Imprecision and User Preferences in Multimedia Queries: A Generic Algebraic Approach*

Paolo Ciaccia[1], Danilo Montesi[2], Wilma Penzo[1], and Alberto Trombetta[3]

[1] DEIS - CSITE-CNR, Bologna, Italy, {pciaccia,wpenzo}@deis.unibo.it
[2] DSI, Dept. of Computer Science, Milano, Italy, montesi@dsi.unimi.it
[3] DI, Dept. of Computer Science, Torino, Italy, tromb@di.unito.it

Abstract. Specification and efficient processing of similarity queries on multimedia databases have recently attracted several research efforts, even if most of them have considered specific aspects, such as indexing, of this new exciting scenario. In this paper we try to remedy this by presenting an integrated algebraic framework which allows many relevant aspects of similarity query processing to be dealt with. As a starting point, we assume the more general case where "imprecision" is already present at the data level, typically because of the ambiguous nature of multimedia objects' content. We then define a *generic* similarity algebra, SAME^W, where semantics of operators is deliberately left unspecified in order to better adapt to specific scenarios. A basic feature of SAME^W is that it allows user preferences, expressed in the form of *weights*, to be specified so as to alter the default behavior of most operators. Finally, we discuss some issues related to "approximation" and to "user evaluation" of query results.

1 Introduction

The fact that "traditional" (Boolean) queries are not really appropriate for dealing with multimedia (MM) data has been early recognized, and many systems now exist that allow users to issue queries where some form of "imprecision" is allowed. For instance, in the QBIC system [10] one can look for images which are "similar" to a target one, according to color, texture, and shape of embedded objects. Since MM queries can lead to (very) high response times, many efforts have been spent in order to devise access methods able to efficiently deal with complex features [6]. This line of research has been somewhat complemented by activity aiming to provide users with a full-fledged query language able to express complex similarity queries [17]. Although the processing of such complex queries has been the subject of some recent works [1, 7, 8, 15], a full understanding of the implications (both at a formal and at a system level) of similarity query processing is still lacking. In particular, several important issues have only partially been addressed, such as models able to capture the "essential" aspects of

* This work has been partially supported by the InterData MURST Italian project.

MM objects needed by similarity queries, the impact of "user preferences" on query processing, the management of "approximate" queries (and the relationship of this concept to those of query equivalence and query containment), the complexity and expressiveness of similarity-based query languages, and so on. Furthermore, contributions to above issues typically consider ad-hoc scenarios and/or completely ignore the other coordinates of the problem, thus resulting in a set of difficult-to-integrate recipes.

In this work we address several of the above points in a unified algebraic framework. We have deliberately chosen to "start simple" from the modeling point of view, in order to better focus on those aspects which are peculiar to similarity query processing. Thus, we consider a (extended) relational framework which takes into account the two major sources of "imprecision" arising when querying MM databases [21]: 1) imprecision of *classification* of MM data, and 2) imprecision in the *matching of features* that characterize the content of MM objects. As to the first point we rely on basic concepts from *fuzzy set theory*, and allow representation of "vague classification" both at tuple and at attribute level (Sect. 2). This reflects the fact that in some cases imprecision characterizes an object as a whole, whereas in others it only affects some of its attributes. We then introduce a "similarity algebra", called SAME^W,[1] which extends relational algebra in a conservative way and incorporates the use of "weights" in most of its operators, in order to adapt to user preferences (Sect. 4). We show how complex similarity queries can be easily expressed in SAME^W (Sect. 5) and how equivalence rules can be exploited for the purpose of query rewriting and optimization (Sect. 6). In Sect. 7 we present some extensions to our framework that consider such relevant issues as "approximation of results" and user feedback. Finally, we briefly discuss related work and conclude.

2 The Data Model

For the sake of clarity, we first remind some standard notation and definitions. Given a set of attribute names, $\mathcal{A}$, and a function $dom()$ which associates to each $A \in \mathcal{A}$ a value domain, $dom(A)$, a (named) relation schema is formed by a relation name, R, and a subset of attributes $X = \{A_1, \ldots, A_n\} \subseteq \mathcal{A}$.[2] A tuple t over $R(X)$ is any element of $dom(X) = dom(A_1) \times \ldots \times dom(A_n)$, and $t.A_i \equiv t[A_i]$ denotes the value of A_i in t. For any $Y \subseteq X$, $t[Y]$ denotes the restriction of t on Y, that is, the (sub-)tuple obtained from t by considering only the values of the attributes in Y.

Our data model extends the relational one by allowing both *fuzzy attributes* and *fuzzy relations*. Remind that a fuzzy set F over a "universe" U is a set characterized by a *membership function* $\mu_F : U \to \mathcal{S}$, where $\mu_F(x)$ is the *degree of membership* of x in F, also called "score" or "grade". In the following we will always consider a normalized *score domain* $\mathcal{S} = [0, 1]$.

[1] SAME^W stands for "Similarity Algebra for Multimedia Extended with Weights".

[2] We will adopt the conventional list notation for sets, thus writing A for $\{A\}$ and XY for $X \cup Y$.

Imprecision at the attribute level is captured by the notion of "fuzzy domain". We say that A_i is a fuzzy attribute if its values are pairs of the form $F_j : s_j$, where F_j is a fuzzy set and $s_j \in \mathcal{S}$. The two components of A_i can be referred to as A_i^v (the "value") and A_i^μ (the "score"), respectively. Intuitively, given a tuple t, $t.A_i$ is interpreted as "$t.A_i^v$ is an appropriate value, with score $t.A_i^\mu$, of attribute A_i for t ", or, equivalently, that "t fits A_i^v with score A_i^μ". In practice, $t.A_i^v$ will be a real-world, application-specific concept, used to classify t according to A_i. For instance, an image can be classified, considering its `Brightness`, as `dark` with score 0.8.[3] For lack of space, in this paper we do not explicitly consider the case where $t.A_i$ is set-valued, which is appropriate when multiple non-exclusive classifications are possible (for instance, an image could be classified, according to its `Color`, as `red` with score 0.8 and as `orange` with score 0.7). However, as shown in [4], this is not restrictive, since set-valued attributes can be easily "normalized" into single-valued attributes.

Non-fuzzy (*crisp*) domains include "ordinary" sets of values, like `integer` and `string`, as well as more complex domains, like, say, `color_histogram`, which are required to represent feature values extracted from MM objects. We conveniently call *feature attribute* an attribute defined over such domains, in order to emphasize that for such attributes *similarity predicates* (rather than exact-matching) are the usual way to compare feature values (see Sect. 3.1).

Imprecision can also occur at the whole tuple level, and motivates the introduction of fuzzy relations. A fuzzy relation r over $R(X)$ is a fuzzy subset of $dom(X)$, $r \subseteq dom(X)$, characterized by a membership function μ_R (or simply μ if R is clear from the context)[4] which assigns to each tuple t a grade $\mu_R(t) \in \mathcal{S}$. The notation $t.\mu_R$ will be used with the same meaning of $\mu_R(t)$. We say that t belongs to r ($t \in r$) iff $t.\mu_R > 0$, and r is called a *crisp* instance iff $t.\mu_R = 1$ for each $t \in r$. The intuition about fuzzy relations is that, if a schema $R(X)$ represents some real-world "fuzzy" concept, the introduction of *tuple imprecision* permits to model how much a given object (tuple) "fits" the concept expressed by $R(X)$.

3 The SAME^W Algebra: Preliminaries

SAME^W extends relational algebra (RA) with a set of peculiarities that make it amenable to easily formulate complex similarity queries. SAME^W expressions operate on fuzzy relations and always return a fuzzy relation, and can use ordinary (Boolean), *similarity*, and *fuzzy* predicates. Semantics of expressions is *parametric* in the semantics of logical operators, which can therefore be varied in order to better adapt to user and application requirements. "Weights" can also be used to bias the behavior of most operators.

[3] Clearly, the fuzzy sets which can occur as values of A_i^v must be defined over the same universe.

[4] There is a slight notational inconsistency here, since the membership function should be denoted by μ_r. In practice, however, the best thing is to use, when needed, the relation name, since instances are usually unnamed.

3.1 Predicates and Formulas

A logical formula f is obtained by combining predicates with logical connectives, respecting the syntax $f ::= p|f \wedge f|f \vee f|\neg f|(f)$, where p is a predicate. The *evaluation* of f on a tuple t is a score $s(f,t) \in \mathcal{S}$ which says how much t satisfies f. We simply say that t *satisfies* f iff $s(f,t) > 0$. How $s(f,t)$ depends on (the evaluation on t of) the predicates in f is intentionally left unspecified, in order to achieve parametricity with respect to the semantics of logical operators. The basic constraint we impose is that $s(f,t)$ has to be computed by means of a so-called "scoring function" [8], s_f, whose arguments are the scores, $s(p_i,t)$, of t with respect to the predicates in f, that is:

$$s(f(p_1,\ldots,p_n),t) = s_f(s(p_1,t),\ldots,s(p_n,t)) \ . \tag{1}$$

Besides Boolean predicates, which evaluate to either 1 (true) or 0 (false), we also consider similarity and fuzzy predicates. A similarity predicate has either the form $A \sim v$, where A is a feature attribute, $v \in dom(A)$ is a constant (*query value*), and $\sim$ is a *similarity operator*, or $A_1 \sim A_2$, where both A_1 and A_2 are over the same domain.[5] Then, the evaluation of $p : A \sim v$ on t returns a score, $s(p,t) \in \mathcal{S}$, which says how much $t.A$ is similar to the query value v. For instance, evaluating the predicate `Color` $\sim$ `red` over an image t returns a score assessing the "redness" of t.[6]

Fuzzy predicates, on the other hand, operate on fuzzy attributes. Remind that if A is a fuzzy attribute, then $t.A$ is a pair $t.A^v : t.A^\mu$, where $t.A^v$ is (the name of) a fuzzy set and $t.A^\mu$ is the membership degree of t in $t.A^v$. The evaluation on t of a fuzzy predicate $q : A = w$, where w is a fuzzy set, is the score $s(q,t) = t.A^\mu$, if $t.A^v = w$, otherwise $s(q,t) = 0$. This is to say that we assume that different fuzzy sets are incomparable. The same applies to the predicate $q : A_1 = A_2$. In this case $s(q,t) = 0$ if $t.A_1^v \neq t.A_2^v$, otherwise the score is computed as a "parametric conjunction" of the two membership degrees, that is, $s(q,t) = s_\wedge(t.A_1^\mu, t.A_2^\mu)$, where $s_\wedge$ denotes the AND scoring function.

For the sake of definiteness, in the following we restrict our focus on the class $\mathcal{F}$ of scoring functions corresponding to fuzzy *t-norms* and *t-conorms* [13, 8], for which the AND ($\wedge$) and OR ($\vee$) operators satisfy the following properties:

1. They are both associative and commutative;
2. $s_\wedge(x,1) = x$ and $s_\vee(x,0) = x$ (*boundary* condition);
3. $x_1 \leq x_2 \Rightarrow s_\wedge(x,x_1) \leq s_\wedge(x,x_2)$ and $s_\vee(x,x_1) \leq s_\vee(x,x_2)$ (*monotonicity*).

As to the NOT ($\neg$) operator, it is assumed to satisfy the two properties:

1. $s_\neg(1) = 0$ and $s_\neg(0) = 1$ (*boundary* condition);
2. $x_1 \leq x_2 \Rightarrow s_\neg(x_1) \geq s_\neg(x_2)$ (*monotonicity*).

[5] This is a simplification. It suffices that the domains are compatible for '$\sim$' to be well-defined.

[6] Note that `red` is just a name for a value of $dom($`Color`$)$, and not a string constant.

For instance, $\mathcal{FS}$ (fuzzy standard) and $\mathcal{FA}$ (fuzzy algebraic) [13] semantics are given by the following set of rules:

	$\mathcal{FS}$	$\mathcal{FA}$
$s(f_1 \wedge f_2, t)$	$\min(s(f_1,t), s(f_2,t))$	$s(f_1,t) \cdot s(f_2,t)$
$s(f_1 \vee f_2, t)$	$\max(s(f_1,t), s(f_2,t))$	$s(f_1,t) + s(f_2,t) - s(f_1,t) \cdot s(f_2,t)$
$s(\neg f, t)$	$1 - s(f,t)$	$1 - s(f,t)$

Considering the general case of n-ary t-norms and t-conorms, the following inequalities hold $\forall x_1, \ldots, x_n \in \mathcal{S}$ [13]:

$$s_\wedge(x_1, \ldots, x_n) \leq \min(x_1, \ldots, x_n) \leq x_i \qquad \forall i \in [1..n] \tag{2}$$

$$s_\vee(x_1, \ldots, x_n) \geq \max(x_1, \ldots, x_n) \geq x_i \qquad \forall i \in [1..n] \ . \tag{3}$$

3.2 Dealing with User Preferences: Weights

With a non-Boolean semantics, it is quite natural and useful to give the user the possibility to assign a different relevance to the conditions he states to retrieve tuples. Such "user preferences" can be expressed by means of *weights*, thus saying, for instance, that the score of a predicate on `Color` is twice as important as the score of a predicate on the `Texture` of an image. The seminal work by Fagin and Wimmers [9] shows how *any* scoring function s_f for a formula $f(p_1, \ldots, p_n)$ can be properly extended into a weighted version, s_{f_Θ}, where $\Theta = [\theta_1, \ldots, \theta_n]$ is a vector of weights (also called a "weighting"), in such a way that:

1. s_{f_Θ} reduces to s_f when all the weights are equal;
2. s_{f_Θ} does not depend on $s(p_i, t)$ when $\theta_i = 0$;
3. s_{f_Θ} is a continuous function of the weights, for each fixed set of argument scores.

Let $x_i = s(p_i, t)$ denote the score of t with respect to p_i, and assume without loss of generality $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n$, with $\theta_i \in [0, 1]$ and $\sum_i \theta_i = 1$. Then, Fagin and Wimmers' formula is:

$$s_{f_\Theta}(x_1, \ldots, x_n) = (\theta_1 - \theta_2) \cdot x_1 + 2 \cdot (\theta_2 - \theta_3) \cdot s_f(x_1, x_2) + \cdots + n \cdot \theta_n \cdot s_f(x_1, \ldots, x_n) \ . \tag{4}$$

Although above formula is usually used to weigh the predicates appearing in a (selection) formula, our position is that *whenever scores have to be "combined", then a weighting should be allowed.* For instance, if we take the union of two relations, it might be reasonable to require that tuples in the first relation are "more important" than tuples in the second one. A meaningful example is when we want to integrate results from different search engines, but we trust more one than the other. Accordingly, most of the SAME^W operators[7] that compute new tuples' scores can use weights.

[7] We only leave out Difference, because we were not able to conceive any meaningful "weighted Difference" query. As to Projection, since the duplicate tuples to be combined together are not a priori known, it is not possible to assign weights to them.

4 The SAME^W Algebra

Basic operators of SAME^W conservatively extend those of RA in such a way that, if no "imprecision" is involved in the evaluation of an expression, the semantics of RA applies (see Theorem 4.1). Genericity with respect to different semantics is achieved by defining SAME^W operators in terms of the (generic) scoring functions of the logical operators. Thus, if a given semantics is adopted for formulas, the same is used by SAME^W operators, which avoids counter-intuitive phenomena and preserves many RA equivalence rules. As an example, the semantics of Union ($\cup$) is based on that of the OR ($\vee$) operator.

In the following, $E(X)$ denotes an expression with schema X, and $e = E[db]$ is the fuzzy set of tuples with schema X obtained by evaluating $E(X)$ over the current database db. We say that a tuple t belongs to e ($t \in e$) iff $t.\mu_E > 0$ holds. Two tuples t_1 and t_2 with attributes X are *equal* iff $t_1[A_i] = t_2[A_i]$ holds for each $A_i \in X$. In case of fuzzy attributes, tuple equality thus requires that also the attributes' grades are the same. Two relations e_1 and e_2 are equal iff: 1) they consist of the same set of tuples, and 2) $\forall t_1 \in e_1, \forall t_2 \in e_2 : t_1 = t_2 \Rightarrow t_1.\mu = t_2.\mu$.

We start by extending "traditional" operators of RA, and then introduce new operators which have no direct counterpart in RA.

Selection (σ) The Selection operator applies a formula f to the tuples in e and filters out those which do not satisfy f. The novel point here is that, as an effect of f and of weights, *the grade of a tuple t can change*. Weights can be used for two complementary needs: In the first case, they weigh the importance of predicates in f, as in [9], thus leading to use the scoring function $s_{f_{\Theta_f}}$ in place of s_f.[8] In the second case they are used to perform a *weighted conjunction*, $s^{\Theta}_{\wedge}$, between the score computed by f and the "input" tuple score, $t.\mu_E$. This determines the new tuple score, $t.\mu$:

$$\sigma^{\Theta}_{f_{\Theta_f}}(e) = \{t \mid t \in e \wedge t.\mu = s^{\Theta}_{\wedge}(s(f_{\Theta_f}, t), t.\mu_E) > 0\} \ . \tag{5}$$

Projection (π) As in RA, the Projection operator removes a set of attributes and then eliminates duplicate tuples. Projection can also be used to *discard* scores, both of fuzzy attributes and of the whole tuple. In this case, however, in order to guarantee consistency of subsequent operations, such scores are simply set to 1, so that they can still be referenced in the resulting schema. This captures the intuition that if we discard, say, the tuples' scores, then the result is a crisp relation, that is, a fuzzy relation whose tuples all have score 1.

Formally, let e be a relation with schema $E(X)$, $Y \subseteq X$, and V a set of *v-annotated* fuzzy attributes, $V = \{A_i^v\}$, where V contains exactly those fuzzy attributes for which scores are to be discarded. Note that V can include A_i^v only if $A_i \in X - Y$. Finally, let F stand for either μ or the empty set. Then, the projection of e over YVF is a relation with schema YW, where if $A_i^v \in V$ then

[8] When using weights, f is restricted to be either a conjunction or a disjunction of predicates.

$A_i \in W$, defined as follows:

$$\pi_{YVF}(e) = \{t[YW] | \exists t' \in e : t[YV] = t'[YV] \wedge \forall A_i^v \in V : t.A_i^\mu = 1 \wedge t.\mu = s_\vee\{t''.\mu_E | t''[YV] = t[YV]\} \text{ if } F = \mu, \text{ otherwise } t.\mu = 1\} \ . \quad (6)$$

Thus, tuples' scores are discarded (i.e. set to 1) when $F = \emptyset$, whereas they are preserved when $F = \mu$. In the latter case, new scores are computed by considering the "parametric disjunction", $s_\vee$, of the scores of all duplicate tuples with the same values for YV.

Union ($\cup$) In SAME^W the Union is an n-ary operator,[9] which, given n relations e_i with schemas $E_i(X)$, computes the score of a tuple t as a *weighted disjunction*, $s_\vee^\Theta(t.\mu_{E_1}, \ldots, t.\mu_{E_n})$, with $\Theta = [\theta_1, \ldots, \theta_n]$, of the input tuples' scores, that is:

$$\cup^\Theta(e_1, \ldots, e_n) = \{t \mid (t \in e_1 \vee \ldots \vee t \in e_n) \wedge t.\mu = s_\vee^\Theta(t.\mu_{E_1}, \ldots, t.\mu_{E_n}) > 0\} \ . \quad (7)$$

Note that, because of the presence of weights, Union is not associative anymore. This implies that the n-ary Union cannot be defined in terms of $n-1$ binary unions, as it happens in RA.

Join ($\bowtie$) Also the weighted (natural) Join is an n-ary operator, where the score of a result tuple t is a *weighted conjunction*, $s_\wedge^\Theta(t_1.\mu_{E_1}, \ldots, t_n.\mu_{E_n})$, of the scores of matching tuples:

$$\bowtie^\Theta (e_1, \ldots, e_n) = \{t[X_1 \ldots X_n] \mid \exists t_1 \in e_1, \ldots, \exists t_n \in e_n : t[X_1] = t_1[X_1] \wedge \ldots \wedge t[X_n] = t_n[X_n] \wedge t.\mu = s_\wedge^\Theta(t_1.\mu_{E_1}, \ldots, t_n.\mu_{E_n}) > 0\} \ . \quad (8)$$

Difference ($-$) Given relations e_1 and e_2 with schemas $E_1(X)$ and $E_2(X)$, respectively, their Difference is defined as:

$$e_1 - e_2 = \{t \mid t \in e_1 \wedge t.\mu = s_\wedge(t.\mu_{E_1}, s_\neg(t.\mu_{E_2})) > 0\} \ . \quad (9)$$

Renaming (ρ) The Renaming operator is as in RA, thus we do not repeat its definition here.

The following result proves that the "RA-fragment" of SAME^W, that is, SAME^W restricted to the above operators, is indeed a conservative extension of relational algebra.

Theorem 4.1. *Let E be a SAME^W expression that does not use weights and that includes only operators in $\{\sigma, \pi, \cup, \bowtie, -, \rho\}$. If E does not use similarity operators, the database instance db is crisp and the semantics of the scoring functions is in $\mathcal{F}$, then $E[db] = E_{RA}[db]$, the latter being evaluated with the Boolean semantics of RA.*

Proof. (sketch) It is sufficient to show that every operator in $\{\sigma, \pi, \cup, \bowtie, -, \rho\}$, when applied to crisp relations, yields a crisp relation. Here we only consider

[9] We also use the infix notation, $E_1 \cup^{[\theta_1, \theta_2]} E_2$, when only two operands are present.

the case of Selection, arguments for the other operators being similar. A Selection $\sigma_f(E)$, where f does not contain similarity predicates and $E[db]$ is a crisp relation, computes the score of a tuple $t \in E[db]$ as $s_\wedge(s(f,t), t.\mu_E)$. Since $s_\wedge$ is a t-norm, $t.\mu_E$ is equal to 1, and $s(f,t)$ is either 0 or 1, the Selection outputs a tuple score that equals either 0 or 1, thus yielding a crisp relation. □

Two other operators of SAME^W, that can be derived from those already introduced, are:

Boolean difference ($\overset{B}{-}$) The Boolean difference behaves "as expected", in that if a tuple t belongs to both e_1 and e_2, with schemas $E_1(X)$ and $E_2(X)$, respectively, then it is not part of the result, regardless of its scores (in general, this is not the case for the Difference), that is:

$$e_1 \overset{B}{-} e_2 = \{t \,|\, t \in e_1 \wedge t \notin e_2 \wedge t.\mu = t.\mu_{E_1} > 0\} \ . \tag{10}$$

Boolean difference can be defined in terms of other operators as $e_1 \overset{B}{-} e_2 = e_1 - \pi_X(e_2)$, where $\pi_X(e_2)$, according to the semantics of Projection, simply discards the scores of the tuples in e_2.

Intersection ($\cap$) As in RA, the Intersection can be defined as a particular case of Join, where all the n operands have the same schema X:

$$\begin{aligned} \cap^{\Theta}(e_1, \ldots, e_n) &= \{t \,|\, (t \in e_1 \wedge \ldots \wedge t \in e_n) \wedge t.\mu = s_\wedge^{\Theta}(t.\mu_{E_1}, \ldots, t.\mu_{E_n}) > 0\} \\ &= \bowtie^{\Theta} (e_1, \ldots, e_n) \ . \end{aligned} \tag{11}$$

The two new operators introduced in SAME^W are the *Top* and the *Cut*.

Top (τ) The Top operator retrieves the first k (k is an input parameter) tuples of a relation e, according to a *ranking criterion*, as expressed by a ranking function g. If weights are used to rank tuples according to g_{Θ_g}, then g has to be a formula of predicates over the schema of e.[10] If e has no more than k tuples, then $\tau^k_{g_{\Theta_g}}(e) = e$, otherwise:

$$\begin{aligned} \tau^k_{g_{\Theta_g}}(e) = \{t \,|\, t \in e \wedge |\tau^k_{g_{\Theta_g}}(e)| = k \wedge \forall t \in \tau^k_{g_{\Theta_g}}(e) : \\ \nexists\, t' : t' \in e \wedge t' \notin \tau^k_{g_{\Theta_g}}(e) \wedge g_{\Theta_g}(t') > g_{\Theta_g}(t)\} \end{aligned} \tag{12}$$

with ties arbitrarily broken. When g is omitted, the *default* ranking criterion, based on the score of tuples, applies, thus the k tuples with the highest scores are returned.

Cut (γ) The Cut operator "cuts off" those tuples which do not satisfy a formula g, that is:

$$\gamma_g(e) = \{t \,|\, t \in e \wedge s(g,t) > 0 \wedge t.\mu = t.\mu_E > 0\} \ . \tag{13}$$

Unlike Selection, Cut *does not change tuples' scores*. Thus, if g includes non-Boolean predicates, the two operators would behave differently. However, the

[10] If "bottom" tuples are needed, the *ranking directive* $<$ can be used, written $g_{\Theta_g,<}$.

major reason to introduce Cut is the need of expressing *(threshold) conditions on tuples' scores*, e.g. $\mu > 0.6$. Such a predicate cannot be part of a Selection, since it does not commute with others. This is also to say that the expressions $\gamma_{\mu>0.6}(\sigma_f(E))$ and $\sigma_f(\gamma_{\mu>0.6}(E))$ are *not* equivalent. Indeed, the first expression is *contained* in the second one, that is:

$$\gamma_{\mu>\alpha}(\sigma_f(E)) \sqsubseteq \sigma_f(\gamma_{\mu>\alpha}(E)) \quad . \tag{14}$$

Proof. ($\sqsubseteq$) Let E_L and E_R stand for the left and right hand side expression, respectively. Since the Cut does not modify tuples' scores, if a tuple t satisfies both E_L and E_R, then its score will be the same, i.e. $t.\mu_{E_L} = t.\mu_{E_R}$. If $t \in E_L[db]$, then $t.\mu_{E_L} = s_\wedge(s(f,t), t.\mu_E) > \alpha$ holds. From the monotonicity and boundary condition of t-norms it follows that $t.\mu_E > \alpha$ holds too, thus $t \in \gamma_{\mu>\alpha}(E)[db]$. This is enough to prove that $t \in E_R[db]$.

($\not\sqsupseteq$) Consider a tuple t for which both $t.\mu_E > \alpha$ and $0 < s_\wedge(s(f,t), t.\mu_E) \leq \alpha$ hold. This implies that $t \in E_R[db]$ but $t \notin E_L[db]$. □

5 Examples

Examples in this section, aiming to show the potentialities and flexibility of SAME^W, refer to a *biometric DB* using faces and fingerprints to recognize the identity of a person. Stored data include extracted features relevant for identification,[11] and modeled by the `FaceFV` and `FP_FV` attributes, respectively. Because of the huge size of biometric databases, a viable way to improve performance is, at face and fingerprint acquisition time, to *classify* them with respect to some predefined classes. As to fingerprints, as demonstrated in [14], a "continuous" classification approach, where a fingerprint is assigned with some degree to many (even all) classes, can perform better than an approach based on "exclusive" classification. As to faces, we consider that the `Chin` and the `Hair` are also preventively classified.

Our simplified biometric DB consists of the following relations, where the '$*$' denotes fuzzy attributes and relations that can have fuzzy instances, and primary keys are underlined. The `Freq` attribute is the relative frequency of a fingerprint class.[12] This can be computed by considering the *scalar cardinality* (also called the *sigma count* [13]) of the fuzzy set corresponding to the fingerprint class. A partial instance is shown in Fig. 1.

`Persons(`<u>`PId`</u>`,Name)`
`Faces(`<u>`PId`</u>`,FaceFV,Chin*,Hair*)`
`FingerPrints(`<u>`FPId`</u>`,PId,FingerNo,FP_FV)`
`FPClasses(`<u>`Class`</u>`,Freq)`
`FPType*(`<u>`FPId,Class`</u>`)`

[11] For fingerprints these can be "directional vectors", positions of "minutiae", etc. For faces, position of eyes, nose, etc., can be considered.

[12] Class names are among those adopted by NIST (U.S. National Institute of Standards and Technology).

Persons

PId	Name
P00001	John
P00002	Mary
P00003	Bill

FPClasses

Class	Freq
Arch	3.7%
LeftLoop	33.8%
RightLoop	31.7%

Faces

PId	FaceFV	Chin	Hair
P00001	FFV0001	pointed:0.74	black:0.87
P00002	FFV0002	rounded:0.65	brown:0.75
P00003	FFV0003	pointed:0.93	brown:0.84

FingerPrints

FPId	PId	FingerNo	FP_FV
FP0001	P00001	1	FPFV0001
FP0002	P00001	2	FPFV0002
FP0011	P00002	1	FPFV0011
FP0015	P00002	5	FPFV0015
FP0017	P00002	7	FPFV0017

FPType

FPId	Class	μ
FP0001	Arch	0.65
FP0001	RightLoop	0.25
FP0003	LeftLoop	0.95
FP0005	Arch	0.60
FP0005	LeftLoop	0.20

Fig. 1. An instance of the biometric database

The first query aims to retrieve those persons who have black hair, and whose facial features match the ones (`inFace`) given in input:

$$\sigma_{(Hair='black')\wedge(FaceFV\sim\texttt{inFace})}(Faces) \ .$$

The final score of a tuple t is obtained by combining the scores of both Selection predicates and the initial score of the tuple (this is 1, since `Faces` is a crisp relation). For instance, if the $\mathcal{FA}$ semantics is used, it is $t.\mu = (s(p_1,t)\cdot s(p_2,t))\cdot 1$. Assuming the following similarity table:

$\sim$ inFace

FaceFV	score
FFV0001	0.60
FFV0002	0.84
FFV0003	0.33

the result of the query is therefore $(0.87 \cdot 0.60 = 0.522)$:

PId	FaceFV	Chin	Hair	μ
P00001	FFV0001	pointed:0.74	black:0.87	0.522

Trusting more hair classification than feature matching is achieved by giving the first predicate a weight > 0.5, say 0.7:

$$\sigma_{(Hair='black')^{0.7}\wedge(FaceFV\sim\texttt{inFace})^{0.3}}(Faces) \ .$$

The score of the resulting tuple is now computed as (the score of the crisp relation is omitted since it equals 1 and does not influence the computation):

$$(0.7-0.3)\cdot s(p_1,t)+2\cdot 0.3\cdot s_\wedge(s(p_1,t),s(p_2,t)) = 0.4\cdot 0.87+0.6\cdot(0.87\cdot 0.60) = 0.6612 \ .$$

On the other hand, if the $\mathcal{FS}$ semantics is used, the final score would be:

$$0.4 \cdot 0.87 + 0.6 \cdot \min(0.87, 0.60) = 0.708 \ .$$

If we want only the persons' id's and the scores of the 10 best matching tuples, we can use the Top and Projection operators:

$$\pi_{PId,\mu}(\tau^{10}(\sigma_{(Hair='black')^{0.7}\wedge(FaceFV\sim\texttt{inFace})^{0.3}}(Faces))) \ .$$

In order to see, for the only persons returned by above expression, call it E, how well the fingerprint of the left thumb (`FingerNo` $= 1$) matches a given fingerprint (`inFP`), we can write:

$$\sigma_{(FingerNo=1)\wedge(FP\_FV\sim\texttt{inFP})}(FingerPrints) \bowtie \pi_{PId}(E)$$

or, equivalently:

$$\sigma_{(FingerNo=1)\wedge(FP\_FV\sim\texttt{inFP})}(FingerPrints) \bowtie^{[1,0]} E$$

since both expressions discard the scores computed by E. Indeed, the Projection of E on PId returns a crisp relation, thus the final scores of the resulting tuples are, because of the boundary condition of t-norms, those of the component tuples returned by the Selection expression, call it E'. On the other hand, the weighted Join in the second expression returns a set of tuples whose scores are given by the following rule, where E' still denotes the left operand of the Join:

$$(\theta_1-\theta_2)\cdot t.\mu_{E'}+2\cdot\theta_2\cdot s_\wedge(t.\mu_{E'},t.\mu_E) = (1-0)\cdot t.\mu_{E'}+2\cdot 0\cdot s_\wedge(t.\mu_{E'},t.\mu_E) = t.\mu_{E'} \ .$$

For a more complex query, assume we want to perform a combined match on fingerprints and faces, giving as input an `inFP` and an `inFace`. We then join the result, weighting more (0.6) the match on faces, combine the tuples of a same person (since each person has more than one fingerprint in the DB), and discard all the tuples whose overall score is less than 0.5:

$$\gamma_{\mu\geq 0.5}(\pi_{PId,\mu}(\sigma_{FaceFV\sim\texttt{inFace}}(Faces) \bowtie^{[0.6,0.4]} \sigma_{FP\_FV\sim\texttt{inFP}}(FingerPrints))) \ .$$

Given the following similarity table:

$\sim$ `inFP`

FP_FV	score
FPFV0001	0.72
FPFV0002	0.48
FPFV0011	0.84
FPFV0015	0.38
FPFV0017	0.55

consider the case of person P00001 ('John'), thus the join of the first tuple of `Faces` with the first two tuples of `FingerPrints`. The scores computed by the weighted Join (when the $\mathcal{FS}$ semantics is used) are, respectively:

$$(0.6-0.4)\cdot 0.60+2\cdot 0.4\cdot\min(0.60,0.72) = 0.2\cdot 0.60+0.8\cdot 0.60 = 0.60$$
$$(0.6-0.4)\cdot 0.60+2\cdot 0.4\cdot\min(0.60,0.48) = 0.2\cdot 0.60+0.8\cdot 0.48 = 0.504 \ .$$

Then, the Projection combines the two tuples and returns (P00001,0.60), since $\max(0.60, 0.504) = 0.60$. Since $0.60 \geq 0.5$, the tuple is preserved by the Cut operator.

As a final example query, we want to know who are those persons with an 'Arch' fingerprint and with a pointed chin, giving these conditions weights 0.6 and 0.4, respectively. This can be expressed by means of a weighted Join, where a 0 weight is used for the `Fingerprints` relation on which no predicates are specified:

$$\bowtie^{[0.6,0.4,0]} (\sigma_{Class='Arch'}(FPType), \sigma_{Chin='pointed'}(Faces), FingerPrints) \ .$$

6 Reasoning in SAME^W

Equivalence and containment rules in SAME^W can only partially rely on results from RA since, unless we consider the $\mathcal{FS}$ semantics, rules based on *idempotence* and/or *distributivity* of logical operators are no longer valid (e.g., $E \cup E \not\equiv E$ in $\mathcal{FA}$). Nonetheless, many opportunities for query rewriting are still left. For lack of space, here we present only a selected sample of such rules, focusing on Cut and Top operators and on the effect of weights. Complete proofs are given only for some of the results. In order to simplify the notation, when no direct manipulation of weights is involved, we understand the presence of weights, thus writing, say, f in place of f_{Θ_f}. In general, E_L and E_R will be used to denote the left hand side and the right hand side expression, respectively. As to proofs' style, in order to demonstrate that $E_L \equiv E_R$ ($E_L \sqsubseteq E_R$) holds, we usually split the proof in two parts:

1. We first show that, if $t \in E_L[db]$ and $t \in E_R[db]$ both hold for a generic tuple t, then $t.\mu_{E_L} = t.\mu_{E_R}$.
2. Then we prove that E_L is "Boolean equivalent" to ("Boolean contained" in, respectively) E_R, written $E_L \equiv_b E_R$ ($E_L \sqsubseteq_b E_R$, resp.), that is that the sets of tuples computed by the two expressions are the same (the first is a subset of the second, resp.), *regardless of tuples' scores.*

We start with some basic rules that are also useful for proving more complex results. The following containment relationships hold for any expression E:

$$\gamma_f(E) \sqsubseteq E \tag{15}$$

$$\tau_g^k(E) \sqsubseteq E \tag{16}$$

whereas $\sigma_f(E) \not\sqsubseteq E$, since σ modifies the tuples' grades.

As to monotonicity of unary operators, assume that $E' \sqsubseteq E$ holds. Then:

$$\gamma_f(E') \sqsubseteq \gamma_f(E) \tag{17}$$

$$\sigma_f(E') \sqsubseteq \sigma_f(E) \ . \tag{18}$$

On the other hand, the Top operator is not monotone, that is, $E' \sqsubseteq E$ does not imply $\tau_g^k(E') \sqsubseteq \tau_g^k(E)$, as it can be easily proved.

The weighted Join is monotone. For this, assume that $E'_i \sqsubseteq E_i$ holds $\forall i \in [1..n]$. Then:

$$\bowtie^{\Theta} (E'_1, \ldots, E'_n) \sqsubseteq \bowtie^{\Theta} (E_1, \ldots, E_n) \ . \tag{19}$$

Indeed, if a tuple t belongs to $E_L[db]$, then it also belongs to $E_R[db]$. Since the joining sub-tuples leading to t are necessarily the same in both cases, then also the score of t will be the same.

Moving Cut's Around. Consider the "canonical" Cut condition, $\gamma_{\mu>\alpha}$, abbreviated γ_α, applied to a weighted Join. Our first equivalence rule shows that a Cut γ_{α_i} can be applied to the i-th Join operand, where α_i depends on the values of weights. Assuming $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n$, we have:

$$\gamma_\alpha(\bowtie^{[\theta_1,\ldots,\theta_n]} (E_1, \ldots, E_n)) \equiv \gamma_\alpha(\bowtie^{[\theta_1,\ldots,\theta_n]} (\gamma_{\alpha_1}(E_1), \ldots, \gamma_{\alpha_n}(E_n))) \tag{20}$$

where:

$$\alpha_i = \frac{\alpha - \sum_{j=1}^{i-1} j \cdot (\theta_j - \theta_{j+1})}{1 - \sum_{j=1}^{i-1} j \cdot (\theta_j - \theta_{j+1})} \qquad i \in [1..n] \ . \tag{21}$$

For instance, when $n = 3$, $[\theta_1, \theta_2, \theta_3] = [0.5, 0.3, 0.2]$, and $\alpha = 0.6$, it is $\alpha_1 = 0.6$, $\alpha_2 = 0.5$, and $\alpha_3 = 1/3$. Note that $\alpha_1 = \alpha$ and that $\alpha_i = \alpha$ when all the weights are equal, i.e. $\gamma_\alpha(\bowtie (E_1, E_2, \ldots, E_n)) \equiv \gamma_\alpha(\bowtie (\gamma_\alpha(E_1), \gamma_\alpha(E_2), \ldots, \gamma_\alpha(E_n)))$.

Proof. First observe that, since γ does not change the tuples' grades, $t.\mu_{E_L} = t.\mu_{E_R}$ holds for all tuples t which belong to both $E_L[db]$ and $E_R[db]$.

($\sqsubseteq$) Let $t_i \in E_i[db]$, and let $x_i = t_i.\mu_{E_i}$. A tuple $t = \bowtie^{[\theta_1,\theta_2,\ldots,\theta_n]} (t_1, t_2, \ldots, t_n)$ belongs to $E_L[db]$ iff $t.\mu_{E_L} > \alpha$, where $t.\mu_{E_L}$ is computed according to (4), with the t-norm $s_\wedge$ which takes the place of s_f. In order to show that the α_i cut-off value computed by (21) is safe, first observe that, due to (2), $x_i \geq s_\wedge(x_1, \ldots, x_j)$ holds $\forall j \in [i..n]$, whereas, when $j < i$, we can exploit the inequality $1 \geq s_\wedge(x_1, \ldots, x_j)$. Then, we can majorize $t.\mu_{E_L}$ as follows:

$$(\theta_1 - \theta_2) \cdot 1 + \cdots + (i-1) \cdot (\theta_{i-1} - \theta_i) \cdot 1 + i \cdot (\theta_i - \theta_{i+1}) \cdot x_i + \cdots + n \cdot \theta_n \cdot x_i \geq t.\mu_{E_L} > \alpha \ .$$

Considering that $\sum_{i=1}^{n} \theta_i = 1$, above inequality can be rewritten as:

$$x_i > \frac{\alpha - \sum_{j=1}^{i-1} j \cdot (\theta_j - \theta_{j+1})}{\sum_{j=i}^{n-1} j \cdot (\theta_j - \theta_{j+1}) + n \cdot \theta_n} = \frac{\alpha - \sum_{j=1}^{i-1} j \cdot (\theta_j - \theta_{j+1})}{1 - \sum_{j=1}^{i-1} j \cdot (\theta_j - \theta_{j+1})} \stackrel{\text{def}}{=} \alpha_i \ .$$

It follows that if $t \in E_L[db]$, then the corresponding t_i has a score $t_i.\mu_{E_i} > \alpha_i$, thus passing the γ_{α_i} filter.

($\sqsupseteq$) Trivial, since $\gamma_{\alpha_i}(E_i) \sqsubseteq E_i$, $\forall i \in [1..n]$. □

A similar rule applies when the Cut has the form $\gamma_{\mu<\alpha}$ and follows a weigthed Union:

$$\gamma_{\mu<\alpha}(\cup^{[\theta_1,,\ldots,\theta_n]}(E_1,\ldots,E_n)) \equiv \gamma_{\mu<\alpha}(\cup^{[\theta_1,\ldots,\theta_n]}(\gamma_{\mu<\alpha_1}(E_1),\ldots,\gamma_{\mu<\alpha_n}(E_n))) \tag{22}$$

where $\alpha_i = \alpha/(1-\sum_{j=1}^{i-1} j\cdot(\theta_j-\theta_{j+1}))$, and $\alpha_i=\alpha$ if weights are not used.

Proof. (sketch) The proof almost follows the same steps used to demonstrate (20). The only difference is that, if a tuple t belongs to both $E_L[db]$ and $E_R[db]$, then, in order to have that $t.\mu_{E_L} = t.\mu_{E_R}$, we have to prove that no occurrence of t in the $E_i[db]$'s is filtered out by the inner Cut's appering in E_R. □

A simple yet useful rule to manipulate Cut expressions is:

$$\gamma_{\alpha_1}(\sigma_f(\gamma_{\alpha_2}(\sigma_g(E)))) \sqsubseteq \gamma_{\alpha_1}(\sigma_{f\wedge g}(E)) \tag{23}$$

$$\gamma_{\alpha_1}(\sigma_f(\gamma_{\alpha_2}(\sigma_g(E)))) \equiv \gamma_{\alpha_1}(\sigma_{f\wedge g}(E)) \qquad \text{if } \alpha_1 \geq \alpha_2\ . \tag{24}$$

Proof. (sketch) The validity of (23) directly follows from monotonicity of Cut and Selection ((17) and (18), respectively), after observing that E_R can be rewritten as $\gamma_{\alpha_1}(\sigma_f(\sigma_g(E)))$ and that $\gamma_{\alpha_2}(\sigma_g(E)) \sqsubseteq \sigma_g(E)$ holds due to (15). To prove (24) we exploit the associativity of $s_\wedge$ and Cut's definition to show that if $t \in E_R[db]$ then t also necessarily belongs to $\gamma_{\alpha_2}(\sigma_g(E))[db]$, from which the result easily follows. □

Let us now consider the case where we apply to a *crisp* relation $E[db]$ a "cheap" predicate, p_1, and a "costly" one, p_2, after which we Cut the result. If predicates are weighted, and $\theta_1 \geq \theta_2$, we can apply a Cut just after evaluating p_1, which can lead to considerable cost saving. This also shows how weights on predicates can be transformed into weights on tuples' scores:

$$\gamma_\alpha(\sigma_{p_1^{\theta_1}\wedge p_2^{\theta_2}}(E)) \equiv \gamma_\alpha(\sigma^{[\theta_2,\theta_1]}_{p_2}(\gamma_\alpha(\sigma_{p_1}(E))))\ . \tag{25}$$

Proof. (sketch) Since the Cut does not modify tuples' scores, if t is in the result of both expressions then $t.\mu_{E_L} = t.\mu_{E_R} = s_\wedge^{[\theta_1,\theta_2]}(s(p_1,t),s(p_2,t)) > \alpha$.

($\sqsubseteq$) It is sufficient to show that if $t \in E_L[db]$, then its score is high enough to pass the inner Cut in E_R. This is proved by showing that above inequality implies $s(p_1,t) > \alpha$.

($\sqsupseteq$) It can be shown that E_L can be rewritten as $\gamma_\alpha(\sigma^{[\theta_2,\theta_1]}_{p_2}(\sigma_{p_1}(E)))$, after which containment follows from (17), (18), and (15). □

Note that above equivalence *does not* hold if p_1 is commuted with p_2, when $\theta_1 > \theta_2$ holds. In this case, indeed, a tuple t can satisfy $\sigma_{p_1^{\theta_1}\wedge p_2^{\theta_2}}(E)$ but not $\sigma^{[\theta_1,\theta_2]}_{p_1}(\sigma_{p_2}(E))$. In particular, this is the case when $s(p_2,t)=0$. The reason of this asymmetry directly stems from the asymmetry of Expression (4).

Moving Top's around. Turning to the Top operator, consider the case where the ranking criterion, g_{Θ_g} (or simply g) *does not* refer to tuples' scores. If no ties, according to g, occur in $E_i[db]$ ($i \in [1..n]$), then it is safe to apply a Top to each operand of a weighted Union, that is:

$$\tau_g^k(\cup^{[\theta_1,\ldots,\theta_n]}(E_1,\ldots,E_n)) \equiv \tau_g^k(\cup^{[\theta_1,\ldots,\theta_n]}(\tau_g^k(E_1),\ldots,\tau_g^k(E_n))) \ . \tag{26}$$

Proof. First observe that if E' and E are two expressions such that $E' \sqsubseteq_b E$ (note that we do not require equality of scores), and t belongs to $E'[db]$ (thus to $E[db]$) and to $\tau_g^k(E)[db]$, where g is a ranking criterion which does not consider tuples' scores, then t also belongs to $\tau_g^k(E')[db]$. Since Top does not change tuples' scores, the score of t will be the same in the two top-relations iff so it is in $E'[db]$ and $E[db]$. Now, let $E = \cup^{[\theta_1,\ldots,\theta_n]}(E_1,\ldots,E_n)$ and let $E' = \cup^{[\theta_1,\ldots,\theta_n]}(\tau_g^k(E_1),\ldots,\tau_g^k(E_n))$.

($\sqsubseteq$) Consider a tuple $t \in E_L[db] = \tau_g^k(E)[db]$. Thus, t is among the k best tuples in $E[db]$ according to the ranking established by g. We have to prove that if $t \in E_i[db]$ then t also belongs to $\tau_g^k(E_i)[db]$. Furthermore, we can limit to consider those sub-expressions for which $\theta_i > 0$ holds, the others being uninfluential at all. Since $E_i \sqsubseteq_b E$ (provided $\theta_i > 0$, as we are considering), from the above observation we have that $t \in \tau_g^k(E_i)[db]$. Therefore, whenever $t \in E_i[db]$ holds, then $t \in \tau_g^k(E_i)[db]$ holds too. This proves that $t.\mu_E = t.\mu_{E'}$. Since Top does not change grades, and $E' \sqsubseteq_b E$, we have that $t \in E_R[db] = \tau_g^k(E')[db]$ and that $t.\mu_{E_L} = t.\mu_{E_R}$.

($\sqsupseteq$) Straightforward. □

On the other hand, if the ranking criterion is based on tuples' scores, the above rule can be applied only if Union is unweighted and the $\mathcal{FS}$ (max) semantics is used:

$$\tau^k(\cup(E_1,\ldots,E_n)) \equiv \tau^k(\cup(\tau^k(E_1),\ldots,\tau^k(E_n))) \ . \tag{27}$$

Proof. ($\sqsubseteq$) Assume that $t \in \tau^k(\cup(E_1,\ldots,E_n))[db]$. Since $\mathcal{FS}$ semantics applies, this means that $t.\mu_{E_L} = \max(t.\mu_{E_1},\ldots,t.\mu_{E_n})$ is among the k highest scores considering all the tuples in $\cup(E_1,\ldots,E_n)[db]$. Without loss of generality, assume that $t.\mu_{E_L} = t.\mu_{E_1}$, i.e. tuple t achieves its best score in $E_1[db]$. We can prove that $t \in \tau^k(E_1)[db]$ as follows. Reasoning by absurd, assume that $t \notin \tau^k(E_1)[db]$, thus in $E_1[db]$ there are at least other k tuples t^h ($h \in [1..k]$) such that $t^h.\mu_{E_1} > t.\mu_{E_1}$ holds. From this we can conclude that $t^h.\mu_{E_L} \geq t^h.\mu_{E_1} > t.\mu_{E_1} = t.\mu_{E_L}$, thus t would not belong to $E_L[db]$, which is a contradiction.

($\sqsupseteq$) Straightforward. □

As a negative result, we state that in no case operands of a (weighted) Join followed by a Top can be reduced by pushing down the Top (as done for Union). Let $g = p_1 \wedge p_2 \wedge \ldots \wedge p_n$, with p_i over the schema of E_i. If the $\mathcal{FS}$ semantics is used and no score ties occur with respect to the p_i's and to g, it is:

$$\tau_g^k(\bowtie^{[\theta_1,\ldots,\theta_n]}(\tau_{p_1}^k(E_1),\ldots,\tau_{p_n}^k(E_n))) \sqsubseteq \tau_g^k(\bowtie^{[\theta_1,\ldots,\theta_n]}(E_1,\ldots,E_n)) \ . \tag{28}$$

Proof. Omitted. □

Considering the relationship between Top and Cut operators, in general we have the following containment result:

$$\gamma_f(\tau_g^k(E)) \sqsubseteq \tau_g^k(\gamma_f(E)) \ . \tag{29}$$

Proof. Assume that $t \in E_L[db]$. Then t also belongs to $\tau_g^k(E)$ (due to (15)) and to $\gamma_f(E)$ (due to (16) and (17)). Since $\gamma_f(E) \sqsubseteq E$ also holds due to (15), we can conclude that $t \in \tau_g^k(\gamma_f(E))[db]$. □

On the other hand, equivalence is obtained when $f = g$ or when both operators have their "canonical" form, that is $\tau^k(\gamma_\alpha(E)) \equiv \gamma_\alpha(\tau^k(E))$.

Moving Selection. Since the Selection operator changes tuples' scores, particular care has to be paid when reasoning about it. For instance, the following equivalence holds only for the $\mathcal{FS}$ semantics (and unweighted Union):

$$\sigma_f(\cup(E_1, \ldots, E_n)) \equiv \cup(\sigma_f(E_1), \ldots, \sigma_f(E_n)) \ . \tag{30}$$

Proof. We first show that if t satisfies both E_L and E_R, then $t.\mu_{E_L} = t.\mu_{E_R}$. As to E_L, it is:

$$t.\mu_{E_L} = s_\wedge(s(f,t), s_\vee(t.\mu_{E_1}, \ldots, t.\mu_{E_n})) \tag{31}$$

whereas

$$t.\mu_{E_R} = s_\vee(s_\wedge(s(f,t), t.\mu_{E_1}), \ldots, s_\wedge(s(f,t), t.\mu_{E_n})) \ . \tag{32}$$

Equality of scores then follows from the distributivity of $s_\wedge$ over $s_\vee$, which indeed holds only under $\mathcal{FS}$ semantics. It remains to show that $E_L \equiv_b E_R$. This part is straightforward and we omit the detailed steps here. □

Consider the case where $f = p_1 \wedge p_2 \wedge \ldots \wedge p_n$, with p_i over the schema of E_i, and $\Theta_f = [\theta_1, \theta_2, \ldots, \theta_n]$. The following rule shows what happens if predicates are pushed down a Join, provided Join operands are crisp relations:

$$\bowtie^{\Theta_f}(\sigma_{p_1}(E_1), \ldots, \sigma_{p_n}(E_n)) \sqsubseteq \sigma_{f_{\Theta_f}}(\bowtie(E_1, \ldots, E_n)) \ . \tag{33}$$

Proof. Let $t = \bowtie(t_1, \ldots, t_n)$ be a tuple which belongs to both $E_L[db]$ and $E_R[db]$, with $t_i \in E_i[db]$. Since all the E_i's are crisp, it is

$$t.\mu_{E_L} = s_\wedge^{\Theta_f}(s(p_1,t_1), \ldots, s(p_n,t_n))$$

and

$$t.\mu_{E_R} = s(f_{\Theta_f}, t) = s_\wedge^{\Theta_f}(s(p_1,t), \ldots, s(p_n,t)) \ .$$

Since $s(p_i,t) = s(p_i,t_i)$ holds $\forall i \in [1..n]$, equality of scores is guaranteed. Finally, showing that $E_L \sqsubseteq_b E_R$ is trivial. □

On the other hand, when no weights are used it is immediate to conclude, by exploiting the definition of Join and Selection and the associativity of t-norms, that:

$$\bowtie(\sigma_{p_1}(E_1), \ldots, \sigma_{p_n}(E_n)) \equiv \sigma_f(\bowtie(E_1, \ldots, E_n)) \ . \tag{34}$$

7 Other Issues

7.1 Approximation vs Equality

Although SAME^W is a "similarity" algebra, it still preserves some aspects where similarity is not considered at all. This has the advantage of allowing for a clean definition of operators' semantics, yet in some cases a more flexible view would be desirable. In particular, we observe that equality of expressions' results requires equality of scores for corresponding tuples, a fact that in some scenarios can be exceedingly restrictive. In particular, consider the case when alternative similarity operators can be used for a same feature (e.g. $\sim_1$, $\sim_2$, etc.). As an example, the similarity of two color histograms can be assessed by using a complex quadratic-form distance which takes into account colors' cross-correlations [19], but even using the simpler Euclidean distance. In general, if $\sim_1$ is a "cheap" operator, what are the implications of using it in place of a (much) costly $\sim_2$ operator? In the following we provide some preliminary results on the problem of "approximate answers" in SAME^W by relying only on rather generic assumptions, which therefore can be strenghtened on need. The starting point is the definition of ϵ-*compatibility* of scores.

Definition 7.1. A binary relation of ϵ-compatibility over the elements of $\mathcal{S}$, $\simeq_\epsilon$, where $\epsilon \geq 0$, is a symmetric relation closed under interval containment, i.e. a relation which for all $x_1, x_2, x_3 \in \mathcal{S}$ satisfies $x_1 \simeq_\epsilon x_2 \iff x_2 \simeq_\epsilon x_1$ and $x_1 \leq x_2 \leq x_3, x_1 \simeq_\epsilon x_3 \Rightarrow x_1 \simeq_\epsilon x_2, x_2 \simeq_\epsilon x_3$.

Intuitively, ϵ represents the "tolerance" we have on discrepancies of scores. For instance, specific cases of ϵ-compatibility relations are:

$$|x_1 - x_2| \leq \epsilon \iff x_1 \simeq_\epsilon x_2 \quad \text{and} \quad x_2/(1+\epsilon) \leq x_1 \leq x_2(1+\epsilon) \iff x_1 \simeq_\epsilon x_2 \ .$$

The relationship between the different $\simeq_\epsilon$ relations arising from different values of ϵ is established by two basic axioms:

1. $x_1 \simeq_0 x_2 \iff x_1 = x_2$
2. $\epsilon_1 \leq \epsilon_2, x_1 \simeq_{\epsilon_1} x_2 \Rightarrow x_1 \simeq_{\epsilon_2} x_2$.

Above extends to relations and expressions as follows (both e_1 and e_2 are over set of attributes X):

$$e_1 \simeq_\epsilon e_2 \iff \pi_X(e_1) = \pi_X(e_2) \ \wedge \ \forall t_1 \in e_1, \exists t_2 \in e_2 : t_1 = t_2 \ \wedge \ t_1.\mu \simeq_\epsilon t_2.\mu$$
$$E_1 \cong_\epsilon E_2 \iff \forall db : \ E_1[db] \simeq_\epsilon E_2[db] \ .$$

Turning to consider algebraic operators, a basic problem is to understand how they influence errors on scores. The following holds for the $\mathcal{FS}$ semantics:

Lemma 7.1. *Let* $x_i \simeq_{\epsilon_i} x'_i$, $i \in [1..n]$. *Then,* $\min(x_1, \ldots, x_n) \simeq_\epsilon \min(x'_1, \ldots, x'_n)$ *and* $\max(x_1, \ldots, x_n) \simeq_\epsilon \max(x'_1, \ldots, x'_n)$ *both hold, where* $\epsilon = \max(\epsilon_1, \ldots, \epsilon_n)$.

Proof. We prove the statement for the min scoring function, being the case for max analogous. Since $\epsilon_i \leq \epsilon$, it follows that $x_i \simeq_{\epsilon_i} x'_i$ implies $x_i \simeq_\epsilon x'_i$. Assume that $\min(x_1, \ldots, x_n) = x_i < x_j$ and that $\min(x'_1, \ldots, x'_n) = x'_j < x'_i$, otherwise the result would trivially hold by hypothesis. Without loss of generality, let $x'_j < x_i$. Since $x'_j \simeq_\epsilon x_j$, from $x'_j < x_i$ and $x_i < x_j$ we derive that both $x'_j \simeq_\epsilon x_i$ and $x_i \simeq_\epsilon x_j$ hold, thus proving the result. □

We have the following result which shows that errors will stay limited to ϵ even for arbitrarily complex SPJU (Select, Project, Join, Union) queries.

Theorem 7.1. *Let E be a SAME^W expression with no weights, no negated predicates, and using operators in $\{\sigma, \pi, \bowtie, \cup\}$, and let E' be an expression obtained from E by changing a similarity operator from $\sim$ to $\sim'$ so that $(v_1 \sim v_2) \simeq_\epsilon (v_1 \sim' v_2)$ holds for any pair of values v_1, v_2 in the domain of $\sim$. Then $E \cong_\epsilon E'$ holds under $\mathcal{FS}$ semantics.*

Proof. (sketch) The complete proof considers each operator in $\{\sigma, \pi, \bowtie, \cup\}$. Here we limit the analysis to the Selection operator. Consider the two expressions $E = \sigma_p(E_1)$, where $p : A \sim v$, and E' obtained from E by replacing $\sim$ with $\sim'$. By definition of Selection under $\mathcal{FS}$ semantics, the scores of tuples satisfying E and E' are respectively given by $\min(s(A \sim v, t), t.\mu_{E_1})$ and $\min(s(A \sim' v, t), t.\mu_{E_1})$. By hypothesis, it is $s(A \sim v, t) \simeq_\epsilon s(A \sim' v, t)$. From Lemma 7.1 it follows that $\min(s(A \sim v, t), t.\mu_{E_1}) \simeq_\epsilon \min(s(A \sim' v, t), t.\mu_{E_1})$. This is to say that $\sigma_{A \sim v}(E_1) \cong_\epsilon \sigma_{A \sim' v}(E_1)$. □

Extending the Theorem to the general case of *weighted* SPJU expressions is indeed possible (we omit the proof here). For this, however, we need the following additional assumption of *linearity* on the behavior of ϵ-compatibility relations:

$$x_1 \simeq_\epsilon x'_1, x_2 \simeq_\epsilon x'_2 \quad \Rightarrow \quad \beta \cdot x_1 + (1-\beta) \cdot x_2 \simeq_\epsilon \beta \cdot x'_1 + (1-\beta) \cdot x'_2 \quad (\beta \in [0,1]) \ .$$

Intuitively, linearity means that if discrepancies of scores stay limited at the extremes of an interval, then they are also limited in the whole interval, which seems quite reasonable to demand. Note that both sample ϵ-compatibility relations we have considered are linear, as it can be easily proved.

The Theorem does not apply to expressions with negated predicates and/or Difference. For instance, consider the second sample ϵ-compatibility relation $(x_1 \in [x_2/(1+\epsilon), x_2(1+\epsilon)])$, and take $x_1 = 0.8$, $x_2 = 0.9$, and $\epsilon = 0.2$. Clearly $0.8 \simeq_{0.2} 0.9$, but $(1 - 0.8) \not\simeq_{0.2} (1 - 0.9)$.

Finally, the reason why the Theorem does not extend to expressions using Cut and Top is that such operators explicitly consider tuples' scores for determining which tuples are to be part of the result. For the Cut, however, the relationship between different similarity operators can be exploited as follows.[13] Assume that $\sim'$ is *weaker* than $\sim$. This means that, for any pair of values v_1, v_2 in the domain

[13] This is tightly related to "filter-and-refine" strategies used to speed-up the processing of range queries over complex (multi-dimensional) domains, where one adopts a "simplified" distance function which lower bounds the original one [2].

of $\sim$, $(v_1 \sim' v_2) \geq (v_1 \sim v_2)$ holds. Then the following applies, where π_X is just used to get rid of scores, $p : A \sim v$, and $p' : A \sim' v$:

$$\gamma_\alpha(\sigma_p(\pi_X(E))) \equiv \gamma_\alpha(\sigma_p(\pi_X(\gamma_\alpha(\sigma_{p'}(\pi_X(E)))))) \quad . \tag{35}$$

If $\sim'$ is cheaper to evaluate than $\sim$, one can use the expression on the right to first filter out all those tuples that do not satisfy the condition p' at least with score $> \alpha$. Since $\sim'$ is weaker than $\sim$, the result of this first filter is guaranteed to contain all the tuples returned by the expression on the left.

7.2 Adapting Weights to User Evaluation

As a final issue, we consider the case where the result of an expression E is "evaluated" by the user, who assigns to each tuple $t_i \in e = E[db]$ a "goodness" value g_i. Basically, g_i represents how much the user considers t_i "relevant" for his information needs. The fact that the result of a query does not exactly match user expectation is common in MM systems, and leads to a "closed-loop" interactive process, where user evaluation is fed back to the query engine and then taken into account to compute a (possibly) "better" result, and so on.

For our purpose, *relevance feedback* algorithms that have been proposed in the Information Retrieval (IR) field are not appropriate for two reasons. First, they do not apply to complex algebraic expressions [11]. More precisely, such algorithms work only for keyword-based similarity search, when queries and documents are interpreted as vectors and weights as coordinate values. Second, they do not explicitly consider weights at predicate and expression levels. Recently, Ishikawa et al. [12] have proposed a method ("MindReader") to compute an optimal set of weights (according to a specific objective function), given user goodness values and assuming a "range query" whose shape indeed depends on weights. In our context, this amounts to "guess" the best similarity operator to be used for a certain feature, a problem somewhat orthogonal to the one we consider here.

The generic framework we propose to deal with user judgments is as follows. Let $E_{\{\Theta_j\}}$ be an expression using a set $\{\Theta_j\}$ of weightings, and let $e = E_{\{\Theta_j\}}[db]$. For each tuple $t_i \in e$, let $\mu_i = \mu_{E_{\{\Theta_j\}}}(t_i) = t_i.\mu_{E_{\{\Theta_j\}}}$, and let $g_i = G(t_i)$ be the "goodness" that the user assigns to t_i (G is called a *goodness function*). Then, consider an *objective function* $\mathcal{O}$, $\mathcal{O}(G, \mu_{E_{\{\Theta_j\}}}; e) \in \Re$, that measures the overall goodness of e. A set of weightings $\{\Theta_j^{opt}\}$ is called *optimal* with respect to $\mathcal{O}$ if:

$$\mathcal{O}(G, \mu_{E_{\{\Theta_j^{opt}\}}}; e) \geq \mathcal{O}(G, \mu_{E_{\{\Theta_j\}}}; e) \qquad \forall \{\Theta_j\} \quad .$$

As an example, assume that $\mathcal{O}$ is a function which is maximum when $\mu_i = g_i$ $\forall i$ (thus the g_i's are values in $\mathcal{S}$), say, $\mathcal{O}(G, \mu_{E_{\{\Theta_j\}}}; e) = -\sum_i (\mu_i - g_i)^2$, and consider the expression $\sigma_{f_1}(R_1) \cup^{[\theta_1,\theta_2]} \sigma_{f_2}(R_2)$, where we take the weighted Union of two (crisp) relations, to which the (unweighted) formulas f_1 and f_2 are applied. One can think of this as a weighted integration of results from different

search engines, where f_1 and f_2 are tailored to the engine at hand, and $[\theta_1, \theta_2]$ are the initial weights we assign to the two engines, reflecting how much we "trust" them. The objective is therefore to minimize:

$$\sum_i \left(s_{\vee}^{[\theta_1,\theta_2]}(s_{i,1}, s_{i,2}) - g_i \right)^2$$

where $s_{i,1} = s(f_1, t_i)$ and $s_{i,2} = s(f_2, t_i)$. Since numerical methods exist to minimize above expression, we can consequently determine optimal values for θ_1 and θ_2, that is, the relative importance to be assigned to the two engines in order to make the overall scores as close as possible to their goodness values.

Although we have not explored yet all the (numerical) intricacies of adjusting weights to maximize $\mathcal{O}$, we claim that *it is indeed possible to compute* $\{\Theta_j^{opt}\}$ *for any* SAME^W *expression*. Clearly, the real challenge here appears to be the determination of a good tradeoff between the "soundness" of the objective function and the corresponding cost of computing (at query time) an optimal set of weightings.

8 Related Work

In the last two decades, many works have focused on problems related to extending data models so as to allow representation of "imprecision" in databases. These works are only marginally relevant here, since they mostly concentrate on modeling aspects (see e.g. [18]) and typically ignore issues related to advanced processing of similarity queries, which are a major concern in multimedia environments. Indeed, it is a fact that similarity queries arise even if the database does not store imprecise information at all, provided "similarity operators" are defined. This is also the scenario considered by the VAGUE system [16], where, however, important features are missing, such as weights, the Top operator, and fuzzy attributes. Further, problems related to query optimization are not considered in [16]. Recent work by Adali et al. [1] addresses issues similar to ours, but important differences exist. First, they do not consider weights, that we have shown to introduce new interesting problems in the query optimization scenario. Second, they are mainly concentrated on problems related to the *integration* of heterogeneous "similarity measures", coming from different sources, into a common framework. This results in a quite complex "multi-level" algebraic scenario, on which reasoning becomes difficult and tricky. In this light, we believe that SAME^W is much "cleaner", and that it highly simplifies the (inherently difficult) task of reasoning about similarity queries.

Finally, several works are related to ours from the point of view of query processing and execution. Besides Fagin's works, here we mention our previous work in the area and recent activity related to "Top queries". In [7] we have considered problems related to the efficient index-based execution of similarity queries, where all the predicates refer to a single feature (e.g. find objects which

are similar to *this* shape, but not to *that* one). Results in [7] show how important performance improvements are obtainable by using access methods able to process the "query as a whole", rather than one predicate at a time.

Recent research on so-called Top queries [3] addresses issues arising in a DBMS when the cardinality of the result is limited by the user. In this case, it is assumed that the result set is sorted (using the `ORDER BY` SQL clause), and that the result stream is stopped after k tuples have been produced. There is a tight connection here with our Top (τ) operator, with our "ranking criterion" playing the role of the `ORDER BY` clause. Optimization techniques considered in [3] to "push-down" the Top basically exploit primary key-foreign key joins (as well as analysis of residual predicates) – a thing which we could embed into SAME^W by means of functional dependencies.

9 Conclusion

In this paper we have introduced a "similarity algebra with weights", called SAME^W, that generalizes relational algebra to allow the formulation of complex similarity queries over multimedia databases. SAME^W combines within a single framework several aspects relevant to multimedia queries, such as new operators (Cut and Top) useful for "range" and "best-matches" queries, weights to express user preferences, and "scores" to rank tuples. These aspects, together with other issues which we have only partially addressed here, such as "approximate results" (see Sect. 7.1) and user evaluation (Sect. 7.2), pose new challenges to a query engine, which have not been considered yet in their full generality. For instance, if the user evaluates the result of a query, thus the system should adapt to this by adjusting weights, how does this affect execution costs? Indeed, as shown in Sect. 6, changing the weights will modify the numerical values, thus the processing costs, used by some operators (like the Cut) to limit the cardinality of the arguments of n-ary operators, such as Join and Union.

A point which would also deserve a much more careful investigation concerns the definition of "notions of approximation" which are both practical and useful. This is an issue whose importance is likely to considerably grow in the near future, and that have only partially been addressed in recent years [20, 5].

References

1. S. Adali, P. Bonatti, M.L. Sapino, and V.S. Subrahmanian. A Multi-Similarity Algebra. In *Proc. of the 1998 ACM-SIGMOD Int. Conf. on Management of Data*, pages 402–413, Seattle, WA, June 1998.
2. R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. In *Proc. of the 4th Int. Conf. on Foundations of Data Organizations and Algorithms (FODO'93)*, pages 69–84, Chicago, IL, October 1993.
3. M.J. Carey and D. Kossmann. On Saying "Enough Already!" in SQL. In *Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of Data*, pages 219–230, Tucson, AZ, May 1997.

4. P. Ciaccia, D. Montesi, W. Penzo, and A. Trombetta. SAME^W: A Fuzzy Similarity Algebra for Web and Multimedia Databases. Technical Report T2-R26, InterData project, 1999. Available at URL `ftp://ftp-db.deis.unibo.it/pub/interdata/tema2/T2-R26.ps`.
5. P. Ciaccia and M. Patella. PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces. In *Proc. of the 16th Int. Conf. on Data Engineering (ICDE 2000)*, San Diego, CA, March 2000.
6. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proc. of the 23rd VLDB Int. Conf.*, pages 426–435, Athens, Greece, August 1997.
7. P. Ciaccia, M. Patella, and P. Zezula. Processing Complex Similarity Queries with Distance-based Access Methods. In *Proc. of the 6th Int. Conf. on Extending Database Technology (EDBT'98)*, pages 9–23, Valencia, Spain, March 1998.
8. R. Fagin. Combining Fuzzy Information from Multiple Systems. In *Proc. of the 15th ACM Symposium on Principles of Database Systems (PODS'96)*, pages 216–226, Montreal, Canada, June 1996.
9. R. Fagin and E.L. Wimmers. Incorporating User Preferences in Multimedia Queries. In *Proc. of the 6th ICDT Int. Conf.*, pages 247–261, Delphi, Greece, January 1997.
10. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23–32, September 1995.
11. D. Harman. Relevance Feedback and Other Query Modification Techniques. In W.B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, chapter 11, pages 241–263. Prentice Hall PTR, 1992.
12. Y. Ishikawa, R. Subramanya, and C. Faloutsos. MindReader: Querying Databases through Multiple Examples. In *Proc. of the 24th VLDB Int. Conf.*, pages 218–227, New York, NY, August 1998.
13. G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice Hall PTR, 1995.
14. A. Lumini, D. Maio, and D. Maltoni. Continuous versus Exclusive Classification for Fingerprint Retrieval. *Pattern Recognition Letters*, 18:1027–1034, 1997.
15. D. Montesi and A. Trombetta. Similarity Search through Fuzzy Relational Algebra. In *Proc. of the 1st Int. Workshop on Similarity Search (IWOSS'99)*, Florence, Italy, September 1999.
16. A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Trans. on Office Information Systems*, 6(3):187–214, July 1988.
17. S. Nepal, M.V. Ramakrishna, and J.A. Thom. A Fuzzy Object Language (FOQL) for Image Databases. In *Proc. of the 6th Int. Conf. on Database Systems for Advanced Applications (DASFAA'99)*, pages 117–124, Hsinchu, Taiwan, April 1999.
18. K. Raju and A. Majumdar. Fuzzy Functional Dependencies and Lossless Join Decomposition of Fuzzy Relational Database Systems. *ACM Trans. on Database Systems*, 13(32):129–166, June 1988.
19. T. Seidl and H.-P. Kriegel. Efficient User-Adaptable Similarity Search in Large Multimedia Databases. In *Proc. of the 23rd VLDB Int. Conf.*, pages 506–515, Athens, Greece, August 1997.
20. N. Shivakumar, H. Garcia-Molina, and C.S. Chekuri. Filtering with Approximate Predicates. In *Proc. of the 24th VLDB Int. Conf.*, pages 263–274, New York, NY, August 1998.
21. A. Soffer and H. Samet. Integrating Symbolic Images into a Multimedia Database System using Classification and Abstraction Approaches. *The VLDB Journal*, 7(4):253–274, 1998.