F-G: A Fuzzy Algebra for Approximate Answering in Databases

Berthier A.N. Ribeiro* Richard Muntz†
University of California, Los Angeles

1 Introduction: The Problem

Conventional database systems deal with exact queries which specify constraints that have to be met precisely. For instance, if a user asks for a *suspense* film playing in Hollywood and there is no such film in the database, the user receives a null answer. However, it might be the case that there is a suspense film playing in a theater located in Beverly Hills, which is only 8 miles away. Conventional systems have no mechanism to even specify an approximate constraint such as [location *near-to* Hollywood].

Since conventional database systems cannot handle approximate queries directly, these have to be emulated through a sequence of exact queries. But then, the user has to direct the system through a possibly long series of interactions to satisfy his needs. If the user is not aware of close alternatives, then even this solution is infeasible [5].

An approximate query answering system allows the user to specify a query containing *vague* conditions. For instance, $q = [film\_category \sim suspense] \wedge [location \sim Hollywood]$. Vague queries require the system to find tuples that approximate the query conditions and rank (i.e., order) these tuples according to their *semantic distance* to the query.

A data metric is a function that specifies a *semantic* distance between any two instances from the same attribute domain. We assume that data metrics for the attribute domains in the database are provided. Given a query q formed by conjunctions and disjunctions of vague conditions, one still has to address the problem of *how to combine* metric values from distinct constraints into a final ranking. For instance, consider query q above. If the film attribute value of a tuple t_k approximates *suspense* and the theater location attribute value approximates *Hollywood*, the system has to combine the respective metric values to generate a rank for t_k. The problem is non-trivial and most approaches consist of heuristics that can not be easily grasped by the user or justified in any formal sense.

Fuhr [3] uses a *relevance description* function to model approximation neighborhoods (i.e., data metrics). However, his approach does *not* allow Boolean operators in the query formulation because a query involving *conjuncts* and *disjuncts* would lead to complex probabilistic formulas with parameters that could not be accurately estimated in most cases. Thus, his approach simply avoids the problem that we are addressing. Rabitti and Savino [6] propose to sum up the contributions of an object to the different portions of an *or* condition. An *and* operation is satisfied or not satisfied depending on whether the object approximates all conjunctive conditions better than a minimum similarity threshold i.e., there is a binary decision with no discrimination among those objects that are judged to satisfy the conjunct. Therefore, an object O_1 which barely satisfies the thresholds and another object O_2 which matches all conditions in the *and* operation exactly would receive the same ranking thus looming the user to rank the (possibly) large response set. This solution is clearly inappropriate for large databases. Motro [5] proposes to combine metric distances in an *and* operation by a weighted square sum of the distances for each constraint. However, the meaning of computing the overall distance for an *and* condition by a weighted sum is not clear.

Our work proposes an extended fuzzy algebra called *F-G algebra* that, instead of attempting to combine metric values quantifying individual qualifiers to a scalar value, generates a list of scalar values which can be interpreted according to the query semantics. These lists are compared to decide ordering in the final ranking. F-G algebra is derived from two basic principles that are almost universally accepted: (a) the overall semantic distance measure for a conjunction should match or exceed the semantic distance measure for any of its operands and (b) the overall semantic distance measure for a disjunction should not exceed the semantic

*Supported in part by Organization of American States grant F29180.
†Supported in part by NASA grant NAG5-2225.

distance measure for any of its operands. In addition, the F-G algebra also preserves desirable properties such as commutativity, associativity, and distributivity. Further, instead of relying on the data metrics provided, the user has the option of stating preference relationships among attribute values in qualitative terms (e.g., greater than, lesser than, etc) and use the F-G algebra to order the set of approximate answers according to those preferences. This is important for two reasons: (1) the user might have difficulties in expressing his belief in terms of numeric values [8] and (2) the ranking generated by the F-G algebra is inherently qualitative which, we believe, simplifies its understanding by the user.

The relevance of our research derives from the strong current interest in more sophisticated information management systems. Many applications simply cannot perform well without some notion of approximation built into the information system. Cuppens and Demolombe [2] discuss the importance of approximate queries in decision support systems and advice giving systems. Fuhr [3] describes their use in engineering applications, materials data systems, and business decision making systems. Rabitti and Savino [6] point to image databases. Further, the tremendous growing of the Internet and the advent of the World Wide Web have made access to world-wide information possible. Since people frequently do not know exactly what they are looking for (or at least have to express what they are looking for precisely), it is crucial that the system implements some form of approximation. In general, the areas of approximate, cooperative, incomplete, and uncertain information management can benefit from our work.

The rest of the paper is organized as follows. Section 2 introduces a fuzzy set model for a conventional relational database. The model is straightforward and allows only exact answers. Section 3 extends the model to allow approximate answers and discusses the inadequacy of conventional fuzzy theory for ranking the responses. Section 4 introduces the F-G algebra and its main properties. Our conclusions follow.

2 Fuzzy Set Model for a Conventional Database

In this section we introduce a fuzzy set model for a conventional relational database. The model is straightforward and allows only exact answers. In section 3, we extend the model to allow approximate answers.

Definition 1 *Let A_i be the ith attribute name in a given relation R and let v_j be a value from the respective domain. The pair $[A_i, v_j]$ is called an attribute-value pair (avp) and is referred to as $\vartheta_{i,j}$. We adopt the simplified notation ϑ_j whenever the attribute name is implicitly clear. Let t_k be a tuple of relation R. The notation $t_k[i]$ refers to the value of attribute A_i according to tuple t_k.*

Definition 2 *A query q is a general boolean expression composed of conjuncts and disjuncts of avps. For a given query q, if $\vartheta_{i,j} \in q$ then $\vartheta_{i,j}$ is called a query avp. A query avp $\vartheta_{i,j}$ is modelled as a fuzzy set. For exact answers, a tuple t_k is said to be a member of this set only if $t_k[i] = \vartheta_{i,j}$. This membership is quantified by the grade of membership denoted by $\mu_{\vartheta_{i,j},t_k}$ and defined as*

$$\mu_{\vartheta_{i,j},t_k} = \begin{cases} 1 & if\; t_k[i] = \vartheta_{i,j} \\ 0 & otherwise \end{cases} \tag{1}$$

Definition 3 *A query q is modelled as a fuzzy set formed by the union/intersection of query avp fuzzy sets. The union operation models query disjunctions while the intersection operation models query conjunctions.*

Figure 1 illustrates our fuzzy set model for the query $(\vartheta_{7,1} \vee \vartheta_{7,2}) \wedge \vartheta_{3,1}$. The ovals indicate fuzzy sets. Tuple t_k has membership in the avp fuzzy sets $\vartheta_{7,2}$ and $\vartheta_{3,1}$.

According to Zadeh [9], "*a grade of membership $\mu_{V,u}$ may be interpreted as the degree of compatibility of u with the concept represented by V*". The following definition establishes a parallel between this idea and our fuzzy set model for the database.

Definition 4 *The degree of compatibility of a tuple t_k with the concept represented by a query q is interpreted as the grade of membership of tuple t_k in the query fuzzy set. This grade of membership is denoted by μ_{q,t_k}. According to conventional fuzzy theory, μ_{q,t_k} is computed through min/max operations on the grades of membership of tuple t_k in the fuzzy sets for query avps.*

For the query in figure 1, classic fuzzy theory yields $\mu_{q,t_k} = min(\mu_{q_{or},t_k}, \mu_{\vartheta_{3,1},t_k}) = min(max(\mu_{\vartheta_{7,1},t_k}, \mu_{\vartheta_{7,2},t_k}), \mu_{\vartheta_{3,1},t_k})$. The *min* operation computes the grade of membership for conjunctions (i.e., intersection of fuzzy sets), while the *max* operation computes the grade of membership for disjunctions (i.e., union of fuzzy sets). Given a query q and a tuple t_k, μ_{q,t_k} can be computed trivially.

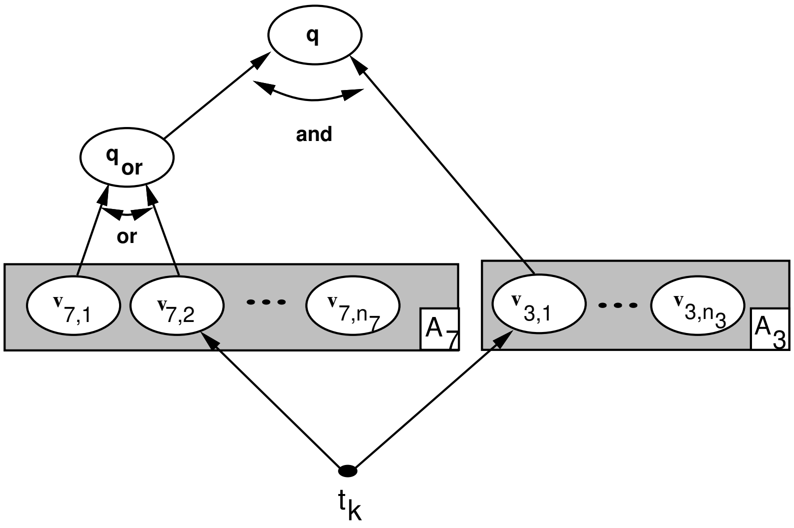

Figure 1: Fuzzy set model for query $q = (\vartheta_{7,1} \vee \vartheta_{7,2}) \wedge \vartheta_{3,1}$.

Lemma 1 *Let $\mathcal{T}_q$ be the set of tuples retrieved by a conventional relational database system in response to a query q. For each tuple $t_k \in \mathcal{T}_q$, our fuzzy set model returns $\mu_{q,t_k} = 1$. Further, for each tuple $t_n \notin \mathcal{T}_q$, our fuzzy set model returns $\mu_{q,t_n} = 0$.*

Proof Trivial. □

3 Fuzzy Set Model for Approximate Answering

In this section we extend our model with the introduction of grades of membership other than 1 and 0. This allows modeling partial matches (i.e., approximation) between a tuple t_k and a query q. We show the inadequacy of conventional fuzzy theory to rank tuples as approximate answers for a conjunctive/disjunctive database query. In section 4 we propose an extension to fuzzy theory called *F-G algebra* that overcomes this problem.

Measuring similarity requires the design of *data metrics*. A data metric is a function $M : D \times D \rightarrow R$ that specifies a *semantic* distance between any two values in the same attribute domain [5]. In this paper, we do not concern ourselves with the design of data metrics for a database[1]; we assume that they are provided. Given a data metric, we can estimate the *degree of similarity* between any two values in the same attribute domain which can then be used as a grade of membership in our fuzzy set model. This degree of similarity can be understood as the inverse of the semantic distance defined by the data metric.

Definition 5 *Let D_i be the domain of an attribute A_i. Also, let $\vartheta_{i,v} \in D_i$ be a query avp. The corresponding query avp fuzzy set is now referred to as $\Upsilon_{i,v}$ to indicate that approximate matches are allowed. A tuple t_k is said to be a member of the set $\Upsilon_{i,v}$ if $t_k[i] \sim \vartheta_{i,v}$ i.e., if the component of tuple t_k for domain D_i approximates the query avp $\vartheta_{i,v}$. The grade of membership of tuple t_k in the set $\Upsilon_{i,v}$ is referred to as $\mu_{\Upsilon_{i,v},t_k}$. If $\mu_{\Upsilon_{i,v},t_k} > 0$, we say that tuple t_k is a* neighbor *of avp $\vartheta_{i,v}$. The set $\Upsilon_{i,v}$, formed by all the neighbor tuples of avp $\vartheta_{i,v}$, is called a* neighborhood.

Consider a database for films and theaters [5] and let q be the vague query: (location $\sim$ Beverly_Hills $\vee$ location $\sim$ Downtown) $\wedge$ (category $\sim$ adventure). The operator '$\sim$' stands for *similar-to*. Figure 2 illustrates the corresponding fuzzy model. Tuple t_1 refers to a *drama* film which is being shown at a theater in Beverly Hills. Tuple t_2 refers to another *drama* film which is being shown at a theater in Westwood. Tuple t_1 is a member of the neighborhood Υ_{bh} because $t_1[7] = bh$. Further, if $t_1[7]$ approximates avp ϑ_{ww} (i.e., $t_1[7]$ is a neighbor of ϑ_{ww}), then tuple t_1 is also a member of the neighborhood Υ_{ww}.

[1]Motro [5] discusses data metrics in detail and suggests techniques for generating them automatically.

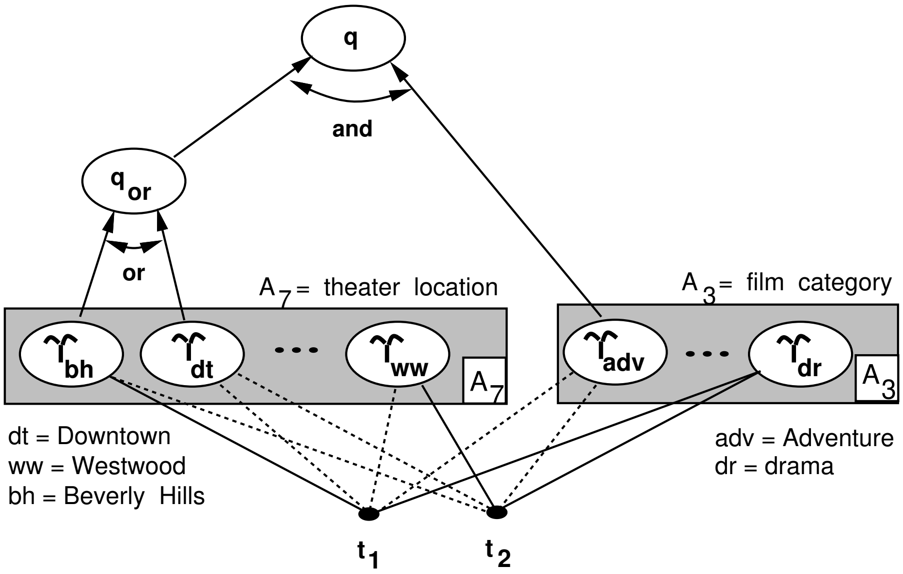

Figure 2: Fuzzy model for query $q = (A_7 \sim bh \vee A_7 \sim dt) \wedge A_3 \sim adv$. Tuple $t_1 = \{A_3 = dr, A_7 = bh\}$ and tuple $t_2 = \{A_3 = dr, A_7 = ww\}$.

The main question we face is how to quantify the *degree of similarity* between a tuple t_k and a database query q (i.e., μ_{q,t_k}) when the tuple *approximates* the individual components of the query to varying degrees. In the following, we discuss the inadequacy of classic fuzzy theory for computing μ_{q,t_k}.

In figure 2, assume that grades of membership are as follows: $\mu_{\Upsilon_{bh},t_1} = 1$, $\mu_{\Upsilon_{dt},t_1} = 0.6$, $\mu_{\Upsilon_{adv},t_1} = 0.45$, $\mu_{\Upsilon_{bh},t_2} = 0.72$, $\mu_{\Upsilon_{dt},t_2} = 0.5$, and $\mu_{\Upsilon_{adv},t_2} = 0.45$. For query q, classic fuzzy theory yields: $\mu_{q,t_1} = min(max(1, 0.6), 0.45) = 0.45$ and $\mu_{q,t_2} = min(max(0.72, 0.5), 0.45) = 0.45$. Thus, $\mu_{q,t_1} = \mu_{q,t_2}$ and their rankings coincide. However, tuple t_1 provides a film in a closer location than tuple t_2. The problem with classic fuzzy theory is that its operators (i.e., min and max) forget about *less* important operands.

In a large database, relations might include thousands of tuples. Given an approximate query composed of multiple avps, conventional fuzzy theory might give many tuples the same rank because it fails to consider the impact of second order operands. In the following, we introduce the F-G algebra and show how it addresses this problem.

4 F-G Algebra

In this section we extend classic fuzzy theory with a fuzzy algebra that we call F-G. The F-G algebra improves the ranking of approximate answers because, contrary to classic fuzzy theory, it considers the effect of second order factors in the rank. Further, it preserves desirable properties such as commutativity, associativity, and distributivity.

Several lemmas are stated here without proof. The proofs can be found in [7].

4.1 Basic Definitions

Let F and G be two binary operations on avps with grades of membership in the range [0,1]. Given an avp x, μ_x stands for its grade of membership in a query avp fuzzy set. In [1] and [4] it was shown that F = min, G = max are the unique solutions that satisfy the following conditions:

1) associativity: $F(x, F(y, z)) = F(F(x, y), z)$, $G(x, G(y, z)) = G(G(x, y), z)$;
2) commutative: $F(x, y) = F(y, x)$, $G(x, y) = G(y, x)$;
3) nondecreasing: $F(x_1, y_1) \leq F(x_2, y_2)$, $G(x_1, y_1) \leq G(x_2, y_2)$, if $\mu_{x_1} \leq \mu_{x_2}$ and $\mu_{y_1} \leq \mu_{y_2}$;
4) $F(x_1, x_1) < F(x_2, x_2)$, $G(x_1, x_1) < G(x_2, x_2)$, if $\mu_{x_1} < \mu_{x_2}$;
5) $F(x, y) = 1$ if $\mu_x = 1$ and $\mu_y = 1$, $G(x, y) = 0$ if $\mu_x = 0$ and $\mu_y = 0$;
6) $F(x, y) \leq min(\mu_x, \mu_y)$, $G(x, y) \geq max(\mu_x, \mu_y)$;

7) distributivity: $F(x, G(y, z)) = G(F(x, y), F(x, z))$, $G(x, F(y, z)) = F(G(x, y), G(x, z))$.
These properties are desirable in a query answering system because of the flexibility they provide for expressing the query intention. Particularly important is property (6). It establishes that the degree of similarity associated with the result of an *and* operation (i.e., an F) cannot exceed the degree of similarity (i.e., grade of membership) of any of its operands which is quite expected from an intuitive point of view. Further, the degree of similarity associated to the result of an *or* operation (i.e., a G) must at least match the largest degree of similarity among all of its operands.

Definition 6 *Let θ (i.e., min) and ω (i.e., max) be two binary operators defined as follows:*

$$\theta(x, y) = \begin{cases} x & \text{if } \mu_x < \mu_y \\ y & \text{otherwise} \end{cases} \qquad \omega(x, y) = \begin{cases} x & \text{if } \mu_x > \mu_y \\ y & \text{otherwise} \end{cases} \tag{2}$$

where x and y are two avps. We redefine F as

$$F(x, y) = [\theta(x, y); \omega(x, y)] \tag{3}$$

where the notation [;] stands for an ordered pair which we call a peerage *(i.e., a list of peers).*

Thus, our new F function takes a pair of avps from a tuple t_k and orders them by increasing grades of membership. This is the semantics of an *and* operation in the F-G algebra. The resultant ordered pair is used to determine the rank of tuple t_k with respect to the F expression (which should be understood as the query). Since the ranks are now pairs, we have to specify how to compare them.

Definition 7 *Let a and A be components of a tuple t_a and let $[a; A]$ be the peerage generated by the function $F(a, A)$. Clearly, $\mu_a < \mu_A$ (for simplicity, throughout the paper, small and capital versions of a same letter indicate that the small letter avp has lower grade of membership than the capital letter avp). Let $[b; B]$ be the peerage generated for another tuple t_b and let '$\cdot \succ$' be a preference relation on tuples. If $\mu_a > \mu_b$ then $t_a \cdot \succ t_b$. If $\mu_a < \mu_b$ then $t_b \cdot \succ t_a$. In case $\mu_a = \mu_b$, the preference is determined by comparing μ_A and μ_B. If $(\mu_a = \mu_b \wedge \mu_A = \mu_B)$ then $t_a \doteq t_b$ i.e., the tuples are equally preferred.*

Notice that preference (i.e., rank) is still determined by the dominating avp (i.e., the leftmost). However, in case of a tie, the preference is determined by the next avp to the right.

Definition 8 *We redefine G as*

$$G(x, y) = [\omega(x, y); \theta(x, y)] \tag{4}$$

The new G function is complimentary to the F function and defines the semantics of an *or* operation. Given a pair of avps, it gives highest priority to the avp with largest grade of membership.

The F-G algebra preserves all of its operands and their grades of membership in the body of the computed peerage. The operands are ordered according to the semantics of the operation applied to them. An F operation orders the pair of operands by increasing grades of membership. A G operation orders the operands by decreasing grades of membership. In the following, we show how to generalize these ideas to more complex expressions involving multiple references to either F or G. Later, we discuss how to deal with expressions that mix references to both F and G.

Let us extend F and G for dealing with triples of avps.

Definition 9

$$\begin{aligned} F(x, F(a, A)) &= F(x, [a; A]) \\ &= [\theta(x, [a; A]); \omega(x, [a; A])] \end{aligned} \tag{5}$$

$$\begin{aligned} G(x, G(a, A)) &= G(x, [A; a]) \\ &= [\omega(x, [A; a]); \theta(x, [A; a])] \end{aligned} \tag{6}$$

Application of F and G for triples is defined as a natural extension of our previous definitions but requires extending the definitions of the operators ω and θ. We first extend θ and ω to deal with a single avp x and a peerage.

Definition 10 *Let x be an avp and $\sqcap$ be a generic peerage. The operation $\theta(x, \sqcap)$ is computed by the following procedure: (1) traverse $\sqcap$ from left to right preserving the symbols $\underline{[}$, $\underline{]}$, $\underline{;}$, and their ordering; (2) substitute each avp $\vartheta_\sqcap$ encountered for $\theta(x, \vartheta_\sqcap)$. Computation of $\omega(x, \sqcap)$ is defined analogously.*

To understand the motivation behind this generalization consider the peerage expression for $F(x, F(a, A))$. Using the definition above, we can write

$$F(x, F(a, A)) = [\theta(x, [a; A]); \omega(x, [a; A])] \quad (7)$$

$$= [[\theta(x, a); \theta(x, A)]; [\omega(x, a); \omega(x, A)]] \quad (8)$$

because $\mu_a < \mu_A$. Three are the possibilities for instantiation of this peerage: (1) $[[x; x]; [a; A]]$ if $\mu_x < \mu_a$, (2) $[[a; x]; [x; A]]$ if $\mu_a < \mu_x < \mu_A$, and (3) $[[a; A]; [x; x]]$ if $\mu_A < \mu_x$. Regardless of their relative ordering, traversal of the peerage from left to right recovers the avps ordered by increasing grades of membership. Thus, our extension of θ and ω preserves the property, associated to the function F, of retrieving the avps ordered by increasing grades of membership. Analogously, the expression $G(x, G(a, A))$ generates a peerage whose traversal from left to right recovers the avps ordered by decreasing grades of membership.

We now extend the operators θ and ω to deal with pairs of peerages.

Definition 11 *Let $\sqcap_a$ and $\sqcap_b$ be two generic peerages. Then, $\theta(\sqcap_a, \sqcap_b)$ is computed by the following procedure: (1) traverse $\sqcap_a$ from left to right preserving the symbols $\underline{[}$, $\underline{]}$, $\underline{;}$, and their ordering; (2) substitute each avp $\vartheta_{\sqcap_a}$ encountered by $\theta(\vartheta_{\sqcap_a}, \sqcap_b)$. Computation of $\omega(\sqcap_a, \sqcap_b)$ is defined analogously.*

Later, we demonstrate that this extension preserves the property of retrieving avps ordered by increasing grades of membership for the F function and by decreasing grades of membership for the G function.

The F-G algebra can generate peerages that are more complex than simple ordered pairs. Comparing complex peerages requires a special procedure that we now describe.

Definition 12 *Let $\sqcap$ be a generic peerage. Consider traversing $\sqcap$ from left to right and collecting symbols according to the following rules: (a) neglect '[', ']', and ';', (b) insert each avp encountered in a list ∇ if the avp is not already there. The list $\nabla_\sqcap$ defines an ordering among the distinct avps in $\sqcap$. Let $\sqcap_{t_1}$ be the peerage generated for a tuple t_1 and $\sqcap_{t_2}$ be the peerage generated for a tuple t_2. The respective orderings of distinct avps are $\nabla_{\sqcap_{t_1}}$ and $\nabla_{\sqcap_{t_2}}$. We compare two ∇ lists by comparing the first avp in one with the first avp in the other. If their grades of membership are equal, we then compare the second avp in both lists. We repeat this comparision until an inequality or the end of the ∇ lists is reached. Let $t_1[j] \in \nabla_{\sqcap_{t_1}}$ and $t_2[k] \in \nabla_{\sqcap_{t_2}}$ be the last pair of avps compared. Then,*

$$\begin{cases} t_1 \cdot\succ t_2 & \text{if } \mu_{t_1[j]} > \mu_{t_2[k]} \\ t_2 \cdot\succ t_1 & \text{if } \mu_{t_1[j]} < \mu_{t_2[k]} \\ t_1 \doteq t_2 & \text{if } \mu_{t_1[j]} = \mu_{t_2[k]} \end{cases} \quad (9)$$

This procedure simply traverses the peerage from left to right collecting distinct avps and storing them in a ∇ list. Each avp can appear in the ∇ list only once. The resultant ∇ list defines an ordering among the distinct avps in the peerage. These avps are the components of a tuple in the database. For instance, consider that tuples t_1 and t_2 are composed of the sets of avps $\{x_1, a_1, A_1\}$ and $\{x_2, a_2, A_2\}$ respectively. Let the F expression be $F(x, F(a, A))$. Assume that $F(x_1, F(a_1, A_1)) = [[x_1; x_1]; [a_1; A_1]]$ and $F(x_2, F(a_2, A_2)) = [[x_2; x_2]; [a_2; A_2]]$. Then, $\nabla_{t_1} = x_1 \prec a_1 \prec A_1$ and $\nabla_{t_2} = x_2 \prec a_2 \prec A_2$. The relative preference (i.e., order) of t_1 and t_2 is decided by first comparing the degrees of membership for x_1 and x_2. If these memberships coincide, the following pair is compared (i.e., a_1 and a_2) and so on.

Definition 13 *The following notation is used throughout the last portion of this paper: $\sqcap \rightarrow$ a generic peerage; $\vartheta_\sqcap \rightarrow$ any one of the avps that compose $\sqcap$; $\mathcal{S}_\sqcap \rightarrow$ set of distinct avps that compose $\sqcap$; and $\nabla_\sqcap \rightarrow$ the ∇ list corresponding to $\sqcap$.*

It can be easily verified that $F(x, F(a, A)) \neq F(F(x, a), A)$ i.e., they generate distinct peerages. Thus, F is not associative. However, since both expressions define the same ordering for the avps that compose a tuple and since this ordering determines the rank of the tuple, we can say that the expressions are equivalent for ranking purposes. In the immediately following discussion we address this issue. Later, we show that the ranking produced by F and G can be computed such that the rankings are identical under associative transformations of expressions.

Definition 14 *We distinguish between an F expression and the ordering of tuples (i.e., ranking) it generates. We refer to the ordering of tuples generated by an F expression by F_o. Analogously, we refer to the ordering of tuples generated by a G expression by G_o.*

Definition 15 *An expression that contains only calls to the F function is called an* F-F *expression. An expression that contains only calls to the G function is called a* G-G *expression. Mixed expressions are called* F-G *expressions.*

In the remaining of this section, we generalize our approach for general (i.e., more complex) expressions. We first discuss F-F and G-G expressions. Afterwards, we consider F-G expressions.

4.2 F-F Expressions

We first establish the concept of an F-F peerage.

Definition 16 *Let $\sqcap$ be a peerage and $\nabla_{\sqcap} = \vartheta_1 \prec \vartheta_2 \prec \cdots \prec \vartheta_n$ be the corresponding ∇ list. If $\mu_{\vartheta_1} \leq \mu_{\vartheta_2} \leq \cdots \leq \mu_{\vartheta_n}$, we call $\sqcap$ an F-F peerage and represent it by the symbol $\sqcap^f$.*

Lemma 2 *Let x, y, z, and w be avps in the database. Expressions of the form $F(x,y)$, $F(x, F(y,z))$, $F(x, F(y, F(z,w)))$, and $F(F(x,y),\ F(z,w))$ are called* canonical F-F forms. *Evaluation of a canonical F-F form always generates a peerage of type F-F.*

We now extend the definition of F to deal with pairs of generic F-F expressions.

Definition 17 *Let $\mathcal{E}_1^f$ and $\mathcal{E}_2^f$ be two F-F expressions. Also, let $\sqcap_{\mathcal{E}_1^f}$ and $\sqcap_{\mathcal{E}_2^f}$ be the respective peerages. Then,*

$$F(\mathcal{E}_1^f, \mathcal{E}_2^f) = \left[\theta(\sqcap_{\mathcal{E}_1^f}, \sqcap_{\mathcal{E}_2^f}); \omega(\sqcap_{\mathcal{E}_1^f}, \sqcap_{\mathcal{E}_2^f})\right] \tag{10}$$

To generate the peerages corresponding to $\mathcal{E}_1^f$ and $\mathcal{E}_2^f$ simply apply this definition recursively.

At each recursive step, equation 10 generates simpler F-F expressions until a canonical F-F form is generated. This canonical form can then be trivially evaluated terminating the recursion cycle.

Lemma 3 *Evaluation of any F-F expression generates a peerage of type F-F.*

This is the key property of an F-F expression, it always returns the avps ordered by increasing grades of membership. Given this property, the following two lemmas follow naturally.

Lemma 4 *Regarding the ordering of tuples they generate, F-F expressions are commutative i.e.,*

$$F_o(\mathcal{E}_1^f, \mathcal{E}_2^f) = F_o(\mathcal{E}_2^f, \mathcal{E}_1^f) \tag{11}$$

Lemma 5 *Regarding the ordering of tuples they generate, F-F expressions are associative i.e.,*

$$F_o(\mathcal{E}_x^f, F(\mathcal{E}_1^f, \mathcal{E}_2^f)) = F_o(F(\mathcal{E}_x^f, \mathcal{E}_1^f), \mathcal{E}_2^f) \tag{12}$$

Commutativity and associativity are important because of the flexibility they provide for expressing the query intention. Further, they are intuitevely appealing from the user point of view.

4.3 G-G Expressions

The definitions for G-G expressions are analogous to those for F-F expressions. The key distinction is that G-G expressions give preference to avps with larger grades of membership while F-F expressions give preference to avps with smaller grades of membership.

Definition 18 *Let $\sqcap$ be a peerage and $\nabla_{\sqcap} = \vartheta_1 \prec \vartheta_2 \prec \cdots \prec \vartheta_n$ be the corresponding ∇ list. If $\mu_{\vartheta_1} \geq \mu_{\vartheta_2} \geq \cdots \geq \mu_{\vartheta_n}$, we call $\sqcap$ a G-G peerage and represent it by the symbol $\sqcap^g$.*

Lemma 6 *Let x, y, z, and w be avps in the database. Expressions of the form $G(x,y)$, $G(x, G(y,z))$, $G(x, G(y, G(z,w)))$, and $G(G(x,y), G(z,w))$ are called* canonical G-G forms. *Evaluation of a canonical G-G form always generates a peerage of type G-G.*

Definition 19 *Let $\mathcal{E}_1^g$ and $\mathcal{E}_2^g$ be two G-G expressions. Also, let $\sqcap_{\mathcal{E}_1^g}$ and $\sqcap_{\mathcal{E}_2^g}$ be the respective peerages. Then,*

$$G(\mathcal{E}_1^g, \mathcal{E}_2^g) = \left[\omega(\sqcap_{\mathcal{E}_1^g}, \sqcap_{\mathcal{E}_2^g}); \theta(\sqcap_{\mathcal{E}_1^g}, \sqcap_{\mathcal{E}_2^g})\right] \tag{13}$$

To generate the peerages corresponding to $\mathcal{E}_1^g$ and $\mathcal{E}_2^g$ simply apply this definition recursively.

At each recursive step, equation 10 generates simpler G-G expressions until a canonical G-G form is generated. This canonical form can then be trivially evaluated terminating the recursion cycle.

Lemma 7 *Evaluation of any G-G expression generates a peerage of type G-G.*

Thus, G-G expressions always retrieve the avps ordered by decreasing grades of membership. The next two lemmas follow immediately.

Lemma 8 *Regarding the ordering of tuples they generate, G-G expressions are commutative i.e.,*

$$G_o(\mathcal{E}_1^g, \mathcal{E}_2^g) = G_o(\mathcal{E}_2^g, \mathcal{E}_1^g) \tag{14}$$

Lemma 9 *Regarding the ordering of tuples they generate, G-G expressions are associative i.e.,*

$$G_o(\mathcal{E}_x^g, G(\mathcal{E}_1^g, \mathcal{E}_2^g)) = G_o(G(\mathcal{E}_x^g, \mathcal{E}_1^g), \mathcal{E}_2^g) \tag{15}$$

4.4 F-G Expressions

Expressions that mix calls to the F and G functions cannot be treated as F-F and G-G expressions. An F-F or G-G expression mingles avps with the intention to establish a final peerage that is of type F-F or G-G, respectively. An F-G expression, however, has to generate a peerage whose type is neither F-F nor G-G. Otherwise, it would be impossible to distinguish pure conjunctive or pure disjunctive expressions from mixed expressions.

We first provide basic definitions for generic expressions.

Definition 20 *As before, F-F and G-G expressions are referred to by the symbols $\mathcal{E}^f$ and $\mathcal{E}^g$ respectively. An F-G expression is referred to by $\mathcal{E}^{fg}$. An expression that is not of type F-F is referred to by $\mathcal{E}^{\overline{f}}$. An expression that is not of type G-G is referred to by $\mathcal{E}^{\overline{g}}$. A generic expression of any type is referred to by $\mathcal{E}$.*

We are now ready to generalize F and apply it to expressions that are not of type F-F.

Definition 21 *Let $\mathcal{E}_1^{\overline{f}}$ and $\sqcap_{\mathcal{E}_1^{\overline{f}}}$ be a non-F-F expression and its peerage respectively. Also, let $\mathcal{E}_2$ and $\sqcap_{\mathcal{E}_2}$ be a generic expression and its peerage respectively. Then,*

$$F(\mathcal{E}_1^{\overline{f}}, \mathcal{E}_2) = \begin{cases} \left[\sqcap_{\mathcal{E}_1^{\overline{f}}}; \sqcap_{\mathcal{E}_2}\right] & \textit{if } \nabla_{\sqcap_{\mathcal{E}_1^{\overline{f}}}} \leq \nabla_{\sqcap_{\mathcal{E}_2}} \\ \left[\sqcap_{\mathcal{E}_2}; \sqcap_{\mathcal{E}_1^{\overline{f}}}\right] & \textit{if } \nabla_{\sqcap_{\mathcal{E}_1^{\overline{f}}}} > \nabla_{\sqcap_{\mathcal{E}_2}} \end{cases} \tag{16}$$

The motivation is as follows. We cannot apply the conventional definition for F because it would attempt to sort the avps by increasing grades of membership for an expression that is not of type F-F. The resultant semantics would be difficult to define, difficult to predict, and likely to be dependent on the argument peerages. We find it more natural to avoid mixing the avps of peerages of different types and treat these peerages as monolithic blocks. Thus, if one of the argument expressions is not of type F-F, the F function treats its arguments as if they were single avps.

Definition 22 *Let $\mathcal{E}_1^{\overline{g}}$ and $\sqcap_{\mathcal{E}_1^{\overline{g}}}$ be a non-G-G expression and its peerage respectively. Also, let $\mathcal{E}_2$ and $\sqcap_{\mathcal{E}_2}$ be a generic expression and its peerage respectively. Then,*

$$G(\mathcal{E}_1^{\overline{g}}, \mathcal{E}_2) = \begin{cases} \left[\sqcap_{\mathcal{E}_1^{\overline{g}}}; \sqcap_{\mathcal{E}_2}\right] & \textit{if } \nabla_{\sqcap_{\mathcal{E}_1^{\overline{g}}}} \geq \nabla_{\sqcap_{\mathcal{E}_2}} \\ \left[\sqcap_{\mathcal{E}_2}; \sqcap_{\mathcal{E}_1^{\overline{g}}}\right] & \textit{if } \nabla_{\sqcap_{\mathcal{E}_1^{\overline{g}}}} < \nabla_{\sqcap_{\mathcal{E}_2}} \end{cases} \tag{17}$$

which is analogous to the definition for function F. Of course, G gives preference to the greater peerage.

Lemma 10 *With respect to the ordering of tuples they generate, functions F and G are distributive over F-G expressions.*

$$F_o(\mathcal{E}_x, G(\mathcal{E}_1^{\overline{g}}, \mathcal{E}_2) \quad = \quad G_o(F(\mathcal{E}_x, \mathcal{E}_1^{\overline{g}}), F(\mathcal{E}_x, \mathcal{E}_2)) \tag{18}$$

$$G_o(\mathcal{E}_x, F(\mathcal{E}_1^{\overline{f}}, \mathcal{E}_2) \quad = \quad F_o(G(\mathcal{E}_x, \mathcal{E}_1^{\overline{f}}), G(\mathcal{E}_x, \mathcal{E}_2)) \tag{19}$$

Notice that F can be distributed over G as long as G does not generate a G-G peerage. Analogously, G can be distributed over F as long as F does not generate an F-F peerage. Distribution is permitted as long as the internal peerage is of mixed type. Peerages of type F-F or G-G are treated as monolithic blocks which cannot be disturbed by external operations such as distribution. The following theorem provides additional insight into the issue of distributivity in general.

Lemma 11 *With respect to the ordering of tuples it generates, function F can* not *be distributed over G-G expressions. Further, function G can* not *be distributed over F-F expressions.*

F-F and G-G expressions order the set of avps by increasing and decreasing grades of membership, respectively. This property is important for two reasons. First, it is intuitevely appealing. It naturally extends the min/max operators to allow multiple criteria decisions. Second, it guarantees that the F and G functions are associative which makes them more attractive. From the user point of view, it is quite expected that the expressions $(A \wedge (B \wedge C))$ and $((A \wedge B) \wedge C)$ have the same meaning. However, associativity prevents distributivity of functions F and G in general. F-F and G-G subexpressions have to be treated as monolithic blocks when computing F-G expressions. Functions F and G can be distributed over F-G expressions though.

4.5 A Small Example

Consider the query q and tuples t_1 and t_2 illustrated in figure 2. As demonstrated in section 2, conventional fuzzy theory improperly gives tuples t_1 and t_2 the same rank relative to query q. For purpose of comparision, let us compute these ranks using F-G algebra. For simplicity, we substitute avps for the corresponding grades of membership inside peerages.

The rank for tuple t_1 is given by $F(G(\mu_{\Upsilon_{bh},t_1}, \mu_{\Upsilon_{dt},t_1}), \mu_{\Upsilon_{adv},t_1}) = [0.45; [1; 0.6]]$. Analogously, the rank for tuple t_2 is given by $F(G(\mu_{\Upsilon_{bh},t_2}, \mu_{\Upsilon_{dt},t_2}), \mu_{\Upsilon_{adv},t_2}) = [0.45; [0.72; 0.5]]$. Final preferences are decided by comparing the two peerages. As one should expect, $t_1 \cdot \succ t_2$ despite the fact that the first order grades of membership coincide.

5 Conclusions

We proposed an extended fuzzy algebra called *F-G algebra* for ranking approximate answers relative to a multiple criteria query request. Instead of combining the multiple grades of membership into a single number, the F-G algebra creates a list according to the query expression. These lists are then compared to decide the ranking of the corresponding tuples. Generation of these lists is based on two basic principles: (a) the similarity measure of an *and* operation should *not* exceed the similarity measure of any of its operands and (b) the similarity measure of an *or* operation should match or exceed the similarity measure of any of its operands. These principles are simple and easy to grasp from the user's point of view and their use is less likely to yield in non intuitive results.

The main contributions of our research are four. First, the F-G algebra is simple and yet general enough to solve the problem of deciding preferences in the presence of multiple approximate criteria. Second, the algebra is derived from two basic principles that can be easily grasped by the user. Third, the algebra preserves desirable properties such as commutativity, associativity, and a restricted form of distributivity. Fourth, the algebra allows the user to state his preferences in qualitative terms and have the tuples ordered according to these preferences. This last point is important for two reasons: (a) the user might find it difficult to express his preferences in numeric terms and (b) the ranking generated by the F-G algebra is inherently qualitative which, we believe, simplifies its understanding by the user.

References

[1] R. Bellman and M. Giertz. On the analytic formalism of the theory of fuzzy sets. *Inform. Sci.*, 5:149–156, 1973.

[2] F. Cuppens and R. Demolombe. Cooperative answering: a methodology to provide intelligent access to databases. In *Second International Conference on Expert Database Systems*, Virginia, U.S.A., 1988.

[3] Norbert Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of 16th VLDB Conference*, Brisbane, Australia, 1990.

[4] H. Hamacher. On logical connectives of fuzzy statements and their affiliated truth function. In *Proc. Third European Meeting Cybernetics and Systems Res.*, Vienna, Austria, 1976.

[5] Amihai Motro. Vague: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, July 1988.

[6] F. Rabitti and P. Savino. An information retrieval approach for image databases. In *Proceedings of 18th VLDB Conference*, Vancouver, British Columbia, Canada, 1992.

[7] Berthier A. Ribeiro. Approximate answers in intelligent systems, 1994. Ph.D. dissertation. In preparation.

[8] S.K.M. Wong and Pawan Lingras. Representation of qualitative user preference by quantitative belief functions. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):72–78, February 1994.

[9] L.A. Zadeh. Fuzzy sets. In D. Dubois, H. Prade, and R.R. Yager, editors, *Readings in Fuzzy Sets for Intelligent Systems*. Morgan Kaufmann, 1993.