International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 10, Issue 2, 2024, PP 1-21
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
https://doi.org/10.20431/2349-4859.1002001
www.arcjournals.org

REALM: The Retrieval Algebra Model and its Implementation

Nektarios Moumoutzis, George Anestis, Stavros Christodoulakis

Laboratory of Distributed Multimedia Information Systems and Applications, School of Electrical and Computer Engineering, Technical University of Crete – TUC/MUSIC, 73100, Chania, Greece

Corresponding Author:** ***Nektarios Moumoutzis, *Laboratory of Distributed Multimedia Information Systems and Applications, School of Electrical and Computer Engineering, Technical University of Crete – TUC/MUSIC, 73100, Chania, Greece*

Abstract: *We propose a model based on a mathematical framework for the integration of database and information retrieval models on top of the traditional relational model. We do not require any extensions of the relational model. Our model supports easily and uniformly naive user languages for accessing information using traditional boolean and several fuzzy retrieval languages. In this model both queries and objects are represented as (base) relations and the evaluation of queries is equivalent to the computation of algebraic expressions of the base relations. The implementation of the model on top of a traditional RDBMS supports the most popular information retrieval models of the literature.*

1. Introduction

A considerable research effort is ongoing recently with the objective to integrate the two basic technologies dealing with information management i.e. Database Management Systems (DBMS) and Information Retrieval Systems (IRS). These two technologies have been separated for years. DBMS focus on data retrieval based on boolean queries whereas in IRS the queries are phrased in terms of features or similarity functions and the retrieved objects are ranked based on their similarity to the query. It is evident that an effective integration of these two technologies could lead to the development of very powerful systems. The major research efforts in order to integrate these two technologies are the following:

- Extensions of existing DBMS with the integration of text searching operators.
- Extensions of the database models to accommodate incomplete or probabilistic information.
- Integration of IRS and DBMS systems.
- Extensions of IRS systems to accommodate relational attributes.

Although a number of research efforts in these directions seem to be promising or have already led to industrial products, it remains an open issue the real integration of these two technologies in terms of user functionality. Such integration should address the following needs:

1. The need to support the formulation and evaluation of complex similarity queries expressed with the logical operators AND, OR, NOT
2. The need to use the same attributes - possibly transformed – in both boolean and similarity queries.
3. The need to use different models for evaluating similarity queries. Experiments have shown that there is no universal evaluation model with satisfactory results in every application domain.
4. The need to combine boolean and similarity queries in one retrieval request. In this case, the boolean part of the request acts as a filter that cuts some objects out, whereas the similarity part of the request is used to rank the remaining objects.

In this work a model that addresses these needs is proposed and its implementation on top of a relational DBMS is described. The objective of this work is to propose an implementation of IRS functionality on top of SQL, based on well understood principles. Moreover to provide an easy way for naive users to experiment with diverse and alternative retrieval models without writing complex SQL queries. The approach is based on the *REtrieval ALgebra Model* (REALM) which is capble of describing IRS's based on different fuzzy models. Algebraic expressions in RELAM represent the evaluation of similarity queries. REALM is implemented on top of SQL providing the ability to extend existing relational databases with IRS functionality very easily.

REALM is based on the mathematical notion of relation as a subset of the Cartesian Product of two arbitrary sets and the mathematical operation of composition between relations. Starting from these well-known mathematical concepts, we prove that we can compute the answer to boolean queries in a primitive IRS, which is using just boolean logic, by an algebraic expression in REALM. The key idea here is to represent the query itself as a relation or as two relations in case of a query in normal form (either conjunctive or disjunctive).

Then we proceed to the generalization of the concept of relation to support fuzzy truth-values. We call *ranking relation* such a generalized relation. We generalize the concept of composition of two relations and develop a generalized composition operation between ranking relations. In a similar manner, we express similarity queries as ranking relations and we prove that the evaluation of such queries corresponds to the computation of algebraic expressions between ranking relations. However in order to do so, a pair of generalized compositions between ranking relations has to be specified for each evaluation model, one for evaluating OR and one for evaluating AND expressions. We give these pairs for the most popular evaluation models.

The result of this work is an algebraic framework that incorporates boolean and similarity data and retrieval queries within the data themselves. This framework leads to an implementation environment on top of standard relational systems, which supports many different retrieval models with standard interfaces. Complex retrieval of different kinds does not require SQL extensions or complex SQL expressions but only a set of insertions in fixed tables, which can be supported by simple interfaces. This raises the level of interaction of naive users with the IR engines.

1.1. Running Example

We present now an example of a relational database that will be used in the rest of the paper to clarify the concepts and the techniques presented. The example is taken from the database used in the Campiello[1] project to describe hotels in a tourist information system. Three tables are used (primary key attributes are shown in bold face): Hotels (**hotel ID**, name), Facilities (**facility ID**, name), and Hotel Facilities (**hotel ID**, **facility ID**, importance).

Hotels

hotelID	name
h1	Samaria
h2	Omalos
h3	Panorama
h4	Kydon

Facilties

facilityID	name
f1	Air Condition
f2	Swimming Pool
f3	Video
f4	Color TV

HotelFacilities

hotelID	facilityID	importance
h1	f3	0.5
h1	f4	0.3
h2	f1	0.7
h2	f3	0.8
h2	f4	0.9
h3	f2	0.7
h3	f4	0.4
h4	f4	0.6
h4	f3	0.2

Figure1. *An instance from the Hotels – Facilities database*

The table Hotels represents hotels in the area of Chania. Each hotel may be associated with a number of different facilities representing services offered by the hotel (e.g. swimming pool, tennis court etc.). Table Facilities stores the different facility types. The association between hotels and facilities is stored in the Hotel Facilities table. A tuple $<h,f,w>$ in this table represents the fact that hotel h is associated with facility f with importance $w \in [0,1]$. We adopt the convention that associations between hotels and facilities with zero importance are not stored in the Hotels Facilities table and inversely any

[1] Campiello project aims to experiment the use of innovative technologies to develop new links between the local communities and visitors of the Historical Cities of Art and Culture. Two cities have been chosen as context of the experiment: Venezia in Italy and Chania in Greece.

<hotel, facility> pair not present in the table represents an association of zero importance (this is similar to the closed world assumption in DBMS's). A sample database is given in Figure1. In the following we give four example similarity queries. The fuzzy model used is Fuzzy Set (see [JOON94]) that operator MAX for the evaluation of logical disjunction and operator MIN for the evaluation of logical conjunction.

Example 1

Let us assume that we would like to answer the following query: *"Find the hotels with swimming pool and rank them by the importance of these facilities"*. The SQL query that finds these hotels sorted by their importance is the following:

```
Q1:   SELECT hotelID,importance
      FROM HotelFacilities
      WHERE facilityID='f2'
      ORDER BY importance
```

Example 2

Let us consider now a more complicated query: *"Find the hotels with swimming pool or air condition and rank them by the importance of these facilities"*. This is a disjunctive similarity query and using the MAX operator, the SQL query that finds these hotels, sorted by their importance, is:

```
Q2:   SELECT hotelID,MAX(importance)
      FROM HotelFacilities
      WHERE facilityID='f2' or facilityID='f1'
      GROUP BY hotelID
      ORDER BY importance
```

Example 3

Let us consider now a more interesting query that may be issued by a potential tourist who tries to find a hotel for his vacations: *"Find the hotels with air condition and color TV and rank them by the importance of these facilities"*. This is a conjunctive query and the corresponding SQL expression using the MIN operator is:

```
Q3:   SELECT hotelID,MIN(importance)
      FROM HotelFacilities
      WHERE facilityID='f1' or facilityID='f4'
      GROUP BY hotelID
      ORDER BY importance
```

Unfortunately this query is not correct. Consider hotel 'Panorama' (see Figure) that has a color TV with importance 0.4 but it does not have air condition facility. The correct rank for 'Panorama' with respect to this query is MIN{0.4,0)=0. However, as already stated, in table Hotel Facilities, there is no reference for hotels ranked 0 with respect to any facility. As a consequence, query Q3 will erroneously compute 0.4 as the aggregate rank for 'Panorama'. The solution to this problem is either to have tuples in the database also for zero values or modify the query to take into account that non-existing tuples count for zero values. The first solution degrades the performance of the database by adding a big number of tuples. The second solution is more elegant. To implement it we should count the facilities associated with each hotel and demand that they are exactly 2 (i.e. the same number as the facilities present in the query). The correct SQL query is the following:

```
Q4:   SELECT hotelID,MIN(importance)
      FROM HotelFacilities
      WHERE facilityID='f1' or facilityID='f4'
      GROUP BY hotelID
      ORDER BY importance
      HAVING Count(faciltyID)=2
```

Example 4

The situation may become more complicated if we want to ask more complex questions like *"Find the hotels with (air condition or swimming pool) and color TV and rank them by the importance of these facilities"*. This query resembles a complex boolean query. However the aggregate rank for an arbitrary hotel is different from a traditional boolean query. What is the corresponding SQL query in this case? The query to find hotels with air condition or swimming pool is the query Q2 above. The query to find the hotels with color TV is the following:

```
Q5:   SELECT hotelID,importance
      FROM HotelFacilities
      WHERE facilityID='f4'
```

The final SQL query is a combination of Q2 and Q5:

```
Q6:   SELECT hotelID,MIN(importance)
      FROM (Q2 UNION Q5)
      GROUP BY hotelID
      HAVING Count(*)=2
```

Although this example query is quite simple in similarity based retrieval environment, it is rather complicated in SQL. Taking into account that tt may be desirable to support more than one evaluation model and that the terms of the query could be arbitrarily complex, we conclude that it is apparently inefficient to develop such queries by hand and ad hoc. We should develop a consistent methodology and translation process and provide tools to make such translations. This is exactly the objective of this work. Not only it provides such a methodology, but also proposes an alternative way of representing similarity queries in relational tables. This is the key idea of the model. Let us see how this idea can help to express the queries we have just developed.

1.2. Representing Similarity Queries in Relational Tables

Consider the class of queries $\underset{i=1}{\overset{n}{AND}}\left(\underset{j=1}{\overset{n_i}{OR}} t_{k_j}\right)$ in conjunctive normal form without negation with atoms taken from facilities. The corresponding queries – using facilityID's to denote facilities used in each query – for the examples given above are: [e1]:(f2), [e2]:(f1)OR(f2), [e3]:(f1)AND(f4), and [e4]:((f1)OR(f2))AND(f4). In order to represent these queries in relational tables we use one table (named OR Table) to store terms $\underset{j=1}{\overset{n_i}{OR}} t_{k_j}$ of all queries and another table (named AND Table) to store the association of such terms and queries. These tables are shown in Figure.

ANDTable

queryID	ORtermID
e1	t1
e2	t2
e3	t3
e4	t2
e4	t4

ORTable

ORtermID	facilityID
t1	f1
t2	f1
t2	f2
t3	f1
t3	f4
t4	f4

Figure2. *Tables storing conjunctive normal form queries without negation, on facility types*

Using the tables Hotel Facilities and OR Table we may compute the aggregate rank of each hotel to each one of the terms appearing in the queries with the following SQL statement:

```
Qa:   SELECT termID, hotelID, MAX(importance)
      FROM HotelFacilities, ORTable
      WHERE HotelFacilities.facilityID=ORTable.facilityID
      GROUP BY ORtermID, hotelID
```

Finally, we may compute the similarity between any <query,hotel> pair using Qa and AND Table by:

```
Qb:   SELECT queryID, hotelID, MIN(importance)
      FROM Qa A, ANDTable B
      WHERE A.ORtermID=B.ORtermID
      GROUP BY queryID, hotelID
      HAVING Count (ORtermID)= (SELECT Count(*) FROM ANDTable C WHERE B.query ID=C.
      query ID  )
      ORDER BY MIN (importance)
```

Although this process may seem more complicated than constructing queries Q1, Q2, Q3 and Q5, the truth is that it is more general and suitable for automating the task of answering similarity queries for the database. Queries Qa and Qb can be used, without modification, to answer any query in conjunctive normal form without negation and not just the specific queries we have been considering.

We just have to insert the appropriate tuples in AND Table and OR Table. To accommodate different evaluation models only a slight modification to queries Qa and Qb is required.

In the rest of the paper we present the framework for this translation process. This paper is organized as follows. In section 2 we introduce a general model for similarity queries. In section 3 we elaborate the algebraic model for the evaluation of boolean and similarity queries based on the notion of ranking relations. In section 4 we show how ranking relations can be implemented in a relational DBMS and how the algebraic expressions involving ranking relations are translated into SQL queries. In section 5 we present related work. The paper concludes with section 6, which also describes future directions in the research work

2. A General Model for Similarity Queries

For the description of IRS we adopt an appropriate version of the framework presented in [JOON94]. Similar formalisms can be found in [FAGI96]. We give three definition of an IRS starting from the simple case of a Boolean IRS, going to the Fuzzy IRS and arriving at the most general form of IRS, the Weighted Fuzzy IRS.

Definition 1: Boolean IRS

A Boolean IRS $\mathtt{A}$ is a quadruple $\mathtt{A}=\langle P,E,Q,F\rangle$ where:

P is the set of index terms used to describe objects and formulate queries.

E is a set of objects called *indexed objects*. Each such object $e\in E$ is considered as a function[2] $e:P\rightarrow\{0,1\}$. Consequently, the set *E* is a subset of $\{0,1\}^P$.

Q is a set of queries that are identified by the system. These are expressions built from the alphabet $P\cup\{AND, OR, NOT\}$ and usually are defined recursively.

F is an evaluation function $F:Q\times E\rightarrow\{0,1\}$ that gives a value from {0,1} to any valid query from *Q*. This function is defined recursively based on the truth tables of logical operators:

$$F(q_1ANDq_2,e)=f_{AND}(F(q_1,e),F(q_2,e))=MIN\{F(q_1,e),F(q_2,e)\}$$

$$F(q_1ORq_2,e)=f_{OR}(F(q_1,e),F(q_2,e))=MAX\{F(q_1,e),F(q_2,e)\}$$

$$F(NOTq_1,e)=f_{NOT}(F(q_1,e))=1-F(q_1,e)$$

$$F(p,e)=e(p)$$

For a query $q\in Q$, the subset *ANS(q)* of *E* containing all elements of *E* for which the evaluation function *F* is 1, is the answer to *q*. This subset is defined as: $ANS(q)=\{e\in E|F(q,e)=1\}$

Definition 2: Fuzzy IRS

A Fuzzy IRS $\mathtt{B}$ is a quadruple $\mathtt{B}=\langle P,E,Q,F\rangle$ where:

P is the set of index terms used to describe objects and formulate queries.

E is a set of objects called *indexed objects*. Each such object $e\in E$ is considered as a function[3] $e:P\rightarrow[0,1]$. Consequently, the set *E* is a subset of $[0,1]^P$.

Q is a set of queries that are identified by the system. These are expressions built from the alphabet $P\cup\{AND, OR, NOT\}$ and usually it is defined recursively.

F is an evaluation function $F:Q\times E\rightarrow[0,1]$ that gives a value from [0,1] to any valid query from *Q*. This function is defined recursively based on the truth tables of the logical operators:

$$F(q_1ANDq_2,e)=f_{AND}(F(q_1,e),F(q_2,e))$$

[2] Essentially the indexed object *e* is associated to any index term *p* for which $e(p)=1$.
[3] Essentially the indexed object *e* is associated to any index term *p* for which $e(p)>0$.

$$F(q_1 OR q_2, e) = f_{OR}(F(q_1, e), F(q_2, e))$$

$$F(NOT q_1, e) = f_{NOT}(F(q_1, e))$$

$$F(p, e) = e(p)$$

For a query $q \in Q$, the subset *ANS(q)* of *E* containing all elements of *E* for which the evaluation function *F* is not zero, is the answer to *q*. This subset is defined as: $ANS(q) = \{e \in E | F(q,e) > 0\}$

It is evident that the above definition differs from the definition of a Boolean IRS in the set of possible values for the evaluation function, which is the closed interval [0,1] and not just the two-valued set {0,1}. Usually the answer set is ordered based on the value of the evaluation function for each $e \in ANS(q)$.

In *definition 2*, it is apparent that the functions that give the truth values for the logical operators play the most crucial role. In the case of the Boolean IRS, these functions are defined uniquely. However, in the case of a Fuzzy IRS, there are different possible groups of these functions, each group defines essentially a different fuzzy model. So, in order to define a fuzzy model, we need to specify the following three functions, called *basic evaluation functions* hereafter: The evaluation function of the logical conjunction $f_{AND} : [0,1] \times [0,1] \to [0,1]$, the logical disjunction $f_{OR} : [0,1] \times [0,1] \to [0,1]$ and the logical negation $f_{NOT} : [0,1] \to [0,1]$.

Some of the most popular fuzzy models and their corresponding basic evaluation functions are given in Table, which is adopted from [JOON94]. In the first column of the table, the initials of each model are given i.e. *FS*: Fuzzy Set, *WK*: Waller Kraft, *PN*: P-Norm, and *IO*: Infinite One.

Table1. *Basic evaluation function for the most popular fuzzy models*

	$f_{NOT}(a)$	$f_{AND}(a,b)$	$f_{OR}(a,b)$
FS	$1-a$	$MIN\{a,b\}$	$MAX\{a,b\}$
WK	$1-a$	$(1-\gamma)\cdot MIN\{a,b\}+\gamma\cdot MAX\{a,b\}$, $0\leq\gamma\leq 0.5$	$(1-\gamma)\cdot MIN\{a,b\}+\gamma\cdot MAX\{a,b\}$, $0.5\leq\gamma\leq 1$
PM	$1-a$	$1-((1-a)^p+(1-b)^p)^{1/p}/2$, $1\leq p\leq\infty$	$(a^p+b^p)^{1/p}/2$, $1\leq p\leq\infty$
IO	$1-a$	$\gamma\cdot(1-MAX\{1-a,1-b\})+(1-\gamma)\cdot(a+b)/2$,	$\gamma\cdot(MAX\{a,b\})+(1-\gamma)\cdot(a+b)/2$, $0\leq\gamma\leq 1$

From Table it can be seen that some of the fuzzy models presented, do not satisfy the usual properties of boolean conjunction, disjunction and negation. The most significant of these properties is associativity, but unfortunately from the above models only Fuzzy Set satisfies it. The significance of this property is that it allows for the computation of complex expressions without the need to specify a specific order in the execution of the sub-expressions. When this property does not hold, the order of evaluation is significant, meaning that different order in the evaluation of an expression gives different result so that the comparison of results is problematic. In order to bypass this problem n-ary evaluation functions are used. For simplicity, n-ary evaluation functions are also used for the models that do satisfy associativity. In Table2 the extensions of the basic evaluation function for the fuzzy models already presented are given.

Table2. *N-ary basic evaluation functions for some popular fuzzy models*

	$F_{NOT}(A)$	$F_{AND}(A_1,\ldots,A_N)$	$F_{OR}(A_1,\ldots,A_N)$
FS	$1-a$	$MIN\{a_1,\ldots,a_n\}$	$MAX\{a_1,\ldots,a_n\}$
WK	$1-a$	$(1-\gamma)\cdot MIN\{a_1,\ldots,a_n\}+\gamma\cdot MAX\{a_1,\ldots,a_n\}$, $0\leq\gamma\leq 0.5$	$(1-\gamma)\cdot MIN\{a_1,\ldots,a_n\}+\gamma\cdot MAX\{a_1,\ldots,a_n\}$, $0.5\leq\gamma\leq 1$
PN	$1-a$	$1-\left(\sum_{i=1}^{n}(1-a_i)^p \Big/ n\right)^{1/p}, 1\leq p\leq\infty$	$\left(\sum_{i=1}^{n}a_i^p \Big/ n\right)^{1/p}, 1\leq p\leq\infty$
IO	$1-a$	$\gamma\cdot(1-MAX\{1-a_1,\cdots,1-a_n\})+(1-\gamma)\cdot\left(\sum_{i=1}^{n}a_i \Big/ n\right),$ $0\leq\gamma\leq 1$	$\gamma\cdot MAX\{a_1,\cdots,a_n\}+(1-\gamma)\cdot\left(\sum_{i=1}^{n}a_i \Big/ n\right),$ $0\leq\gamma\leq 1$

All terms in queries accepted by a Fuzzy IRS are of equal importnce. However, in many cases it is necessary to give different importance (weight) to different terms. In order to satisfy this need a new kind of IRS is defined:

Definition 3: Weighted Fuzzy IRS

A Weighted Fuzzy IRS C is a quadruple $\mathrm{C} = \langle P, E, Q, F\rangle$ where:

P is the set of index terms used to describe objects and formulate queries.

E is a set of objects called *indexed objects*. Each such object $e \in E$ is considered as a function[4] $e: P \rightarrow [0,1]$. Consequently, the set E is a subset of $[0,1]^P$.

Q is a set of queries that are identified by the system. Subqueries of each query are associated with weights from [0,1].

F is an evaluation function $F: Q \times E \rightarrow [0,1]$ that gives a value from [0,1] to any valid query from Q. This function is defined recursively based on the truth tables of the logical operators:

$$F(\langle q_1, w_1\rangle AND \langle q_2, w_2\rangle, e) = f_{AND}(\langle F(q_1,e), w_1\rangle, \langle F(q_2,e), w_2\rangle)$$

$$F(\langle q_1, w_1\rangle OR \langle q_2, w_2\rangle, e) = f_{OR}(\langle F(q_1,e), w_1\rangle, \langle F(q_2,e), w_2\rangle)$$

$$F(NOT q_1, e) = f_{NOT}(F(q_1, e))$$

$$F(p,e) = e(p)$$

For a query $q \in Q$, the subset *ANS(q)* of E containing all the elements of E for which the evaluation function F has a non-zero value, is the answer to q. This subset is defined as:

$$ANS(q) = \{e \in E \mid F(q,e) > 0\}$$

It is evident that the above definition differs from the definition of a Fuzzy IRS in that sub-expression in a query, are associated with weights. For a particular query $q \in Q$, the set of indexed objects from E for which the evaluation function F is non-zero, constitute the answer set for this query. That is: $ANS(q) = \{e \in E \mid F(q,e) > 0\}$.

Table3. *Weighted n-ary basic evaluation functions for weighted fuzzy models*

	$f_{AND}((a_1,w_1),\ldots,(a_n,w_n))$	$f_{OR}((a_1,w_1),\ldots,(a_n,w_n))$
PN	$1-\left(\sum_{i=1}^{n}(1-a_i)^p \cdot w_i^p \Big/ \sum_{i=1}^{n} w_i^p\right)^{1/p}, 1 \le p \le \infty$	$\left(\sum_{i=1}^{n} a_i^p \cdot w_i^p \Big/ \sum_{i=1}^{n} w_i^p\right)^{1/p}, 1 \le p \le \infty$
IO	$\gamma \cdot \left(1-\frac{MAX\{(1-a_1)\cdot w_1,\cdots,(1-a_n)\cdot w_n\}}{MAX\{w_1,\cdots,w_n\}}\right) + (1-\gamma)\cdot\left(\sum_{i=1}^{n} a_i \cdot w_i \Big/ \sum_{i=1}^{n} w_i\right), \quad 0 \le \gamma \le 1$	$\gamma \cdot \frac{MAX\{a_1 \cdot w_1,\cdots,a_n \cdot w_n\}}{MAX\{w_1,\cdots,w_n\}} + (1-\gamma)\cdot\left(\sum_{i=1}^{n} a_i \cdot w_i \Big/ \sum_{i=1}^{n} w_i\right), \quad 0 \le \gamma \le 1$

In Table 3 we give the basic evaluation function for the Weighted Fuzzy models. *Note that Weighted Fuzzy IRS's are the most general IRS's. A Boolean IRS can be considered as a Fuzzy IRS in the Fuzzy Set model with index term weights 1. Moreover, a Fuzzy IRS in the Fuzzy Set or in the Waller-Kraft model, can be considered as a Weighted Fuzzy IRS with sub-expression weights 1.*

3. Algebraic Evaluation of Boolean and Similarity Queries

In this section we develop a theoretical framework with the aim to provide a uniform handling of similarity and boolean queries. The core of the model is the notion of ranking relation, which is an extension of the traditional boolean relation. Starting from a Boolean IRS, we prove that the

[4] Essentially the indexed object e is associated to any index term p for which $e(p)>0$.

evaluation of a set of queries can be stated by algebraic expressions on boolean relations: One relation stands for the indexed objects and some others stand for the queries. Extending the notion of boolean relation to the notion of ranking relation, we show how to express similarity queries in a fuzzy IRS (or a weighted fuzzy IRS) by ranking relations. We then prove that the evaluation of a set of queries in such a system can be stated again as an algebraic expression between ranking relations.

3.1. Boolean Relations and Query Evaluation in a Boolean IRS

A boolean relation r between two sets V and W, is a subset of the Cartesian product $V\times W$, i.e. $r\subseteq V\times W$. It is equivalent to define r using its characteristic function f_r which is:

$$f_r : V\times W \to \{0,1\},\quad f_r(v,w)=\begin{cases}1, & \langle v,w\rangle \in r\\ 0, & \langle v,w\rangle \notin r\end{cases}$$

Inversely, if the characteristic function f_r is given, the relation r can be reconstructed:

$$r = f_r^{-1}[\{1\}] = \{\langle v,w\rangle \in V\times W|\ f_r(v,w)=1\}$$

Consequently, the set of relations between sets V and W, is the set of functions from the Cartesian product $V\times W$ to the set $\{0,1\}$. We use the symbol $\{0,1\}^{V\times W}$ for this set. In the following we use the characteristic function as the relation itself. In summary, a boolean relation is defined as follows:

Definition 3: Boolean Relation

A boolean relation r between the sets V and W is defined as a function from the Cartesian product $V\times W$ to the set $\{0,1\}$. In symbols: $r:V\times W\to\{0,1\}$

3.1.1. Basic Operations in Boolean Relations

The basic operations between boolean relations that are used here are the usual set-theoretic operations. The difference is that we use the characteristic functions. For union, intersection, and complement the definitions are straightforward (see [SCST93]):

Union: $(r\cup s)(v,w)=MAX\{r(v,w),s(v,w)\}$

Intersection: $(r\cup s)(v,w)=MIN\{r(v,w),s(v,w)\}$

Complement: $\bar{r}(v,w)=1-r(v,w)$

We need also to define the *composition* of relations. Given two boolean relations $r\subseteq V\times W$ and $s\subseteq W\times Z$, their composition is defined as follows:

$$r \circ s = \{\langle v,z\rangle \in V\times Z|\ \exists w\in W : \langle v,w\rangle \in r AND \langle w,z\rangle \in s\}$$

This definition can be stated equivalently as [see SCST93]:

$$r \circ s = \left\{\langle v,z\rangle \in V\times Z \middle| \underset{w\in W}{OR}(\langle v,w\rangle \in r AND \langle w,z\rangle \in s)\right\}$$

Although these two definitions are equivalent, the second one is more flexible as it allows us to re-define the composition using characteristic functions. Assuming the existence of proper evaluation functions for the logical disjunction and conjunction we may write:

$$(r\circ s)(v,z) = \underset{w\in W}{OR}(r(v,w) AND s(w,z))$$

It is trivial to show that $p\ OR\ q = MAX\{p,q\}$ and $p\ AND\ q = MIN\{p,q\}$. So, the last definition for composition can be re-written using characteristic functions to give the final definition for the composition of two boolean relations:

Definition 4: Composition of Boolean Relations

The composition between two boolean relations $r \subseteq V \times W$ and $s \subseteq W \times Z$ is defined as follows:

$$(r \circ s)(v,z) = \underset{w \in W}{MAX} \{MIN\{r(v,w), s(w,z)\}\}$$

3.1.2. Using Boolean Relations to Evaluate Boolean Queries

Let $\mathbb{A} = \langle P, E, Q, F \rangle$ be a Boolean IRS as defined in the previous section. We are now ready to demonstrate how the evaluation function F can be computed algebraically using boolean relations. We start from simple queries first and then we go to more complex ones. Before starting, note that the relationship between index terms from P and indexed objects from E can be stated as a boolean relation $r{:}P \times E \rightarrow \{0,1\}$ with $r(p,e)=e(p)$.

Evaluation of simple one term queries

Let us assume first that the queries recognized by the system are trivial ones of the form $q=p$. In this case, the set of valid queries is exactly the set P of index terms. The evaluation function $F{:}P \times E \rightarrow \{0,1\}$ is $F(p,e)=e(p)$. However, the relation r defined above is exactly the same. So, if the queries recognized by the system are trivial queries of the form $q=p$, then the evaluation function is given by the following equation:

F= r (1)

Evaluation of simple one term negation queries

Let us assume now that the queries recognized by the system are of the form $q=NOTp$. Using the symbol P_{NOT} for the set $\{NOTp \mid p \in P\}$ of valid queries, the evaluation function $F{:}P_{NOT} \times E \rightarrow \{0,1\}$ is $F(NOTp,e)=1-e(p)=1-r(p,e)=\bar{r}(p,e)$. That is, in this case:

$$F = \bar{r} \qquad (2)$$

Evaluation of simple disjunctive queries

The queries next to be studied are simple disjunctive queries of the form $q = \underset{j=1}{\overset{n}{OR}}\, p_j$. Any such query can be considered as the subset $\{p_1, p_2, \ldots, p_n\} \subseteq P$. Consequently, the set Q of valid queries in this case is essentially a subset of the powerset of P, i.e. $Q \subseteq \wp(P)$. We can define now a relation s that represents the valid simple disjunctive queries as follows:

$$s{:}Q \times P \rightarrow \{0,1\} \text{ with } s(q,p) = \begin{cases} 1, & p \in q \\ 0, & p \notin q \end{cases}$$

The following theorem constitutes the basis of our methodology as it expresses the fact that the evaluation function F can be computed algebraically from the boolean relations r and s.

Theorem 1:

Let $\mathbb{A} = \langle P, E, Q, F \rangle$ be a Boolean IRS. Assuming that it accepts only simple disjunctive queries of the form $q = \underset{j=1}{\overset{n}{OR}}\, p_j$, we define boolean relations r and s as above. Then, the evaluation function F of $\mathbb{A}$ is given by the following expression:

$$F = s \circ r$$

Proof

Let $\langle q, e \rangle \in Q \times E$. We are going to show that:

$$F(q,e) = (s \circ r)(q,e) \qquad (3)$$

Taking the right hand side of this equation, we have (see *definition 4*):

$$(s \circ r)(q,e) = \underset{p \in P}{OR}\left(s(q,p) AND r(p,e)\right) = \underset{p \in P}{MAX}\{MIN\{s(q,p), r(p,e)\}\}$$

and by re-ordering the elements:

$$(s \circ r)(q,e) = MAX\left\{\underset{p \in q}{MAX}\{MIN\{s(q,p), r(p,e)\}\}, \underset{p \in (P-q)}{MAX}\{MIN\{s(q,p), r(p,e)\}\}\right\} \tag{4}$$

Moreover, for $p \notin q$ it is $s(q,p) = 0$, so $MIN\{s(q,p), r(p,e)\} = MIN\{0, r(p,e)\} = 0$ for each $p \in (P-q)$. Consequently, $\underset{p \in (P-q)}{MAX}\{MIN\{s(q,p), r(p,e)\}\} = 0$. Substituting this result in (4), we take:

$$(s \circ r)(q,e) = \underset{p \in q}{MAX}\{MIN\{s(q,p), r(p,e)\}\}$$

However, for $p \in q$ it is $s(q,p) = 1$. So the last equation gives:

$$(s \circ r)(q,e) = \underset{p \in q}{MAX}\{r(p,e)\} \tag{5}$$

Taking now the left-hand side of (3) we have by *definition 1* and equation (1):

$$F(q,e) = \underset{p \in q}{MAX}\left(F(p,e)\right) = \underset{p \in q}{MAX}\left(r(p,e)\right) \tag{6}$$

Combining (5) and (6) we conclude that equation (3) holds. □

Evaluation of simple disjunctive queries with negation

In case the queries that are recognized by the system are allowed to have negations, we have the form of simple disjunctive queries with negation: $q = \underset{j=1}{\overset{n_q}{OR}}\, \dot{p}_{k_{q,j}}$, with $\dot{p}_{k_{q,j}} \in \dot{P} = P \cup P_{NOT}$. Now, the set of valid queries can be considered as a subset of the powerset of $\dot{P}$, i.e. $Q \subseteq \wp(\dot{P})$. The boolean relation that represents Q is the following:

$$\dot{s} : Q \times \dot{P} \to \{0,1\}, \text{ with } \dot{s}(q,p) = \begin{cases} 1, & \dot{p} \in q \\ 0, & \dot{p} \notin q \end{cases}$$

Due to the fact that $Q \times \dot{P} = Q \times (P \cup P_{NOT}) = (Q \times P) \cup (Q \times P_{NOT})$, see [SCST93], the last relation can be defined in an alternative way as the union of two disjoint relations:

$$\dot{s} = s \cup s_{NOT}$$

where

$$s : Q \times P \to \{0,1\} \text{ with } s(q,p) = \begin{cases} 1, & p \in q \\ 0, & p \notin q \end{cases} \text{ and}$$

$$s_{NOT} : Q \times P_{NOT} \to \{0,1\} \text{ with } s(q, NOTp) = \begin{cases} 1, & NOTp \in q \\ 0, & NOTp \notin q \end{cases}$$

It is a similar proof (details in [MOUM98]) as for *theorem 1* to show that:

$$F = (s \circ r) \cup (s_{NOT} \circ \bar{r})$$

Evaluation of simple conjunctive queries

The next class of simple queries to be studied is simple conjunctive queries of the form $q = \underset{j=1}{\overset{n}{AND}}\, p_j$. Again, such a query can be considered the subset $\{p_1, p_2, \ldots, p_n\} \subseteq P$ as in the case of simple disjunctive queries. Consequently, the set Q of valid queries in this case is a subset of the powerset of P, i.e. $Q \subseteq \wp(P)$. The relation that defines the association between such queries and index terms from P is the following:

$$s{:}Q\times P\rightarrow\{0,1\} \text{ with } s(q,p)=\begin{cases}1, & p\in q\\ 0, & p\notin q\end{cases}$$

Unfortunately, the evaluation function F in this case cannot be computed using the composition operation as it was the case for simple disjunctive queries. However, we may define a new operation for this purpose, which we will call *conjunctive composition* and use the symbol • for it.

Definition 5: Conjunctive composition

The conjunctive composition between two boolean relations $r\subseteq V\times W$ and $s\subseteq W\times Z$ is defined as follows[5]:

$$(r\bullet s)(v,z)=\underset{w\in W}{AND}\left(NOT r(v,w) OR s(w,z)\right) \text{ or}$$

$$(r\bullet s)(v,z)=\underset{w\in W}{MIN}\{MAX\{1-r(v,w),s(w,z)\}\}$$

Based on this operation, we will prove the next theorem, which establishes an algebraic mechanism for the evaluation of simple conjunctive queries.

Theorem 2:

Let $\mathbb{A}=\langle P,E,Q,F\rangle$ be a Boolean IRS. Assuming that it accepts only simple conjunctive queries of the form $q=\overset{n}{\underset{j=1}{AND}}\,p_j$, we define boolean relations r and s as above. Then, the evaluation function F of $\mathbb{A}$ is given by the expression: $F=s\bullet r$

Proof

Let $\langle q,e\rangle\in Q\times E$. We are going to show that:

$$F(q,e)=(s\bullet r)(q,e) \tag{7}$$

Taking the right hand side of this equation, we have

$$(s\bullet r)(q,e)=\underset{p\in P}{AND}\left(NOT s(q,p) OR r(p,e)\right)=\underset{p\in P}{MIN}\{MAX\{1-s(q,p),r(p,e)\}\}$$

and by re-ordering the elements:

$$(s\bullet r)(q,e)=MIN\left\{\underset{p\in q}{MIN}\{MAX\{1-s(q,p),r(p,e)\}\},\underset{p\in(P-q)}{MIN}\{MAX\{1-s(q,p),r(p,e)\}\}\right\} \tag{8}$$

Moreover, for $p\notin q$ it is $s(q,p)=0$, so $MAX\{1-s(q,p),r(p,e)\}=MAX\{1,r(p,e)\}=1$ for each $p\in(P-q)$. Consequently, $\underset{p\in(P-q)}{MIN}\{MAX\{1-s(q,p),r(p,e)\}\}=1$. Substituting this result in (8), we take:

$$(s\bullet r)(q,e)=\underset{p\in q}{MIN}\{MAX\{1-s(q,p),r(p,e)\}\}$$

However, for $p\in q$ it is $s(q,p)=1$ and consequently, $MAX\{1-s(q,p),r(p,e)\}=$ $MAX\{0,r(p,e)\}=r(p,e)$. So the last equation gives:

[5] Note that the conjunctive composition does not satisfy commutativity.

$$(s \bullet r)(q,e) = \underset{p \in q}{MIN}\{r(p,e)\} \tag{9}$$

Taking now the left-hand side of (7) and using *definition 1* and equation (1) we have:

$$F(q,e) = \underset{p \in q}{MIN}(F(p,e)) = \underset{p \in q}{MIN}(r(p,e)) \tag{10}$$

Combining (9) and (10) we conclude that the equation (7) holds. □

Evaluation of simple conjunctive queries with negation

In case the queries that are recognized by the system are allowed to have negations, we have the form of simple conjunctive queries with negation: $q = \underset{j=1}{\overset{n_q}{AND}}\, \dot{p}_{k_{q,j}}$, with $\dot{p}_{k_{q,j}} \in \dot{P} = P \cup P_{NOT}$. Now, the set of valid queries can be considered as subset of the powerset of $\dot{P}$, i.e. $Q \subseteq \wp(\dot{P})$. The relation that represents Q is the following:

$$\dot{s} : Q \times \dot{P} \to \{0,1\}, \text{ with } \dot{s}(q,p) = \begin{cases} 1, & \dot{p} \in q \\ 0, & \dot{p} \notin q \end{cases}$$

Due to the fact that $Q \times \dot{P} = Q \times (P \cup P_{NOT}) = (Q \times P) \cup (Q \times P_{NOT})$, the last relation can be defined in an alternative way as the union of two disjoint relations:

$$\dot{s} = s \cup s_{NOT}$$

where

$$s : Q \times P \to \{0,1\} \text{ with } s(q,p) = \begin{cases} 1, & p \in q \\ 0, & p \notin q \end{cases} \text{ and}$$

$$s_{NOT} : Q \times P_{NOT} \to \{0,1\} \text{ with } s(q, NOTp) = \begin{cases} 1, & NOTp \in q \\ 0, & NOTp \notin q \end{cases}$$

It is a similar proof as for *theorem 2* (see [MOUM98] for details) to show that:

$$F = (s \bullet r) \cap (s_{NOT} \bullet \bar{r})$$

Evaluation of queries in normal form

Having studied simple queries (either disjunctive or conjunctive) we are now ready to study the most general form of boolean queries: Queries in normal form. We remind the reader that any boolean query can be re-written in an equivalent form in either conjunctive or disjunctive normal form. This means that a mechanism to evaluate queries in normal form can be used to answer any boolean query. We will study queries in conjunctive normal form. The study of disjunctive normal form queries is analogous.

Assuming a Boolean IRS $\mathbb{A} = \langle P, E, Q, F \rangle$ we consider first queries of the form $q = \underset{i=1}{\overset{n}{AND}}\left(\underset{j=1}{\overset{n_i}{OR}}\, p_{k_j}\right)$ without negation. Let Q_{OR} be the set of disjuncts in all valid queries. Any disjunct can be considered as a subset of P, so we may define a relation that associates any disjunct with the index terms in it:

$$s : Q_{OR} \times P \to \{0,1\} \text{ with } s(q_{OR}, p) = \begin{cases} 1, & p \in q_{OR} \\ 0, & p \notin q_{OR} \end{cases}$$

The evaluation function F_{OR} of these disjuncts, due to *theorem 1*, is given by:

$$F_{OR} = s \circ r \tag{11}$$

In a similar manner, any query $q = \underset{i=1}{\overset{n}{AND}}\left(\underset{j=1}{\overset{n_i}{OR}}\, p_{k_j}\right)$ can be considered as a subset of Q_{OR} containing the disjuncts $\underset{j=1}{\overset{n_i}{OR}}\, p_{k_j}, i = 1,2,\ldots,n$. The relation that represents the association between queries and disjuncts is defined as:

$$t : Q \times Q_{OR} \to \{0,1\} \text{ with } t(q, q_{OR}) = \begin{cases} 1, & q_{OR} \in q \\ 0, & q_{OR} \notin q \end{cases}$$

The evaluation function for this queries, due to *theorem 2* is $F = t \bullet F_{OR}$ and due to equation (11), this expression is:

$$F = t \bullet (s \circ r)$$

It is now easy to elaborate the most general expression for the evaluation of conjunctive normal form queries (with negation):

$$F = t \bullet ((s \circ r) \cup (s_{NOT} \circ \bar{r}))$$

In case of disjunctive normal form queries, we have:

$$F = t \circ ((s \bullet r) \cap (s_{NOT} \bullet \bar{r}))$$

where the relations *s* and *t* are defined as follows:

$$s : Q_{AND} \times P \to \{0,1\} \text{ with } s(q_{AND}, p) = \begin{cases} 1, & p \in q_{AND} \\ 0, & p \notin q_{AND} \end{cases}$$

$$t : Q \times Q_{AND} \to \{0,1\} \text{ with } t(q, q_{AND}) = \begin{cases} 1, & q_{AND} \in q \\ 0, & q_{AND} \notin q \end{cases}$$

where Q_{AND} is the set of all conjuncts present in valid queries.

3.2. Ranking Relations and Query Evaluation in a Fuzzy IRS

In the previous section we have developed a methodology to compute algebraically the evaluation function of a Boolean IRS. The key idea in this methodology is the representation of queries by boolean relations and the representation of relations through their characteristic functions. This methodology is extended for a Fuzzy IRS through the extension of boolean relation to ranking relations.

3.2.1. The notion of Ranking Relation

In a boolean relation, the characteristic function can take only one of two possible values: either 1 (true) or 0 (false). In a ranking relation the characteristic function should be allowed to range from 0 to 1. So the definition of a ranking relation is the following:

Definition 6: Ranking Relation

A ranking relation *r* between two sets *V* and *W* is a function *r*:*V*×*W*→[0,1]. The set of ranking relations from *V* to *W* will be denoted by $[0,1]^{V \times W}$.

3.2.2. Basic operations in Ranking Relations

The *complement* of *r*, denoted by $\bar{r}$ is defined as $\bar{r}(v,w) = f_{NOT}(r(v,w)) = 1 - r(v,w)$.

The union and intersection of ranking relations are defined as:

$$\textit{Union}: \quad (r \cup_M s)(v,w) = f_{OR}(r(v,w), s(v,w))$$

$$\textit{Intersection}: \quad (r \cap_M s)(v,w) = f_{AND}(r(v,w), s(v,w))$$

It is apparent that different fuzzy models imply different evaluation for union and intersection as they are defined in terms of the basic evaluation functions. In a similar fashion, the composition and the conjunctive composition are different for each fuzzy model. In general we will define the composition of ranking relations as follows:

Definition 7: Composition of Ranking Relations

The composition between two ranking relations $r \in [0,1]^{V\times W}$ and $s \in [0,1]^{W\times Z}$ is defined as a new ranking relation (r©s):V×Z→[0,1]. The definition is based on a two other operations, a *multiplicative* operation ⊗:[0,1]×[0,1]→A and an *additive* operation ⊕:A × A →[0,1] as follows:

$$(r © s)(v,z) = \bigoplus_{w\in W} \left(r(v,w) \otimes s(w,z)\right)$$

A is a set of intermediate values, usually the interval [0,1]. It is apparent that for any pair of operations ⊕ and ⊗, a different composition function is defined. The objective is to define these operations properly in order to be able to evaluate queries in Fuzzy IRS.

3.2.3. Using Ranking Relations to Evaluate Similarity Queries in a Fuzzy IRS

Let B=$\langle P,E,Q,F\rangle$ be a Fuzzy IRS. The relation r:P×E→[0,1] with $r(p,e)=e(p)$ is the *basic evaluation ranking relation of* B.

Evaluation of Simple One Term Queries

Let us assume first that the queries recognized by the system are trivial ones of the form $q=p$. The set of valid queries is exactly the set P of index terms. The evaluation function F:P×E→[0,1] is $F(p,e)=e(p)$. However, the basic evaluation ranking relation r defined above is exactly the same. So, in this case, the evaluation function is given by the following equation:

$$F= r$$

Evaluation of Simple One Term Negation Queries

Let us assume now that the queries recognized by the system are of the form $q=NOTp$. Using the symbol P_{NOT} for the set $\{NOTp \mid p\in P\}$ of valid queries, the evaluation function F:P_{NOT}×E→[0,1] is $F(NOTp,e)=1-e(p)=1-r(p,e)=\bar{r}(p,e)$. That is, in this case:

$$F = \bar{r}$$

Evaluation of Simple Disjunctive Queries

In case the valid queries are in simple disjunctive form, any such query $q = \underset{j=1}{\overset{n}{OR}}\, p_j$, can be considered as the subset $\{p_1, p_2, \ldots, p_n\} \subseteq P$. The set Q of valid queries is then $Q \subseteq \wp(P)$ and we can define a ranking relation that associates queries and index terms as follows:

$$s{:}Q\times P\to\{0,1\} \text{ with } s(q,p) = \begin{cases} 1, & p\in q \\ 0, & p\notin q \end{cases}$$

In order to evaluate algebraically theses queries like the way *theorem 1* established in the case of a Boolean IRS, we should define an appropriate composition operation © based on two operations ⊕ and ⊗. The definition of this composition operation depends on the basic evaluation function for the disjunction which is different for each fuzzy model. If we denote any such fuzzy model as M, then we will use the notation $©_M^{OR}$ for the composition and $\oplus_M^{OR}$, $\otimes_M^{OR}$ for the corresponding operations. This definition should ensure that the following equation holds for any $\langle q,e\rangle \in Q\times E$:

$$F(q,e) = \left(s ©_M^{OR} r\right)(q,e) \Leftrightarrow f_{OR}\left(F(p_1,e),\ldots,F(p_n,e)\right) = \bigoplus_{p\in P} {}_M^{OR} \left(s(q,p) \otimes_M^{OR} r(p,e)\right) \tag{12}$$

Having defined the composition in such a way, the evaluation function F is given by the following expression:

$$F = s ©_M^{OR}\, r$$

Evaluation of Simple Conjunctive Queries

In case the valid queries are in simple conjunctive form, any such query $q = \overset{n}{\underset{j=1}{AND}}\, p_j$, can be considered as the subset $\{p_1, p_2, \ldots, p_n\} \subseteq P$. The set Q of valid queries is then $Q \subseteq \wp(P)$ and we can define a ranking relation that associates queries and index terms as follows:

$$s{:}Q\times P\to\{0,1\} \text{ with } s(q,p)=\begin{cases}1, & p\in q\\ 0, & p\notin q\end{cases}$$

In order to evaluate algebraically theses queries like the way *theorem 2* established in the case of a Boolean IRS, we should define an appropriate composition operation © based on two operations ⊕ and ⊗. The definition of this composition operation depends on the basic evaluation function for the conjunction which is different for each fuzzy model. If we denote any such fuzzy model as *M*, then we will use the notation $©_M^{AND}$ for the composition and $\oplus_M^{AND}$, $\otimes_M^{AND}$ for the corresponding operations. This definition should ensure that the following equation holds for any $\langle q,e\rangle \in Q\times E$:

$$F(q,e) = \left(s ©_M^{AND} r\right)(q,e) \Leftrightarrow f_{AND}\left(F(p_1,e),\ldots,F(p_n,e)\right) = \bigoplus_{p\in P}{}_M^{AND}\left(s(q,p)\otimes_M^{AND} r(p,e)\right) \qquad (13)$$

Having defined the composition in such a way, the evaluation function *F* is given by the following expression:

$$F = s ©_M^{AND} r$$

Evaluation of Queries in Normal Form

Using relations *t*, *s*, and s_{NOT} to represent queries as in section 3.1, the evaluation function *F* for conjunctive normal form queries is computed by the equation $F = t ©_M^{AND}\left(s ©_M^{OR} r\right)$ whereas for queries in disjunctive normal form by $F = t ©_M^{OR}\left(s ©_M^{AND} r\right)$. When negation is allowed in the atoms of the query, the corresponding expressions are: $F = t ©_M^{AND}\left(s ©_M^{OR} r \cup_M s_{NOT} ©_M^{OR} \bar{r}\right)$ for queries in conjunctive normal form and $F = t ©_M^{OR}\left(s ©_M^{AND} r \cap_M s_{NOT} ©_M^{AND} \bar{r}\right)$ for queries in disjunctive normal form. Detailed proofs can be found in [MOUM98].

3.3. Ranking Relations and Query Evaluation in a Weighted Fuzzy IRS

Let C $=\langle P,E,Q,F\rangle$ be a Weighted Fuzzy IRS. In this case a simple disjunctive query $q = \overset{n}{\underset{j=1}{OR}}\left(p_j, w_j\right)$ - or a simple conjunctive query in the form $q = \overset{n}{\underset{j=1}{AND}}\left(p_j, w_j\right)$ - can be considered as the subset $\{\langle p_1,w_1\rangle,\langle p_2,w_2\rangle,\ldots,\langle p_n,w_n\rangle\} \subseteq P\times[0,1]$, and a set *Q* of such queries as a subset of the powerset of the Cartesian product *P*×[0,1], i.e. $Q \subseteq \wp(P\times[0,1])$. The ranking relation

$$s{:}Q\times P\to[0,1] \text{ with } s(q,p)=\begin{cases}w, & \langle p,w\rangle\in q\\ 0, & \forall w\in(0,1]\,\langle p,w\rangle\notin q\end{cases}$$

expresses the association of queries from *Q* to index terms from *P*.

The evaluation function is again $F = s ©_M^{OR} r$ in the case of simple disjunctive queries and $F = s ©_M^{AND} r$ in the case of simple conjunctive queries, provided that the composition operators $©_M^{OR}$ and $©_M^{AND}$ have been defined properly in order to hold equation (12) and (13) respectively. In case of general queries in normal form, we will have again $F = t ©_M^{AND}\left(s ©_M^{OR} r \cup_M s_{NOT} ©_M^{OR} \bar{r}\right)$ for queries in conjunctive normal form and $F = t ©_M^{OR}\left(s ©_M^{AND} r \cap_M s_{NOT} ©_M^{AND} \bar{r}\right)$ for queries in disjunctive normal form. Relations *r*, *s*, and s_{NOT} are defined appropriately to represent queries. Detailed proofs can be found in [MOUM98].

3.4. Definition of Composition Operations in Weighted Fuzzy Models

As stated in section 2, the most general IRS are Weighted Fuzzy ones. So, let C=$\langle P,E,Q,F\rangle$ be a Weighted Fuzzy IRS. According to the methodology we have developed in the previous sub-section, for each fuzzy model *M*, we have to define two composition operations $©_M^{OR}$ and $©_M^{AND}$. Of course, for each one of these two operations, we have to define the corresponding multiplication and addition operation, so in total we have to define the operations $\oplus_M^{OR}$, $\otimes_M^{OR}$, $\oplus_M^{AND}$, $\otimes_M^{AND}$ in order to satisfy equations (12) and (13). In this subsection we are going to give the general methodology of deriving these opertation in terms of an example. We are also going to give the operations for all the models presented in section 2 without proof. The interesting reader can find these proofs in [MOUM98].

3.4.1. Deriving the Disjunctive Composition for p-norm

Let $q = \overset{n}{\underset{j=1}{OR}}\left(p_j, w_j\right)$ be a simple disjunctive query in a Weighted Fuzzy IRS on the p-Norm model. From equation (13) we have:

$$F(q,e) = \left(s ©_{PN}^{OR} r\right)(q,e) \Leftrightarrow$$

$$\Leftrightarrow f_{OR}\left(\left\langle F\left(p_{q_1},e\right), w_{q_1}\right\rangle,\ldots,\left\langle F\left(p_{q_n},e\right), w_{q_n}\right\rangle\right) = \underset{p\in P}{\oplus_{PN}^{OR}}\left(s(q,p)\otimes_{PN}^{OR} r(p,e)\right) \Leftrightarrow$$

$$\Leftrightarrow f_{OR}\left(\left\langle r\left(p_{q_1},e\right), w_{q_1}\right\rangle,\ldots,\left\langle r\left(p_{q_n},e\right), w_{q_n}\right\rangle\right) = \underset{p\in P}{\oplus_{PN}^{OR}}\left(s(q,p)\otimes_{PN}^{OR} r(p,e)\right) \tag{14}$$

Substituting the basic evaluation function for the disjunction and transforming properly the resulting equation, the corresponding operations are defined. More precisely we start from the left-hand side of equation (14) and we substitute f_{OR}. The resulting expression has to be brought in a from in which all index terms from *P* are present and not just the index terms present in the query *q*. It is apparent that the rest of the terms in *P* not present in *q*, have zero weight. From this last form, the proper definition of the composition operation should be apparent.In order to avoid confusion, we substitute the parameter *p* in the definition of the evaluation function f_{OR} by *g*. Starting from the LHS of (14) we have:

$$f_{OR}\left(\left\langle r\left(p_{q_1},e\right), w_{q_1}\right\rangle,\ldots,\left\langle r\left(p_{q_n},e\right), w_{q_n}\right\rangle\right) =$$

$$\left(\frac{\left(\sum_{i=1}^{n} r\left(p_{q_i},e\right)^g \cdot w_{q_i}^g\right)}{\sum_{i=1}^{n} w_{q_i}^g}\right)^{1/g} = \left(\frac{\sum_{i=1}^{n} r\left(p_{q_i},e\right)^g \cdot s\left(q,p_{q_i}\right)^g}{\sum_{i=1}^{n} s\left(q,p_{q_i}\right)^g}\right)^{1/g} = \left(\frac{\sum_{i=1}^{n} r\left(p_{q_i},e\right)^g \cdot s\left(q,p_{q_i}\right)^g + 0}{\sum_{i=1}^{n} s\left(q,p_{q_i}\right)^g + 0}\right)^{1/g} =$$

$$= \left(\frac{\left(\sum_{i=1}^{n} r\left(p_{q_i},e\right)^g \cdot s\left(q,p_{q_i}\right)^g\right) + \left(\sum_{i=1}^{m} r\left(p_{\bar{q}_i},e\right)^g \cdot s\left(q,p_{\bar{q}_i}\right)^g\right)}{\sum_{i=1}^{n} s\left(q,p_{q_i}\right)^g + \sum_{i=1}^{n} s\left(q,p_{\bar{q}_i}\right)^g}\right)^{1/g} = \left(\frac{\sum_{p\in P} r(p,e)^g \cdot s(q,p)^g}{\sum_{p\in P} s(q,p)^g}\right)^{1/g} =$$

$$= \left(\sum_{p\in P} r(p,e)^g \cdot \frac{s(q,p)^g}{\sum_{p\in P} s(q,p)^g}\right)^{1/g} = \left(\sum_{p\in P} r(p,e)^g \cdot \left(\frac{s(q,p)}{\sqrt[g]{\sum_{p\in P} s(q,p)^g}}\right)^g\right)^{1/g}$$

It is apparent from the last expression, that we have to modify the query weights (ie. the elements of ranking relation *s*) as $w'_{q_i} = \dfrac{w_{q_i}}{\sqrt[g]{\sum_{i=1}^{n} w_{q_i}^g}}$. So, we define a modified ranking relation s', with

$s'(q,p) = \frac{s(q,p)}{\sqrt[g]{\sum_{p\in P} s(q,p)^g}}$. Using s' the evaluation expression becomes:

$$f_{OR}\left(\left\langle r\left(p_{q_1},e\right),w_{q_1}\right\rangle,\ldots,\left\langle r\left(p_{q_n},e\right),w_{q_n}\right\rangle\right) = \left(\sum_{p\in P} r(p,e)^g \cdot s'(q,p)^g\right)^{1/g}$$

In conclusion, the definition for the composition operations, be re-substituting g by p is $a \otimes_{PN}^{OR} b = (a\cdot b)^p$ for multiplication and $a_1 \oplus_{PN}^{OR} a_2 \oplus_{PN}^{OR} \ldots \oplus_{PN}^{OR} a_n = \left(\sum_{i=1}^{n} a_i\right)^{1/p}$ for addition.

3.4.2. The Complete Set of Composition Operations in Weighted Fuzzy Models

For ***p*-Norm**, query weights are modified as $w'_{q_i} = \frac{w_{q_i}}{\sqrt[p]{\sum_{i=1}^{n} w_{q_i}^p}}$, for ***Infinite One*** as $w'_{q_i} = \frac{w_{q_i}}{\sum_{i=1}^{n} w_{q_i}}$, for ***Fuzzy Set*** and ***Waller-Kraft***, query weights are either 1 or 0. The composition operations derived from equations (12) and (13), taking into account the modified query weights are the following:

***P*-norm**

$$a \otimes_{PN}^{OR} b = (a\cdot b)^p,\quad a_1 \oplus_{PN}^{OR} \ldots \oplus_{PN}^{OR} a_n = \left(\sum_{i=1}^{n} a_i\right)^{1/p}$$

$$a \otimes_{PN}^{AND} b = a^p \cdot (1-b)^p = (a\cdot(1-b))^p,\quad a_1 \oplus_{PN}^{AND} \ldots \oplus_{PN}^{AND} a_n = 1 - \left(\sum_{i=1}^{n} a_i\right)^{1/p}$$

Infinite One

$$a \otimes_{IO}^{OR} b = \langle a, a\cdot b\rangle,\quad \langle a_1,b_1\rangle \oplus_{IO}^{OR} \ldots \oplus_{IO}^{OR} \langle a_n,b_n\rangle = \gamma\cdot\frac{MAX\{b_1,\ldots,b_n\}}{MAX\{a_1,\ldots,a_n\}} + (1-\gamma)\cdot\sum_{i=1}^{n} b_i$$

$$a \otimes_{IO}^{AND} b = \langle a, a\cdot(1-b)\rangle,\quad \langle a_1,b_1\rangle \oplus_{IO}^{AND} \ldots \oplus_{IO}^{AND} \langle a_n,b_n\rangle = \gamma\cdot\left(1-\frac{MAX\{b_1,\ldots,b_n\}}{MAX\{a_1,\ldots,a_n\}}\right) + (1-\gamma)\left(1-\sum_{i=1}^{n} b_i\right)$$

Fuzzy Set

$$a \otimes_{FS}^{OR} b = MIN\{a,b\} = a\cdot b,\quad a_1 \oplus_{FS}^{OR} \ldots \oplus_{FS}^{OR} a_n = MAX\{a_1,\ldots,a_n\}$$

$$a \otimes_{FS}^{AND} b = MAX\{1-a,b\},\quad a_1 \oplus_{FS}^{AND} \ldots \oplus_{FS}^{AND} a_n = MIN\{a_1,\ldots,a_n\}$$

Waller-Kraft

$$a \otimes_{WK}^{OR} b = \langle MAX\{1-a,b\}, MIN\{a,b\}\rangle,$$
$$\langle a_1,b_1\rangle \oplus_{WK}^{OR} \ldots \oplus_{WK}^{OR} \langle a_n,b_n\rangle = (1-\gamma)\cdot MIN\{a_1,\ldots,a_n\} + \gamma\cdot MAX\{b_1,\ldots,b_n\}$$

$$a \otimes_{WK}^{AND} b = \langle MAX\{1-a,b\}, MIN\{a,b\}\rangle,\quad \langle a_1,b_1\rangle \oplus_{WK}^{AND} \ldots \oplus_{WK}^{AND} \langle a_n,b_n\rangle = (1-\gamma)\cdot MIN\{a_1,\ldots,a_n\} + \gamma\cdot MAX\{b_1,\ldots,b_n\}$$

4. Translation of REALM in SQL

In this section we describe the translation of algebraic expression in REALM to SQL queries and show the general scheme of answering similarity queries on top of relational databases. We also briefly sketch the implementation that has been done. The basic idea is to represent ranking relations as relational tables and basic operation in ranking relations as SQL queries.

4.1. The SQL Query that Computes the Composition

The basic operation in REALM expressions is composition. As we have already seen a composition operation is defined in terms of two operations $\otimes$ and $\oplus$. Having specified these two operations it is quite easy to translate a composition expression into an SQL query. Let us see how. Assume that we would like to implement the expression *r©s* in SQL. Furthermore, assume that ranking relation *r* is stored in a relational table R(A,B) and ranking relation *s* is stored in table S(B,C)[6]. The elements of the composition of these two ranking relations can be computed with the following query:

```
    Qcomp:   SELECT R.A, S.C, ⊕(⊗(R.W,S.W))
             FROM R, S
             WHERE R.B=S.B
             GROUP BY R.A, S.C
```

In case the result should be ordered, an ORDER BY clause is added. It is not so difficult to see why this query correctly computes the composition of ranking relations R and S. Consider the query:

```
   Qcomp1:   SELECT R.A, S.C, ⊗(R.W,S.W))
             FROM R, S
             WHERE R.B=S.B
```

The tuples in the answer table are exactly the factors in the expression computing each term the composition in composition *r©s*. Grouping together these factors and taking the aggregate function to compute the composition expression, we arrive at the query expression Qcomp. The issue now is to determine the functions ⊕ and ⊗ in order to compute correctly the expression ⊕(⊗(R.W,S.W)). It is apparent that these expressions are already known and that they are exactly the same as the expressions computing the composition. However there is a tricky point here. For some composition operations (e.g. the conjunctive composition for Fuzzy Set) there is a need for some modification in the operation in order to be implementable in SQL taking into account that only non-zero weighted tupples are stored in the relational database (see example 3 in section 1.1) Due to space limitation we do not give here these modifications. The interested reader can find them in [MOUM98].

4.2. A General Translation Process

In order to have a general translation process we should first specify the general form of similarity queries and the corresponding expressions in REALM. For example if we decide to accept queries in conjunctive normal form without negation, the REALM expression that should be implemented in SQL is of the form $F = t©_M^{AND}\left(s©_M^{OR} r\right)$. Having decided the general form of queries, we should then specify the tables that store the corresponding ranking relations and the evaluation model used. Then a series of SQL queries can be constructed. In case of the conjunctive normal form queries we are considering, we need to construct two SQL queries (based on the template Qcomp query) corresponding to the two composition operations in the REALM expression. When a query needs to be computed, we should insert the appropriate tuples in the relational tables corresponding to the ranking relation used to represent the query. Then the SQL queries are executed and the final result is taken (usually ordered, using an ORDER BY clause in the last SQL query in the chain of SQL queries computing the REALM expression).

4.3. Implementation of REALM

REALM offers a powerful mathematical tool for the description of similarity queries and different evaluation models. It has been implemented on top of a relational DBMS as a distributed system. The general architecture of this system is shown in Figure 3. ‘Schema editor’ is a graphical tool, which is used to define the mapping between ranking relations and relational tables. It is also responsible to create the relational tables that are used to store the ranking relations corresponding to similarity queries (like the tables OR Table and AND Table in the example given in section 1). The mappings are then stored in the ‘Map File’. Based on this mappings, the ‘Retrieval Engine’ is able to translate retrieval requests from any ‘Application’ into SQL queries sent to the ‘Relational DB’ and give back the results to the ‘Application’ when requested. The ‘Retrieval Engine’ is also responsible for decomposing a retrieval request in a series of INSERT statements to populate the tables used to store the queries (like OR Table and AND Table in the example of section 1). The current implementation supports the Microsoft Access DBMS and offers queries in conjunctive normal form. This system has been used successfully in a number of applications including a digital library application in the conext of the ESPRIT 20638 VENIVA project. Details on this application as well as on the implementation of REALM can be found in [ANES97].

[6] It is apparent that tables R and S may also contain other attributes. However for the description of the general translation scheme this detail is not significant.

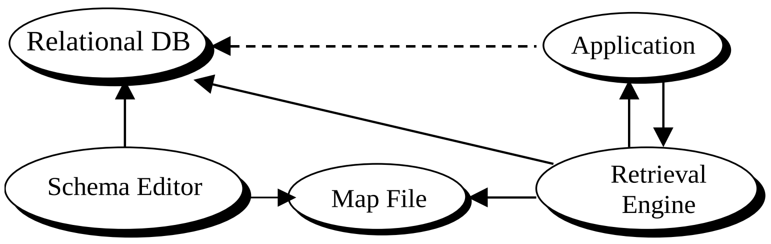

Figure3. *The architecture of REALM system*

5. Related Work

A related research area is the so-called mediators or middleware for heterogeneous information sources. Examples in this field are systems that provide uniform interface and queries across diverse information sources either on the Internet such as Garlic [HKWY97] or in the context of a multimedia information system such as the one developed in the TSIMMIS project [HBGN97]. A major objective in these systems is the uniform presentation of query capabilities of the information sources and the translation of user queries to queries in the underlying systems [VAPA97] as well as the transformation of source ranks (coming from the underlying systems) to target ranks (shown to the user) [GRGA97]. Another problem studied is the optimization of query execution in order to find to k top-ranked objects without the need to access all the ranks from the information sources [FAGI96]. A similar approach based on heuristic thresholds on rank values is also reported in [PERS94] in the context of processing inverted lists in an IRS. A rule based approach to query optimization in mediators is presented in [HKWY97]. The difference of our approach is that we assume a relational DBMS as the underlying system and not any information source and we build additional query capabilities on top of it instead of using the already existing capabilities. However there is a strong correlation between our notion of fuzzy evaluation functions and the corresponding models found in [FAGI96].

Another approach is the integration of content-specific search engines in relational DBMS [DEMA97, GFHR97, DDSS95, LYST88]. In all of these efforts, extensibility characteristics of specific DBMS are exploited to build text-specific data types. In [LYST88] special access methods are implemented to support the new data type. In [GFHR97] relational tables are used to store the text index whereas [DDSS95] uses for this purpose an enhancement of the database engine which is essentially a capability to store a table as ‘index only’ or as an inverted table structure. [DEMA97] is based on a query rewrite scheme that exploits so-called table functions, which are used to pass results from external search engines into the database engine. This is apparently different from the approaches in [GFHR97, DDSS95, LYST88] and resembles a mediator. The difference lies in the close coupling of the relational DBMS with the external search engines which facilitates effective optimization of queries. The difference with this kind of approaches from ours is that in our framework there is no need for extensibility features in the DBMS. Moreover our approach is not text-specific, it can accommodate similarity queries on any kind of objects as long as the association between objects and index terms are stored in relational tables. Moreover our approach can accommodate different fuzzy models for query evaluation.

An alternative direction for accommodating non-boolean queries in IRS and DBMS is the use of probabilistic models. Probabilistic theories are quite attractive due to their theoretical coherence and deductive power. [COOP94] considers the question whether the trappings of the probabilistic formalism strengthen or encumber IRS research. The major conclusion is that the cost of creating and trouble-shooting probabilistic theories in Information Retrieval is high, so “time will tell whether the theoretical baggage that accompanies the probabilistic method is more benefit or an encumbrance”. In the DBMS field one of the most interesting work is [LLRS97], which describes a probabilistic relational data model, and a generic probabilistic relational algebra that neatly captures various strategies satisfying the postulates within a single unified framework. These strategies resemble different fuzzy models in our work. In the same paper, algorithms are also given for maintaining materialized probabilistic views and a prototype implementation on top of a relational DBMS is described. This work has some common ground with ours in the sense that the model given is built on top of a relational DBMS and different models can be supported. However our approach is beneficial from the implementation point of view as it is more simple and straightforward. Moreover, probabilistic models are not well suited for some application fields as fuzzy models as reported in [COOP94].

In [SHAL84 and JOON94] models of extended boolean logic in IRS are presented and studied. Our work is based on these models and we provide mechanisms for implementing them on top of relational DBMS. In [JAMM95] a domain-independent framework for defining queries in terms of similarity of objects is developed. This framework has three components: a pattern language, a transformation rule language and a query language. This work merely specifies a means of expressing similarity of objects representing sequences (such as stock prices over time, molecules, text strings or images). Our work is not considering the mechanisms for extracting the initial similarity measure. It takes it for granted and proposes a framework for combining these similarity measures in queries.

[WHSM97] implements a yellow page service that enables web users to search throughout a listing of 11 million businesses in over 17000 categories using an IR engine to search through complex listing objects. The essence of the approach is to map logical units of the database (e.g. listing objects and their associated sub-object networks) into an indexed collection of structured text documents. These documents are then searched using an IRS search engine instead of the database's native query processor. The IR engine is the Verity's VDK engine and the object oriented DBMS is Object Design's ObjectStore database. The database is used only for persistency, the query processor is not used at all. Loose integration of the IRS engine and the DBMS results in several drawbacks such as update problems and data duplication. This work is similar to our approach in terms of functionality. In terms of implementation it adopts an IRS as the basis instead of a DBMS. In addition, [WHSM97] describes just an application but no general mechanisms for translation of database objects to structured documents and user queries to retrieval requests is given.

6. Conclusions and Future Work

In this paper we have presented a model for answering boolean and similarity queries based on different fuzzy models. The main advantage of our approach is the immediate implementation on top of existing relational database management systems. To the best of our knowledge there is no similar effort reported in the corresponding bibliography assuming no extensibility feateures for the underlying relational DBMS. Organizations with existing relational databases may take advantage of our work to build powerful similarity retrieval applications using their already existing data. This can help decision making processes or the development of WWW sites, a trend quite common today Further research considers the elaboration of proper processes for the handling of hierarchies [MOUM98]. Relevance feedback is well suited in this model as the index terms and the indexed objects can be used interchangeably. Another research direction is the support for special access structures and query optimization algorithms.

This work has been carried out in the framework of the project No 25572, CAMPIELLO – "Interacting in collaborative environments to promote and sustain the meeting between inhabitants and tourists", funded by the European Union under the ESPRIT Long Term Research scheme (i^3 – inteligent information interfaces domain, task 4.4, the connected community schema).

References

[ANES97] G. Anestis: "Design and Implementation of a Boolean and Similarity Retrieval System on top of Relational Database Management Systems", Diploma Thesis, Department of Electronic and Computer Engineering, Technical University of Crete, Chania, 1997. https://dias.library.tuc.gr/view/manf/14758

[BESH97] K. Beard, V. Sharma: "Multidimensional ranking for data in digital spatial libraries", International Journal on Digital Libraries, vol. 1, number 2, pp 153-160, September 1997.

[COOP94] W. S. Cooper: "The Formalism of Probability Theory in IR: A Foundation or an Encumbrance?", In Proceedings of the 17th ACM SIGIR International Conference on Research and Development in Information Retrieval, 1994, 242-247.

[DDSS95] S. DeFacio, A. Daoud, L. A. Smith, J. Srinivasan: "Integrating IR and RDBMS Using Cooperative Indexing" In Proceedings of the 18th ACM SIGIR International Conference on Research and Development in Information Retrieval, 1995, pp 84-92.

[DEMA97] S. Dessloch, N. Mattos: "Integrating SQL Databases with Content-specific Search Engines", Proceedings of the 23rd VLDB Conference Athens, Greece, 1997, pp 528-537.

[FAGI96] R. Fagin: "Combining Fuzzy Information from Multiple Systems", Proceedings of the 15th Symposium on Principles of Database Systems, Montreal, Canada, June 1996, pp 216-226.

[GFHR97] D. A. Grossman, O. Frieder, D. O. Holmes, D. C. Roberts: "Integrating Structured Data and Text: A Relational Approach", Journal of the American Society of Information Science, vol. 48, no. 2, February 1997.

[GRGA97] L. Gravano, H. Garcia-Molina: "Merging Ranks from Heterogeneous Internet Sources", Proceedings of the 23rd VLDB Conference Athens, Greece, 1997, pp 196-205.

[HAKW97] L. Haas, D. Kossmann, E. Wimmers, J. Yang: "Optimizing Queries across Diverse Data Sources", Proceedings of the 23rd VLDB Conference Athens, Greece, 1997, pp 276-285.

[HBGN97] J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, R. Yerneni: "Template-based wrappers in the TSIMMIS system", In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, pp 532-535.

[HJSA95] G. Hjaltason., H. Samet: "Ranking in Spatial Databases", Proceedings of 4th Symposium on Advances in Spatial Databases (SSD '95), Lecture Notes in Computer Science No. 951, Springer-Verlag, 1995, 83-95.

[JAMM95] H. V. Jagadish, A. O. Mendelzon, T. Milo: "Similarity-Based Queries", in proceedings of the 14th Symposium on Principles of Database Systems, San Jose CA, USA, 1995, pp 36-45.

[JOON94] H. L. Joon, "Properties of Extended Boolean Models in Information Retrieval", Proceedings of the 17th ACM SIGIR International Conference on Research and Development in Information Retrieval, 1994, 182-190.

[KRGK94] R. Kruse, J. Gebhardt, F. Klawonn: "Foundations of Fuzzy Systems", John Wiley & Sons Ltd., ISBN 0-471-94243-X, 1994

[LLRS97] L. Lakshmanan, N. Leone, R. Ross, V. S. Subrahmanian: "ProbView: A Flexible Probabilistic Database System", ACM Transactions on Database Systems, Vol. 22, No. 3, September 1997, pp 419-469.

[LYST88] C. Lynch, M. Stonebraker: "Extending User-Defined Indexing with Applications to Textual Databases", Proceedings of the 14th VLDB Conference Los Angeles CA, USA, 1988, pp 306-317.

[MOUM98] N. Moumoutzis: "The Design of a System Supporting the Development of Interactive Geographical Applications", MEng Thesis, Department of Electronic and Computer Engineering, Technical University of Crete, Chania, 1998.

[PERS94] M. Persin: "Document Filtering for Fast Ranking", In Proceedings of the 17th ACM SIGIR International Conference on Research and Development in Information Retrieval, 1994, pp 339-348.

[SCST93] G. Schmidt, T. Ströhlein: "Relations and Graphs", Springer-Verlag, ISBN 3-540-56254-0, 1993.

[SHAL84] G. Shalton: "The use of Extended Boolean Logic in Information Retrieval", In Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp 277-285.

[SHAL89] G. Shalton: "Automatic Text Processing", Addison-Wesley, Reading, MA, 1989

[VAPA97] V. Vassalos, Y. Papakonstantinou: "Describing and Using Query Capabilities of Heterogeneous Sources", Proceedings of the 23rd VLDB Conference Athens, Greece, 1997, pp 256-265.

[WHSM97] S. Whitehead, H. Sinha, M. Murphy: "GTE SuperPages: Using IR Techniques for Searching Complex Objects", Proceedings of the 23rd VLDB Conference Athens, Greece, 1997, pp 553-557.

Citation: *Nektarios Moumoutzis et al. "REALM: The Retrieval Algebra Model and its Implementation" International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), vol 10, no. 2, 2024, pp. 1-21. DOI: https://doi.org/10.20431/2349-4859.1002001.*