sets of instantiations) Let \( s_1^{\Delta} \) be a membership function with \( s_1^{\Delta} = \text{sim}^{\Delta} \) and \( s_2^{\Delta} = \text{sim}^{\Delta} \). The similarity measure between two set of instantiations is defined as \( \text{sim}^{\Delta}(I_1, I_2) : 2^I \times 2^I \rightarrow [0, 1] \).

Example 10 Let \( I_1 \) and \( I_2 \) be two sets of instantiations s.t.: 
\( I_1 = \{ \Delta_1, \Delta_2, \Delta_3 \} \) with 
- \( \Delta_1 = \{ w(T), -w(T) \} \) \( \forall \) \( T \in \text{FOL} \) 
- \( \Delta_2 = \{ w(T), -w(T) \} \) \( \forall \) \( T \in \text{FOL} \) 
- \( \Delta_3 = \{ w(T), -w(T) \} \) \( \forall \) \( T \in \text{FOL} \) 

\( I_2 = \{ p(T), -p(T) \} \) \( \forall \) \( T \in \text{FOL} \) 

\( \text{sim}^{\Delta}(I_1, I_2) = \text{Tve}^{1,1,1,1}(I_1, I_2) = \frac{a}{a+b+c} = \frac{73}{146} \approx 0.50 
\)

4 Axiomatic Evaluation

Before determining the principles satisfied by our similarity measures, we introduce the notion of well-behaved SM. It is a bridge between the (lower level) properties of the measures that we use (e.g. the Tversky measures) and the (higher level) properties of the similarity measure between arguments defined from such a SM.
Definition 27 (Well-Behaved SM)

A SM $M = \langle \sim^L = \text{sim}_L^{(R,P,C)}, s_c = \text{sim}_C^{L_{\alpha,\gamma,\epsilon}}, s_i = \text{sim}_I^{L_{\alpha,\gamma,\epsilon}}, \text{sim}^{\mathbb{L}_{\alpha,\gamma,\epsilon}} \rangle$ is well-behaved iff:

1. (a) $g(1, 1) = 1$,
   - $g(0, 0) = 0$,
2. (b) $p(P, P) = 1$,
   - $p(P, Q) = 0$ if $P \neq Q$.
3. (c) $\forall i \in \{1, \ldots, n\}$ s.t. $a_i = b_j$ then $a_i \in \{1, \ldots, n\}$.

Definition 28 (Well-Behaved SM)

A SM $M = \langle \sim^L = \text{sim}_L^{(R,P,C)}, s_c = \text{sim}_C^{L_{\alpha,\gamma,\epsilon}}, s_i = \text{sim}_I^{L_{\alpha,\gamma,\epsilon}}, \text{sim}^{\mathbb{L}_{\alpha,\gamma,\epsilon}} \rangle$ is well-behaved iff:

1. (a) $g(1, 1) = 1$,
   - $g(0, 0) = 0$,
2. (b) $p(P, P) = 1$,
   - $p(P, Q) = 0$ if $P \neq Q$.
3. (c) $\forall i \in \{1, \ldots, n\}$ s.t. $a_i = b_j$ then $a_i \in \{1, \ldots, n\}$.

Table 2 – Principles satisfaction by similarity measures. \* (resp. o) means the measure satisfies (resp. violates) the principle. $\text{sim}_M$ is a shorthand for $\text{sim}^{\mathbb{L}_{\alpha,\gamma,\epsilon}}$.

<table>
<thead>
<tr>
<th>$\text{sim}_{\text{jac}}$</th>
<th>$\text{sim}_{\text{dic}}$</th>
<th>$\text{sim}_{\text{sor}}$</th>
<th>$\text{sim}_{\text{adb}}$</th>
<th>$\text{sim}_{\text{avg}}$</th>
<th>$\text{sim}_{\text{max}}$</th>
<th>$\text{sim}_{\text{min}}$</th>
<th>$\text{sim}_{\text{dic}}$</th>
<th>$\text{sim}_{\text{sor}}$</th>
<th>$\text{sim}_{\text{adb}}$</th>
<th>$\text{sim}_{\text{avg}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximal</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Symmetry</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Substitution</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Syntax Indep.</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Minimality</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Monotony</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Strict Monotony</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Dominance</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>Strict Dominance</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
<td>(c)</td>
</tr>
</tbody>
</table>

Theorem 2 Let $M \in \mathbb{M}$ be a well-behaved and $\text{sim}_{\text{Avr}(L)}$ be a similarity measure based on $M$.

- $\text{sim}_{\text{M}_{\text{Avr}}}$ satisfies Symmetry (resp. Syntax Independence) if all the functions in $M$ are symmetric (resp. syntax independent).
- $\text{sim}_{\text{M}_{\text{Avr}}}$ satisfies Strict Monotony and Strict Dominance if it satisfies condition 2(c') : consider $X_0, X_1, X_2 \subseteq \mathbb{X}$ s.t. $X_1 \subseteq X_2$ and $X_2 \setminus X_1 = \{x_2\}$. If $\sim^{\mathbb{L}_{\alpha,\gamma,\epsilon}}(X_0, X_1) < 1$ and $\exists x_0 \in X_0 \setminus X_1 = \{x_1\}$ s.t. $\sim^{\mathbb{L}_{\alpha,\gamma,\epsilon}}(X_0, x_0) = 0$ then $\sim^{\mathbb{L}_{\alpha,\gamma,\epsilon}}(X_0, X_2) > \sim^{\mathbb{L}_{\alpha,\gamma,\epsilon}}(X_0, X_1)$.

We extend some results from [4].

Proposition 3 Let $\text{sim}_{\text{Avr}(L)}$ be a similarity measure.

- Consider $A, B \in \text{Arg}(L)$. If $\text{sim}_{\text{Avr}(L)}$ satisfies Maximality, Monotony, Strict Monotony and Strict Dominance then $A \simeq B$ iff $\text{sim}_{\text{Avr}(L)}(A, B) = 1$.
- If $\text{sim}_{\text{Avr}(L)}$ satisfies Symmetry, Maximality, Strict Monotony, Dominance, and Strict Dominance then $\text{sim}_{\text{Avr}(L)}$ satisfies Substitution.

Let us prove that the functions $g, p$ and $c$ used in the paper satisfy the expected properties of a well-behaved SM.

Lemma 1 For $g \in \{\text{min}, \text{avg}\}$, $p = \text{eq}$ and $c = \text{pws}$, $(g, p, c)$ satisfies item 1. of Def. 27.

We can show similar results for the Tversky measures that we use to define $\text{sim}_{\mathbb{C}^{L_{\alpha,\gamma,\epsilon}}, \text{sim}_{\mathbb{L}^{L_{\alpha,\gamma,\epsilon}}}}$. We consider the measures described in Table 1.

Lemma 2 If $\text{Tve}^{\alpha, \beta, \gamma, \epsilon, \delta_{\text{Avr}}}$ is a Tversky measure, with $\oplus = \max$, and $\text{sim} is$

- either $\text{sim}_{\mathbb{L}^{(R,P,C)}}$ (from Definition 1) s.t. $(g, p, c)$ satisfies item 1. of Def. 27,
- or a similarity measure satisfying the item 2. of Def. 27,
then $\text{Tve}^{\alpha, \beta, \gamma, \epsilon, \delta_{\text{Avr}}}$ satisfies the item 2. of Def. 27.

Proposition 4 For $x \in \{\text{jac}, \text{dic}, \text{sor}, \text{adb}, \text{ss}_2, \text{ns-jac}, \text{ns-dic}, \text{ns-sor}, \text{ns-adb}, \text{ns-ss}_2\}$, define $\text{sim}_{\text{Avr}(L)}$. Then define the similarity model $\text{SM}_{\text{Avr}} = \langle \text{sim}_{\text{min}}, \text{eq}, \text{pws} \rangle$, $x^2, \text{sim}_{\mathbb{L}_{\alpha,\gamma,\epsilon}}, x^1, \text{sim}_{\mathbb{C}_{\alpha,\gamma,\epsilon}}, \text{sim}_{\mathbb{L}_{\alpha,\gamma,\epsilon}}\rangle$. The satisfaction of principles by the measures is given in Table 2.
Notice that Proposition 4 implies that all the principles are compatible. Moreover with the result of item 1 of Proposition 3, we can deduce that our 5 symmetric extended Tversky measures satisfying a stronger form of maximality, since equivalent arguments are maximally similar. For non-symmetric measures, we show that they can obtain full similarity in a particular case of sub-argument.

**Proposition 5** Let \( A, B \in \text{Arg}(L) \) be two arguments. Assume that \( M \) is a SM s.t. \( \text{sim}_M^{\alpha,L}, \text{sim}_M^{\alpha,L} \) and \( \text{sim}_M^{\alpha,L} \) are Tversky measures s.t. \( \alpha \neq \beta \) for at least one of them (i.e. it is non-symmetric). If \( B \) is a sub-argument of \( A \), then \( \text{sim}_{M_{\alpha,L}}^{\alpha,L}(A,B) \geq \eta \). Moreover, if \( \text{UC}(\text{Conc}(B)) \subseteq \text{UC}(\text{Conc}(A)) \), then \( \text{sim}_{M_{\alpha,L}}^{\alpha,L}(A,B) = 1 \).

### 5 Conclusion

In this paper, we have proposed the rich methodology of similarity models which are able to express large families of similarity measures between Order-Sorted First Order Logic (OS – FOL) arguments, thanks to various parameters which allow to define generalized versions of similarity measures from the literature. For the first time in the logical argumentation literature, we define non-symmetric similarity measures. A set of nine principles for these OS – FOL arguments has been proposed with a set of well-behaved properties ensuring some principles. We have shown that our symmetric measures satisfy all the principles, while their non-symmetric counterparts only satisfy a subset.

This work paves the way to several interesting research questions. First of all, we can consider additional measures (e.g. Ochiai [25], Kulczynski [20]) and principles (e.g. triangular inequality, non-zero, independent distribution [14]) to allow a more accurate comparison of similarity measures. Another research line could be to consider situations where different predicates are partially similar. For instance, one can consider that \( \text{greaterOrEqual}(A,B) \) is somehow similar to \( \text{strictlyGreater}(A,B) \). Following the same idea as in [6], we also plan to use our similarity measures as a parameter of acceptability semantics. Finally, we want to apply our work on real data expressed in fragments of OS – FOL.

### 6 Acknowledgement

This work benefited from the support of the project AG-GREEY ANR-22-CE23-0005 of the French National Research Agency (ANR) and the Project GIUSTIZIA AGILE, CUP J89J22000900005.

### Références


