Defining and Combining Symmetric and Asymmetric Similarity Measures

Derek G. Bridge[⋆]

Department of Computer Science,
University College, Cork,
Ireland
d.bridge@cs.ucc.ie
http://www.cs.ucc.ie/dbridge.html

Abstract. In this paper, we present a framework for the definition of similarity measures using lattice-valued functions. We show their strengths (particularly for combining similarity measures). Then we investigate a particular instantiation of the framework, in which sets are used both to represent objects and to denote degrees of similarity. The paper concludes by suggesting some generalisations of the findings.

1 Introduction

There are many different ways of computing the similarity of object representations. These include:

- the *feature-based approach*, in which objects are represented by sets of features, and similarity is based on feature commonality and difference (e.g. [13]);
- the *geometric approach*, in which objects are represented by points in an n-dimensional space (usually specified by sets of pairs of attributes and atomic values), and similarity is based on the inverse of the distance between objects in the space (e.g. [12]); and
- the *structural approach*, which uses graphical representations, in which nodes denote objects and edges denote relations between objects, and similarity is based on graph matching (e.g. [3]).

This tri-part classification is an over-simplification. It may omit distinctive approaches. And it may seemingly drive wedges between related approaches. For example, work on the similarity of first-order terms ([4], [5]) and feature structures ([10]) has elements of both the geometric and the structural approaches. We include the classification merely in order to indicate the huge variety of approaches to be found in the CBR literature and elsewhere.

This paper discusses a general mathematical framework in which a wide variety of similarity measures can be defined and combined. Having reported

[⋆] This paper develops earlier work on lattice-based functions done with Hugh Osborne, to whom my thanks.

examples that are in the spirit of the geometric approach in previous papers [7], [8], in this paper we focus more on some feature-based and structural approaches. Furthermore, where our previous papers focussed on defining the framework's operators, this paper mentions axiomatisation (although it does not aim to give a full axiomatisation) and shows that even seemingly reasonable axiomatisations reported in the literature (e.g. [6]) may be too restrictive.

2 Lattice-Valued Functions

We will be explaining what we mean by *similarity functions, difference functions* and *excess functions.* We will denote an arbitrary similarity function by $\sim$, an arbitrary distance function by $\leftrightsquigarrow$, and an arbitrary excess function by $\trianglelefteq$.

2.1 Similarity Functions

A similarity function takes in two object representations and delivers a value that denotes the similarity of the two representations. Similarity functions reported in the literature have typically delivered values of type boolean or of some numeric type. (For example, there are only boolean-valued and real-valued functions in Richter's classification of similarity functions [11].) We refer to boolean-valued similarity functions, $\sim :: \alpha \rightarrow \alpha \rightarrow \mathsf{Bool}$, as *absolute similarity functions*: two objects are judged to be either similar or not similar. While absolute similarity functions are simple, they do not correspond particularly well to people's intuitive concept of similarity, in which there is the notion of *degrees* of similarity.

To overcome this disadvantage of absolute measures, most similarity functions are numeric-valued, e.g. $\sim :: \alpha \rightarrow \alpha \rightarrow \Re$, often the result-type being restricted to some interval, e.g. $\sim :: \alpha \rightarrow \alpha \rightarrow [0, 1]$. We call these *relative similarity functions.* These seem to serve many purposes well, but there are reasons to be dissatisfied with them. In particular, on occasion, the numbers used are arbitrary. This occurs when similarity function designers need something richer than absolute similarity, i.e., they need degrees of similarity, but they do not need to quantify, or cannot properly quantify, the degrees. Any numbers used in these circumstances will be contrived.

In earlier work [7], [8], [9], we generalised over absolute and relative similarity functions giving what we called *metric similarity functions.* These are lattice-valued functions, $\sim :: \alpha \rightarrow \alpha \rightarrow \mathcal{L}$. A lattice, $\mathcal{L}$, is a partially-ordered set of values, $(S, \sqsubseteq)$, that satisfies certain properties. In fact, we require *complete lattices.* A complete lattice satisfies further properties, a consequence of these further properties being that the set has unique largest and smallest elements.[1] We will not give any further characterisation of lattices or complete lattices. A good understanding of this paper is possible for those who know what a partial order

[1] The requirement for complete lattices can be weakened but only by outlawing certain operators, such as prioritisation (see later), which require complete lattices. (In fact, to be more accurate, an axiom that excess functions must satisfy (Table 1) will only be provably satisfied if excess functions have complete lattices as their return-types.)

is, even if they do not know what a (complete) lattice is, and we proceed on this basis. See [2] for mathematical definitions.

Ours is not the first use of lattices in CBR. But, in our framework and in the work reported in [6] and [10], it is the degrees of similarity (the values of the *result-type* of the similarity function) that form a lattice, whereas, in earlier work such as [1] (and also in machine learning work), it is the cases in the case base (the values of the *arguments* of the similarity function) that form a lattice.

We claim that the advantages of our metric framework are that: it subsumes absolute and relative measures (i.e., these are instantiations of the framework); it introduces (again as instantiations) many other ways of measuring similarity (only a few of which other researchers have reported in the literature); and it allows the easy combination of similarity functions. We exemplify these points below.

Consider an absolute similarity function, $\sim :: \alpha \rightarrow \alpha \rightarrow \mathsf{Bool}$. Its counterpart in our framework is simply $\sim' :: \alpha \rightarrow \alpha \rightarrow (\mathsf{Bool}, \sqsubseteq_{\mathsf{Bool}})$, which has a *lattice* as its return-type, where $\sqsubseteq_{\mathsf{Bool}}$, for example, ranks **False** less than **True**.[2] The function $\sim'$ basically computes the same boolean as $\sim$, but its result values are now ordered. Similarly, a relative similarity function, $\sim :: \alpha \rightarrow \alpha \rightarrow [0, 1]$, might have as its counterpart $\sim' :: \alpha \rightarrow \alpha \rightarrow ([0, 1], \leq)$ (assuming small numbers denote lower similarity than higher numbers).[3]

Our claim that our framework subsumes many other ways of measuring similarity is exemplified by the set-valued functions that we discuss in Section 3 and by the following example. Suppose we let $\mathcal{H}$ be the set of linguistic hedges $\{$*very, quite, fairly, barely*$\}$ and we let $\sqsubseteq_{\mathcal{H}}$ denote the ordering depicted as a Hasse diagram in Fig. 1,[4] then we can construct a new kind of similarity function, one that is linguistic-hedge-valued: $\sim :: \alpha \rightarrow \alpha \rightarrow (\mathcal{H}, \sqsubseteq_{\mathcal{H}})$. This is a four-valued function, whose values are only partially ordered; it might be useful to a system designer who needs something richer than absolute similarity, but would find it difficult or misleading to quantify degrees of similarity.

$$\begin{tikzcd}
 & \textit{very} & \\
\textit{quite} \arrow[ur] & & \textit{fairly} \arrow[ul] \\
 & \textit{barely} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Fig. 1. The ordering $\sqsubseteq_{\mathcal{H}}$

[2] In other words, $\sqsubseteq_{\mathsf{Bool}} = \Rightarrow$.

[3] Where we have $\sim :: \alpha \rightarrow \alpha \rightarrow \Re$, the counterpart would need to be a function returning values from a lattice defined on, e.g., $\Re \cup \{\infty, -\infty\}$. The special values ∞ and $-\infty$ are needed to give a complete lattice, having topmost and bottommost elements.

[4] $\sqsubseteq_{\mathcal{H}}$ is intended only by way of a simple example. We have no investment in, or commitment to, the particular ordering shown.

Finally, consider the combination of two similarity functions. For concreteness, and to illustrate the advantages of our approach, let one be numeric-valued, $\sim_1 :: \alpha \rightarrow \alpha \rightarrow ([0,1], \leq)$, and the other be linguistic-hedge-valued, $\sim_2 :: \beta \rightarrow \beta \rightarrow (\mathcal{H}, \sqsubseteq_{\mathcal{H}})$. We want a similarity function defined on pairs drawn from $\alpha \times \beta$. But what should the return-type be? It is dealing with this problem that leads others to inter-convert similarity functions (e.g. see [14] for ways of combining boolean-valued and numeric-valued functions, all of which involve converting the boolean-valued function to a numeric one). But, in our framework, a similarity function may have as its result-type *any* type on which a (complete) lattice is defined. We can have a function that has as its result-type number-hedge pairs, $([0,1] \times \mathcal{H})$, so the degree of similarity between two objects might be, e.g., $\langle 0.2, \textit{very} \rangle$, $\langle 0.67, \textit{quite} \rangle$, $\langle 0.2, \textit{barely} \rangle$, etc. These pairs *are* the degrees of similarity. All that is additionally required is to know how the pairs are ordered, i.e., is $\langle 0.2, \textit{very} \rangle$ less than, greater than or incomparable with $\langle 0.67, \textit{quite} \rangle$?

In principle, we can define any ordering on these pairs that we want (provided we obtain a complete lattice). In practice, it often makes sense to define the new ordering from the orderings on the constituents of the pair. We show two (of many) ways of doing this: *product* and *prioritisation.*

Consider lattices $\mathcal{L}_1 \triangleq (S_1, \sqsubseteq_1)$ and $\mathcal{L}_2 \triangleq (S_2, \sqsubseteq_2)$. The *product* is the lattice $(S_1 \times S_2, \sqsubseteq_\times)$ where

$$\langle x_1, x_2 \rangle \sqsubseteq_\times \langle y_1, y_2 \rangle \triangleq x_1 \sqsubseteq_1 y_1 \wedge x_2 \sqsubseteq_2 y_2$$

Using product, $\langle 0.2, \textit{barely} \rangle$ is less than $\langle 0.2, \textit{very} \rangle$ and it is less than $\langle 0.67, \textit{quite} \rangle$. The *prioritisation* of $\mathcal{L}_1$ over $\mathcal{L}_2$ is the lattice $(S_1 \times S_2, \sqsubseteq_{>>})$, where

$$\langle x_1, x_2 \rangle \sqsubseteq_{>>} \langle y_1, y_2 \rangle \triangleq x_1 \sqsubset_1 y_1 \vee (x_1 = y_1 \wedge x_2 \sqsubseteq_2 y_2)$$

i.e., the first ordering takes priority but, in the event of ties in the first ordering, the second ordering is used. Using prioritisation in this way, $\langle 0.2, \textit{barely} \rangle$ is less than $\langle 0.2, \textit{very} \rangle$ which is less than $\langle 0.67, \textit{quite} \rangle$.

This ability to combine different kinds of similarity function, without inter-conversion, is one of the most striking aspects of our framework. We should also add that product and prioritisation are not our only ways of combining and manipulating lattice-valued functions. Some other operators for manipulating such functions are defined in [7], [8].

Unaddressed by our discussion so far is whether the similarity functions we define in this (or other) frameworks must satisfy any axioms. Giving full axiomatisations is not an aim of this paper, but some discussion of axioms is needed because any ways we define for building new functions from existing ones (such as using products and prioritisations, above) should ensure that the new functions will satisfy the axioms. (For example, an operator similar to product, but based on disjunction rather than conjunction, is unlikely to violate any axioms we might place on similarity functions but will not necessarily respect the axiom that we give in Table 1 for excess functions.)

Many authors claim that similarity functions must be *reflexive* [11], which, in our framework, would be stated as $x \sim y = \top$, where $\top$ is the topmost element in the lattice. Tversky [13] argues that this is too strong: there may be different degrees of similarity for different 'identical' objects. Consequently, in [9] and here (see Table 1), we use a weaker axiom: $x \sim x \sqsupseteq x \sim y$, i.e., an object is at least as similar to itself as it is to any other object.

Even more controversial is *symmetry*, $x \sim y = y \sim x$. Richter argues for it [11]; Jantke argues against it (but, ironically, he does so in a paper in which the new similarity function he introduces is symmetric) [5]; Tversky argues against it [13]; in [9], we took symmetry as an axiom and defined similarity measures in a way that guaranteed symmetry. But, in this paper, we do not adopt symmetry as an axiom, and it is a contribution of this paper that we show that we can define both asymmetric and symmetric similarity measures in our framework. This is quite rare in CBR and so we believe it to be a strength of our framework.

In accord with much work on similarity, we do not require *transitivity*. However, there may be a need for further axioms to apply in certain instantiations of our framework. For example, for similarity measures defined on structured object representations, such as those discussed in Sections 3 and 4 of this paper, a *monotonicity* axiom might be needed.

2.2 Difference Functions

While this discussion has focused on similarity functions, our generalisation to lattice-valued functions can be applied to other kinds of function too. Difference functions,[5] for example, can be boolean-valued, are more often numeric-valued and might justifiably be generalised to being lattice-valued, with the same motivations as for similarity functions.

It is interesting to note the relationship in our framework between difference functions and similarity functions. It is not uncommon to invert difference functions to give similarity functions. In our formalisation, inversion is arguably more straightforward than in other frameworks. Given a difference function $\leftrightsquigarrow$ with result lattice $(S, \sqsubseteq)$, we do not try to find some inverse operator $\leftrightsquigarrow^{-1}$ (based on reciprocals or the like). We simply invert the lattice: $(S, \sqsubseteq^{-1})$ or $(S, \sqsupseteq)$. What this function computes will remain unchanged; the way we interpret its results will differ. What was denoting a large difference will now denote a small similarity. We will see concrete examples of this later in this paper.

Everything else we said about lattice-valued similarity functions applies to lattice-valued difference functions, e.g. they are easily combined by producting or prioritising their result lattices. Their axioms are as for similarity functions.

2.3 Excess Functions

It is also common to use orderings in Computer Science and AI. (Within AI, they are used, for example, to model an agent's preferences, in which case they are referred to as 'preference relations'.) These are boolean-valued functions, $\trianglelefteq :: \alpha \rightarrow \alpha \rightarrow \mathsf{Bool}$. But we can easily imagine wanting to use numeric-valued

[5] Elsewhere difference functions are sometimes called 'distance functions' (e.g. [11]).

counterparts to these, $\trianglelefteq :: \alpha \rightarrow \alpha \rightarrow \Re$, where we want to quantify the degree by which one object is exceeded by (or preferred over) another. And so we can also use our generalisation to give lattice-valued functions, $\trianglelefteq :: \alpha \rightarrow \alpha \rightarrow \mathcal{L}$: a value from a lattice denotes the degree by which the second object exceeds the first. We have used such functions in our work [7], [8], and will use them in this paper, referring to them as *excess functions*. If $x \trianglelefteq y = d$, we say 'y exceeds x by d'. (This now means that, in AI, we can generalise preference relations so that they deliver the degree to which one object is preferred over another.)

Combining different excess functions is therefore as straightforward as combining similarity and difference functions, e.g. using product and prioritisation. But where this discussion differs from the two above is that there is an important axiom to be satisfied. The axiom is inspired by triangle inequality. It is given, along with other axioms, in Table 1.

Type	Axioms
Similarity $\alpha \rightarrow \alpha \rightarrow (S, \sqsubseteq)$	$x \sim x \sqsupseteq x \sim y$
Difference $\alpha \rightarrow \alpha \rightarrow (S, \sqsubseteq)$	$x \leftrightsquigarrow x \sqsupseteq x \leftrightsquigarrow y$
Excess $\alpha \rightarrow \alpha \rightarrow (S, \sqsubseteq)$	$((x \trianglelefteq y \sqsubseteq y \trianglelefteq x) \wedge (y \trianglelefteq z \sqsubseteq z \trianglelefteq y)) \Rightarrow (x \trianglelefteq z \sqsubseteq z \trianglelefteq x)$

Table 1. Types and axioms

We leave open the question of whether excess functions should (like preference relations) be expected to satisfy (generalised versions of) reflexivity and antisymmetry. We have not found a mathematical motivation for imposing such axioms. For example, our definition of the maxima of a set under an excess function [7] satisfies the properties we expect of maxima without imposing reflexivity and antisymmetry.

In the rest of this paper, we apply our framework to particular object representations, specifically to structured object representations that have a natural partial ordering. For concreteness and simplicity, we look at sets first, and then we generalise our findings.

3 Set-Based Object Representations

Consider representing objects as sets of features or properties which those objects possess. For concreteness, suppose that the set of all possible (or relevant) features is $\{a, b, c, d\}$ and that we have three objects O_1, O_2 and O_3 where $O_1 \triangleq \{a, b, c\}, O_2 \triangleq \{a, b, d\}$ and $O_3 \triangleq \{a, d\}$.

Clearly, *one way* of computing the similarity of two such objects would be to compute the intersection of their set representations, giving the features they have in common. The cardinality of the intersection (or a form of member-counting that is sensitive to the identity of the members of the set, so that some features can be more important than others) might then be used to denote the degree of similarity. We can do all this in our framework since the result-type of a function can be a lattice on some set of numeric values. But here we investigate an alternative.

We can instead let the set intersections themselves denote the degrees of similarity. Hence, not only will objects be represented by sets, but their degrees of similarity will be denoted by sets. The similarity of O_1 and O_2 *is* $\{a, b\}$, that of O_1 and O_3 is $\{a\}$, and that of O_2 and O_3 is $\{a, d\}$. But which is the most similar pair? In other words, of the degrees of similarity we have computed, $\{a, b\}$, $\{a\}$ and $\{a, d\}$, which is (or are) the highest? We need an ordering on sets. Since larger sets denote more commonality and so more similarity, the obvious ordering is the usual subset ordering, $\subseteq$. This means that O_1 and O_2 are more similar than are O_1 and O_3. However, O_1 and O_2 are neither more nor less similar than O_2 and O_3.

Given a universal set, in the example this being $\{a, b, c, d\}$, we clearly have another instance of our framework. The ordering $\subseteq$ defined on the power set of $\{a, b, c, d\}$ gives us a (complete) lattice, having $\{a, b, c, d\}$ as its topmost element and $\emptyset$ as its bottommost element. The similarity function has type $\sim :: \mathcal{P}(\{a, b, c, d\}) \to \mathcal{P}(\{a, b, c, d\}) \to (\mathcal{P}(\{a, b, c, d\}), \subseteq)$ and is defined by $x \sim y \triangleq x \cap y$. (Obviously, a corresponding difference function, $\leftrightsquigarrow$, can be defined in which $x \leftrightsquigarrow y \triangleq x \cap y$ but where the lattice used is $(\mathcal{P}(\{a, b, c, d\}), \supseteq)$.)

This idea, of using a structured object to denote the degree of similarity of two other structured objects, and its applications to sets (as well as to other structured representations) is proposed by Matuschek & Jantke [6]. Their concern, however, is axiomatisation and so they separate a Common Substructure axiom ($x \sim y \subseteq x \wedge x \sim y \subseteq y$, i.e., the object that denotes the degree of similarity should somehow respect the common substructure) from an optional Maximality axiom ($(z \subseteq x \wedge z \subseteq y) \Rightarrow z \subseteq x \sim y$).[6] Clearly, set intersection is a similarity function that satisfies both these axioms.

What is of more interest is to go beyond their work by showing some natural similarity functions for sets which do not satisfy their axioms. We will show one that does not satisfy the Common Substructure axiom, and one which does not use simple sets at all to denote degrees of similarity (but does not resort to using numbers). We will also show one which is not symmetric.[7]

Not Respecting the Common Substructure Axiom. Before defining a similarity function that does not respect the Common Substructure axiom, let's look at some other functions that could usefully be set-valued.

We can define an excess function that takes in two sets and delivers a set as its result-type. For example, we could base the definition on relative complement (set difference). In this case, $\trianglelefteq :: \mathcal{P}(\{a, b, c, d\}) \to \mathcal{P}(\{a, b, c, d\}) \to (\mathcal{P}(\{a, b, c, d\}), \subseteq)$ and $x \trianglelefteq y \triangleq y \backslash x$. The degree to which O_1 exceeds O_3 is $\{b, c\}$, O_3 exceeds O_1 by

[6] Matuschek & Jantke present their axioms in a form that is more general than that used here. Furthermore, we have included here only one of two competing maximality axioms that they discuss.

[7] We should make clear that symmetry is not an axiom in [6].

$\{d\}$, and so on. This definition satisfies the generalised 'triangle equality' axiom (Table 1).

A difference function can also be defined, the obvious definition being the disjoint union, $\uplus$, of the two sets (i.e., the elements they do not have in common). The type is $\leftrightsquigarrow :: \mathcal{P}(\{a,b,c,d\}) \to \mathcal{P}(\{a,b,c,d\}) \to (\mathcal{P}(\{a,b,c,d\}), \subseteq)$; the definition is $x \leftrightsquigarrow y \triangleq x \uplus y$. The difference between O_1 and O_3, for example, is $\{b,c,d\}$.

We can obtain difference functions by bringing together two excess functions. The difference between x and y is some combination of the excess of x over y and the excess of y over x. Specifically, disjoint union is the union of the two excesses (where excess is computed using relative complement in each case): $x \uplus y \triangleq x\backslash y \cup y\backslash x$.

From this difference function, we get another way to measure similarity. Following a tradition in CBR and elsewhere, we invert the difference function. And, as we saw earlier in this paper, this is easily done: we simply invert the result lattice. This gives us a similarity function whose type is $\sim :: \mathcal{P}(\{a,b,c,d\}) \to \mathcal{P}(\{a,b,c,d\}) \to (\mathcal{P}(\{a,b,c,d\}), \supseteq)$, and where $x \sim y \triangleq x \uplus y$. (Equivalently, $x \sim y \triangleq x\backslash y \cup y\backslash x$.) Thus the similarity of O_1 and O_2 is $\{c,d\}$ and that of O_1 and O_3 is $\{b,c,d\}$, which means (remembering that the result lattice is ordered such that smaller sets denote higher similarity) that O_1 and O_2 are more similar, by this measure, than O_1 and O_3.

Not Using Sets to Denote Degrees of Similarity. In the paper by Matuschek & Jantke [6], the axioms require the degree of similarity to be expressed using the same structured representation as is used for the objects. This is too restrictive. Natural similarity functions can be defined that do not observe this requirement (without resorting to conversion to numeric-valued functions and the like).

For example, suppose we want a similarity function that combines the two similarity functions we have defined so far (the one based on shared features, and the one based on the inverted distance function using non-shared features). We exploit the flexibility of our framework by defining a similarity function that has as its result-type *pairs of sets* (much as we combined numeric-valued and linguistic-hedge-valued functions earlier, also resulting in a function whose result-type was pairs). The type of the similarity function will be $\sim :: \mathcal{P}(\{a,b,c,d\}) \to \mathcal{P}(\{a,b,c,d\}) \to (\mathcal{P}(\{a,b,c,d\}) \times \mathcal{P}(\{a,b,c,d\}), \sqsubseteq)$ and the definition of $\sim$ will be $x \sim y \triangleq \langle x \cap y, x \uplus y \rangle$. O_1 and O_2 are similar to degree $\langle \{a,b\}, \{c,d\} \rangle$ (their intersection paired with their disjoint union); O_1 and O_3 are similar to degree $\langle \{a\}, \{b,c,d\} \rangle$. How is the ordering on these pairs, $\sqsubseteq$, defined? As when we combined similarity functions earlier, it makes sense to define the new ordering from the constituent orderings. The ordering for the first element of the pair is $\subseteq$. The ordering for the second element is $\supseteq$, since this is where large sets denote big difference and so low similarities. Then, if shared and non-shared features are equally important, we might use product, $\sqsubseteq \triangleq \subseteq \times \supseteq$ (i.e., $\langle x_1, x_2 \rangle \sqsubseteq \langle y_1, y_2 \rangle$ iff $x_1 \subseteq y_1 \wedge x_2 \supseteq y_2$); if shared features are more important than non-shared ones, we might prioritise $\subseteq$ over $\supseteq$, $\sqsubseteq \triangleq \subseteq >> \supseteq$ (i.e., $\langle x_1, x_2\rangle \sqsubseteq \langle y_1, y_2\rangle$ iff $x_1 \subset y_1 \vee (x_1 = y_1 \wedge x_2 \supseteq y_2)$); if non-shared features are more important than shared ones, we might prioritise $\supseteq$ over $\subseteq$.

Asymmetric Similarity. The three similarity functions we have shown so far in this section (based on intersection, disjoint union with inverted lattice, and the two combined) have been symmetric. Using Tversky's paper [13] for inspiration, we can also show a way of building up asymmetric similarity functions in our framework. One example is $x \sim y \triangleq \langle x \cap y, x \backslash y, y \backslash x\rangle$. This function is not symmetric. The similarity of O_1 and O_2 is $\langle\{a, b\}, \{c\}, \{d\}\rangle$ but that of O_2 and O_1 is $\langle\{a, b\}, \{d\}, \{c\}\rangle$.[8] We need to place an ordering on these triples and it should be clear by now, without giving details, that we will probably combine the constituent orderings, $\subseteq, \supseteq$ and $\supseteq$, using products, prioritisations and the like.

Tversky's Model. In [13], Tversky was also axiomatising similarity functions for set-valued object representations. His first axiom (the Matching axiom) is that $x \sim y = F(x \cap y, x \backslash y, y \backslash x)$. For the most part, his remaining axioms constrain the function F, aiming to guarantee that F will deliver a suitable numeric-valued similarity function. One way of thinking about this is to regard Tversky as trying, in some sense, to inter-convert from one way of denoting degrees of similarity (triples of sets) to another (numbers).

Of course, there is no reason why we should not do the same, if we so desire it. Our framework accommodates both functions that deliver triples of sets and also functions that deliver numbers. We could define a function that takes a lattice defined on triples of sets and produces another lattice defined on numbers, thus allowing us to map from one kind of similarity function to another.[9]

This section has focused on sets. In our Concluding Remarks, we generalise this section's findings to apply to other structured representations.

4 Concluding Remarks

Nothing hinges on our use of sets to exemplify the previous section of this paper. The main ideas are the use of lattice-valued functions, and specifically their use on structured representations. Just as Matuschek & Jantke [6] present their axioms in a form that is not restricted to sets but applies to any structured representation that has a natural partial ordering, we ought to generalise our similarity measures for sets to other representations (in our case, ones that form

[8] Another way of getting possibly asymmetric functions is to compute the excess of y over x differently from the way we compute the excess of x over y (e.g. by weighting one more highly than the other). Then, a measure that combines the two may not be symmetric.

[9] See the definition of 'right composition' in [7], [8], which has precisely this purpose (although it is defined there in a form that is specific to excess functions).

complete lattices). The generalisation continues to assume that the object representation has a natural partial ordering (forms a complete lattice) and can therefore also be used to denote the degrees of similarity.

Suppose we have an arbitrary complete lattice, $\mathcal{L} \triangleq (S, \sqsubseteq)$. Let $\top \in S$ denote the topmost element in the ordering and $\bot \in S$ denote the bottommost. The meet, greatest lower bound (glb) or infimum of $x \in S$ and $y \in S$, $x \sqcap y$, is the unique element $z \in S$ such that

$$z \sqsubseteq x \wedge z \sqsubseteq y \wedge \forall z'((z' \sqsubseteq x \wedge z' \sqsubseteq y) \Rightarrow z' \sqsubseteq z)$$

The join, least upper bound (lub) or supremum of $x \in S$ and $y \in S$, $x \sqcup y$, is the unique element $z \in S$ such that

$$x \sqsubseteq z \wedge y \sqsubseteq z \wedge \forall z'((x \sqsubseteq z' \wedge y \sqsubseteq z') \Rightarrow z \sqsubseteq z')$$

(In the case of sets where $S \triangleq \mathcal{P}(\{a, b, c, d\})$ and $\sqsubseteq \triangleq \subseteq$, then $\top = \{a, b, c, d\}$, $\bot = \emptyset, \sqcap = \cap$ and $\sqcup = \cup$.) In a complete lattice, we are guaranteed that the glb and lub will exist.

Since several of the ways of defining similarity make use of excess functions, a useful early step is the definition of an excess function. In general, the object representation type and the result-type may be different, $\trianglelefteq :: \alpha \rightarrow \alpha \rightarrow (S, \sqsubseteq)$, $\alpha \neq S$, and we can only advise that the excess function satisfy the axiom in Table 1. However, where the argument and result-types are the same (as we are assuming for now), it may be that some analogue of relative complement (set difference) will form a useful definition for an excess function. One possible general schema for this is:

$$x \trianglelefteq y \triangleq \sqcap \{z \mid z \sqsubseteq y \wedge z \sqcap x = \bot\}$$

(the greatest lower bound of the substructures of y that exclude information that is in x). (Instantiating this for sets gives $x \trianglelefteq y = y \backslash x$, as desired.)

Once excess functions are defined, we can define at least four similarity functions that correspond to the four developed in the previous section. They are given in Table 2, in which f_1 and f_2 denote functions for combining orders, such as product or prioritisation. (Obviously, corresponding difference functions can be defined by inverting the lattices.)

Definition	Lattice
$x \sim y \triangleq x \sqcap y$	$(S, \sqsubseteq)$
$x \sim y \triangleq x \trianglelefteq y \sqcup y \trianglelefteq x$	$(S, \sqsupseteq)$
$x \sim y \triangleq \langle x \sqcap y, x \trianglelefteq y \sqcup y \trianglelefteq x \rangle$	$(S \times S, f_1(\sqsubseteq, \sqsupseteq))$
$x \sim y \triangleq \langle x \sqcap y, x \trianglelefteq y, y \trianglelefteq x \rangle$	$(S \times S \times S, f_2(\sqsubseteq, \sqsupseteq, \sqsupseteq))$

Table 2. Similarity functions

The similarity function schemata in the Table may not be useful in all cases. They should not be used blindly. They may apply usefully only to certain object representations. For certain representations (e.g. first-order terms/feature structures), $x \trianglelefteq y \triangleq \{z \mid z \sqsubseteq y \wedge z \sqcap x = \bot\}$ might give a non-singleton set of results, which is why we take the glb: $x \trianglelefteq y \triangleq \sqcap\{z \mid z \sqsubseteq y \wedge z \sqcap x = \bot\}$. But, taking the glb in these cases, may lose information. Indeed, it may even be the case that the result will (often or always) be $\bot$. We would not have a satisfactory definition of an excess function in these cases and, therefore, the three similarity functions based on excess functions would be equally unsatisfactory.

Even when useful excess functions can be so-defined, it may be that $x \trianglelefteq y \sqcup y \trianglelefteq x$ gives $\bot$. This will be the case when $\trianglelefteq$ has been defined in a way that means that $x \trianglelefteq y$ and $y \trianglelefteq x$ give conflicting information; then, their combination using $\sqcup$ will give $\bot$. In these cases (which include first-order terms/feature structures), the second and third similarity functions in the Table would be unsatisfactory.

There remain many options not covered by the table, e.g. using a different excess function for the excess of y over x from that used for x over y, or applying functions to convert the lattice to one defined on numbers, and so on. It should also be remembered that these are definitions for the case where the object representation is also suitable for use as the representation of the degrees of similarity. This is what allows $x \sqcap y$ (or some substructure of $x \sqcap y$) to be suitable as (part of) the definition of similarity. It is important to remember that our framework is not confined to such scenarios: there is no reason why the object representation and the degree of similarity should be the same. Even then, the Table is suggestive of ways of building symmetric and asymmetric similarity functions.

References

1. Ashley, K.D.: Reasoning with cases and hypotheticals in HYPO, *International Journal of Man-Machine Studies*, vol.(34), pp.753-796, 1991
2. Birkhoff, G.: *Lattice Theory*, American Mathematical Society, 1967
3. Bunke, H. & Messmer, B.T.: Similarity Measures for Structured Representations, in S.Wess, K.-D.Althoff & M.M.Richter (eds.), *Topics in Case-Based Reasoning (Procs. of First European Workshop on Case-Based Reasoning)*, LNAI-837, pp.106-118, Springer-Verlag, 1994
4. Emde, W. & Wettschereck,D.: Relational Instance-Based Learning, in L.Saitta (ed.), *Procs. of Thirteenth International Conference on Machine Learning*, pp.122-130, 1996
5. Jantke, K.P.: Nonstandard Concepts of Similarity in Case-Based Reasoning, in H.-H.Bock, W.Lenski & M.M.Richter (eds.), *Information Systems in Data Analysis: Prospects — Foundations — Applications (Procs. of Seventeenth Annual Conference of GfKl)*, pp.28-43, Springer-Verlag, 1994
6. Matuschek,D. & Jantke,K.P.: Axiomatic Characterization of Structural Similarity for Case-Based Reasoning, in D.D.Dankel (ed.), *Procs. of Florida AI Research Symposium*, Florida AI Research Society, pp.432-436, 1997
7. Osborne. H. & Bridge, D.G.: Similarity Metrics: A Formal Unification of Cardinal & Non-Cardinal Similarity Measures, in D.B.Leake & E.Plaza (eds.), *Case Based Reasoning Research & Development*, pp.235-244, Springer, 1997

8. Osborne, H. & Bridge, D.G.: We're All Going on a Summer Holiday: An Exercise in Non-Cardinal Case Base Retrieval, in G.Grahne (ed.), *Frontiers in Artificial Intelligence and Applications (Procs. of Sixth Scandinavian Conference on Artificial Intelligence)*, pp.209-219, IOS Press, 1997
9. Osborne, H. & Bridge, D.G: Models of Similarity for Case-Based Reasoning, *Procs. of the Interdisciplinary Workshop on Similarity and Categorisation*, pp.173-179, 1997
10. Plaza, E.: Cases as terms: A feature term approach to the structured representation of cases, in M.Veloso & A.Aamodt (eds.), *Case-Based Reasoning Research and Development (Procs. of First International Conference on Case-Based Reasoning)*, LNAI-1010, pp.265-276, Springer, 1995
11. Richter, M.M.: Classification and Learning of Similarity Measures, in Opitz, Lausen and Klar (eds.), *Studies in Classification, Data Analysis and Knowledge Organization*, Springer-Verlag, 1992
12. Shepard,R.N.: Toward a Universal Law of Generalization for Psychological Science, *Science*, vol.237, pp.1317-1323, 1987
13. Tversky, A: Features of Similarity, *Psychological Review*, vol.84(4), pp.327-352, 1977
14. Wilson, D.R. & Martinez, T.R.: Improved Heterogeneous Distance Functions, *Journal of Artificial Intelligence Research*, vol.6, pp.1-34, 1997