Query Systems in Similarity-Based Databases

Logical Foundations, Expressive Power, and Completeness

Radim Belohlavek*
Dept. Computer Science
Palacky University, Olomouc
17. listopadu 12, Olomouc, Czech Republic
radim.belohlavek@upol.cz

Vilem Vychodil*
Dept. Computer Science
Palacky University, Olomouc
17. listopadu 12, Olomouc, Czech Republic
vilem.vychodil@upol.cz

ABSTRACT

This paper presents logical foundations for an extension of Codd's relational model of data which aims at utilizing various aspects of similarity in data processing. A need for development of solid foundations for such extensions, sometimes called similarity-based relational databases, has repeatedly been emphasized by leading database experts. This paper argues that, contrary to what may be perceived from the literature, solid foundations for similarity-based databases can be developed in a conceptually sound way. In this paper, we outline the foundations and propose two query systems for similarity-based databases: a domain relational calculus (DRC) and a relational algebra (RA). We compare the expressive power of DRC and RA in our model and prove relational completeness of RA over DRC. A major implication of the paper is that similarity-based data querying can be made an integral part of an extended, similarity-based, relational model of data which is based on first-order predicate logics using residuated structures of truth values in much the same way as the querying in the Codd's model is based on the classic first-order predicate logic.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*relational databases*; I.2.3 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*relation systems*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic

General Terms

Languages, Management, Theory

Keywords

similarity-based databases, domain relational calculus, query system, relational databases, residuated lattices

*The authors are also with T. J. Watson School, SUNY Binghamton, Parkway E, Vestal, NY 13902–6000, USA. E-mail: rbelohla@binghamton.edu, vychodil@binghamton.edu

SAC'10 March 22-26, 2010, Sierre, Switzerland.

1. INTRODUCTION AND MOTIVATION

This paper presents logical foundations for an extension of Codd's relational model of data which aims at utilizing various aspects of similarity in data processing. The problem of uncertainty and imprecision management is considered a challenge with no satisfactory solutions obtained so far. As an example, the report from the Lowell debate by 25 senior database researchers [1] says "... current DBMS have no facilities for either approximate data or imprecise queries." According to this report, the management of uncertainty and imprecision is one of the six currently most important research directions in database systems.

In this paper, we address a particular facet of uncertainty which gained considerable attention in the past, namely similarity and imprecision and related topics such as approximate/imprecise matches and similarity-based queries. We argue that clear conceptual foundations for similarity-based databases, which have not yet been provided, can be developed in a purely logical and relational way by revisiting the classic Codd's relational model of data. Namely, we show that domain relational calculus, which is an important query system in Codd's model of data, can be formalized in first-order predicate logic using residuated structures of truth values. In addition to that, we show a corresponding relational algebra and prove its relational completeness. A major implication of our observations is that similarity-based data querying can be made an integral part of an extended, similarity-based, relational model of data which is based on residuated logics in much the same way as the querying in the Codd's model is based on the classic first-order predicate logic. The paper is meant to be a programmatic contribution which outlines the foundations of similarity-based databases, with emphasis on domain relational calculus and similarity-based queries.

An Outline of Similarity-Based Databases

A notion central to similarity-based databases is that of a ranked data table over domains with similarities which represents a counterpart of a data table (a relation) from the classic Codd's model of data. A ranked data table over domains with similarities can be seen as having three components: (*i*) an ordinary data table as in the Codd's model; (*ii*) ranks (truth values associated to each tuple in the table) indicating degrees to which tuples match a similarity-based query; and (*iii*) similarity relations on domains which define degrees to which values from a domain are similar (i.e., close or indistinguishable). The concepts are illustrated in

the following table of "houses for sale":

	id	*price*	*bdrm*	*type*
0.98	62	$195,000	2	Single Family
0.85	14	$230,000	2	Penthouse
0.73	23	$145,000	1	Log Cabin
0.40	59	$320,000	4	Ranch
0.15	49	$370,000	4	Single Family

The numbers $0.98, 0.85, \ldots, 0.15$ in the left-hand column are the ranks (they are not values of any attribute) from a scale of truth values, usually a unit interval with 0 (falsity) indicating no match and 1 (truth) indicating a full match. The remaining part of the table can be seen as a data table in the classic sense. Similarities on domains are not shown directly in the table and have to be specified independently. The similarities on domains can be seen as an additional subjective information about domains which is supplied by users of the database system. For instance, the domain of house prices can be equipped with a similarity $\approx_{price}$ which prescribes, for each two prices p_1 and p_2, a degree $p_1 \approx_{price} p_2$ to which p_1 is similar to p_2. The degree is taken from the same scale as the ranks. For instance, $\approx_{price}$ can be defined by $p_1 \approx_{price} p_2 = s(|p_2 - p_1|)$ using an antitone scaling function $s\colon [0,\infty) \to [0,1]$ with $s(0) = 1$ (i.e., identical prices are fully similar). Analogously, one can consider a similarity of the number of bedrooms. Similarities on the domain of (finitely many) house types can be described by a two-dimensional table containing selected degrees. The similarity on the domain of house IDs can remain a two-valued identity because it does not make much sense to look for "houses with similar unique identifier".

Let us note that the aforementioned table can be seen as a result of similarity-based query "show all houses which are sold for (approximately) $200,000". In general, every ranked table can be interpreted in such a way—this is an important feature which our model shares with Codd's relational model. As a result, degrees of similarity and approximate matches employed in our model replace 1 (representing exact match, or equality) and 0 (mismatch, unequality) of the classic Codd's model. In the classic model, 1 and 0 are manipulated according to the rules of classic first-order logic (in this sense, Codd's model is based on classic logic). In order to manipulate the degrees of similarity and approximate matches in very much the same way 1 and 0 are manipulated in Codd's model, we employ the recently developed calculus of first-order fuzzy logic [10, 11]. This way we obtain a generalization of Codd's model which is based on solid logical foundations and has desirable properties, cf. [4, 6]. The fomalization we offer can further be used to provide insight into several isolated approaches that have been provided in the past, see e.g. [8], [9], [13], [15], [16], [17], and a comparison paper [6].

2. PRELIMINARIES

In this section we recall basic notions of fuzzy logic and fuzzy relational systems. Details can be found in [3, 10, 11].

We are going to use complete residuated lattices with hedges [11, 12] as structures of truth values. A complete residuated lattice with a hedge is an algebra $\mathbf{L} = \langle L, \wedge, \vee, \otimes, \to, {}^*, 0, 1\rangle$ such that $\langle L, \wedge, \vee, 0, 1\rangle$ is a (complete) lattice, $\langle L, \otimes, 1\rangle$ is a commutative monoid, and $\otimes$ (multiplication) and $\to$ (residuum) satisfy the adjointness property: $a \otimes b \le c$ iff $a \le b \to c$ $(a, b, c \in L)$. Moreover, the hedge * is a unary operation satisfying (i) $1^* = 1$, (ii) $a^* \le a$, (iii) $(a \to b)^* \le a^* \to b^*$, and (iv) $a^{**} = a^*$ $(a, b \in L)$. The truth values $a \in L$ are interpreted as *degrees of truth* [11], i.e., we sometimes call $a \in L$ a truth degree (0 stands for full falsity and 1 stands for full truth). The multiplication $\otimes$ and residuum $\to$ serve as truth functions of logical connectives "fuzzy conjunction" and "fuzzy implication". The hedge * can be seen as a truth function for a unary connective "very true", see [12]. A common choice of $\mathbf{L}$ is a structure with $L = [0,1]$ (unit interval), $\wedge$ and $\vee$ being minimum and maximum, $\otimes$ being a left-continuous t-norm with the corresponding $\to$, and * being either identity or so-called globalization:

$$a^* = \begin{cases} 1, & \text{if } a = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The globalization should be understood as a truth function of logical connective "full truth" which is (in formulas) commonly denoted by Δ. An $\mathbf{L}$-set (a fuzzy set) A in universe U is a map $A\colon U \to L$, $A(u)$ being interpreted as "the degree to which u belongs to A". In a similar way, a binary $\mathbf{L}$-relation (a binary fuzzy relation) B on U is a map $B\colon U \times U \to L$, $B(u_1, u_2)$ interpreted as "the degree to which u_1 and u_2 are related according to B". See [3] for details.

Recall that a Cartesian product $\prod_{i\in I} A_i$ of an I-indexed system $\{A_i \mid i \in I\}$ of sets A_i $(i \in I)$ is a set of all maps $f\colon I \to \bigcup_{i\in I} A_i$ such that $f(i) \in A_i$ holds for each $i \in I$.

3. RELATIONAL MODEL OF SIMILARITY-BASED DATABASES

This section introduces ranked data tables over domains with similarities and related notions of similarity-based database systems. In this and the following sections, we assume that $\mathbf{L}$ is a complete residuated lattice with hedge *.

Attributes and Relation Schemes

In this paper, we use Y to denote a *set of all attributes* and denote the attributes from Y by $y, y', y_1, y_2, \ldots$ We assume that Y is infinite and enumerable. Attributes should be seen as names denoting columns of data tables in our model. Any subset $R \subseteq Y$ shall be called a *relation scheme* and plays the same role as in the ordinary model.

Domains with Similarities

As in the Codd's model, for each attribute $y \in Y$ we consider its *domain* D_y. That is, D_y is a nonempty (at most) enumerable set of all possible values of the attribute $y \in Y$. In order to formalize similarities on domains, we equip each domain D_y with a binary $\mathbf{L}$-relation $\approx_y$ on D_y which satisfies the following conditions:

(Ref) for each $u \in D_y$: $u \approx_y u = 1$, and

(Sym) for each $u, v \in D_y$: $u \approx_y v = v \approx_y u$.

Each binary $\mathbf{L}$-relation $\approx_y$ on D_y satisfying (Ref) and (Sym) will be called a *similarity*. It is easily seen that (Ref) and (Sym) are graded counterparts of reflexivity and symmetry. Sometimes it is useful to require stronger properties for $\approx_y$. Namely, one can postulate:

(Sep) for each $u, v \in D_y$: $u \approx_y v = 1$ iff u equals v;

(Tra) for each $u, v, w \in D_y$: $u \approx_y v \otimes v \approx_y w \le u \approx_y w$.

Property (Sep) is called separability and it can be seen as a stronger form of reflexivity: elements u, v in domain D_y

are similar to degree 1 iff u and v are identical. Property (Tra) is called $\otimes$-transitivity and it represents a transitivity of $\approx_y$ with respect to the residuated multiplication $\otimes$. In a more detail, (Tra) states that the degree to which u is similar to w is at least the degree to which u is similar to v and to which v is similar to w, where the interpretation of the (graded) logical "and" is given by $\otimes$.

For a domain D_y and similarity $\approx_y$ on D_y, we call the couple $\langle D_y, \approx_y \rangle$ a *domain with similarity.*

Tuples

Let $R \subseteq Y$ be a relation scheme and denote by $\mathrm{Tupl}(R)$ the Cartesian product of domains D_y ($y \in R$). Thus,

$$\mathrm{Tupl}(R) = \textstyle\prod_{y \in R} D_y.$$

Each $r \in \mathrm{Tupl}(R)$ is called a *tuple over* R. Obviously, $r(y) \in D_y$ for each $y \in R$; $r(y)$ will be called the *y-value of* r. For tuples $r \in \mathrm{Tupl}(R)$ and $s \in \mathrm{Tupl}(S)$ with $R \cap S = \emptyset$, we define a *concatenation rs of r and s* by

$$(rs)(y) = \begin{cases} r(y), & \text{if } y \in R, \\ s(y), & \text{if } y \in S. \end{cases}$$

Clearly, $rs \in \mathrm{Tupl}(R \cup S)$, i.e., rs is a tuple over $R \cup S$.

Ranked Data Tables over Domains with Similarities

Let $R \subseteq Y$ be a relation scheme. A *ranked data table on R over $\{\langle D_y, \approx_y \rangle \,|\, y \in R\}$* is any finite $\mathbf{L}$-set in $\mathrm{Tupl}(R)$. Ranked data tables will be denoted $\mathcal{D}, \mathcal{D}', \mathcal{D}_1, \ldots$

Following the definition, if $\mathcal{D}$ is a ranked data table on R over $\{\langle D_y, \approx_y \rangle \,|\, y \in R\}$ then $\mathcal{D}$ is a map $\mathcal{D}\colon \mathrm{Tupl}(R) \to L$ assigning to each tuple r over R a degree $\mathcal{D}(r) \in L$. The degree $\mathcal{D}(r)$ is called a *rank of r in $\mathcal{D}$.*

Remark 1. We make the following observations about any ranked data table $\mathcal{D}$ on $R \subseteq Y$ over $\{\langle D_y, \approx_y \rangle \,|\, y \in R\}$.

(a) Since $\mathcal{D}$ is a finite $\mathbf{L}$-set, there are only finitely many tuples $r \in \mathrm{Tupl}(R)$ with $\mathcal{D}(r) \neq 0$. Therefore, ranked data tables have finitely many tuples with nonzero ranks and they can be depicted by "finite tables" where each tuple (row of the table) has its rank. The finiteness is postulated mainly for computational reasons so that all tuples in $\mathcal{D}$ can be enumerated after finitely many steps. If $\mathcal{D}$ represents a result of similarity-based query Q then $\mathcal{D}(r)$ shall be interpreted as *the degree to which tuple r matches the query Q.*

(b) Note that $\mathcal{D}$ is, in fact, an n-ary $\mathbf{L}$-relation between domains D_y ($y \in Y$) since $\mathcal{D}$ is a map from $\prod_{y \in R} D_y$ to L. If $\mathbf{L}$ is a two-valued Boolean algebra, then $\mathcal{D}$ becomes the ordinary relation between the domains. Thus, as a particular case of a ranked data table over domains with similarities, we get a concept of a relation that is used in the Codd's relational model of data. These two observations should be understood that (i) our model is a nontrivial extension of the Codd's one and (ii) our model is based on (graded) relations and thus stays purely relational. ■

Since the domain with similarity $\langle D_y, \approx_y \rangle$ for each attribute $y \in Y$ is fixed (i.e., if y appears in a data table as an attribute, its domain with similarity is always $\langle D_y, \approx_y \rangle$), we shall call a ranked data table on R over $\{\langle D_y, \approx_y \rangle \,|\, y \in R\}$ just a ranked data table on R or a ranked data table if R is obvious. In the sequel, the term "ranked data table" will be abbreviated by "RDT". We also use a notion of a "nonranked data table" by which we mean a ranked data table where all ranks are either 0 or 1.

4. DOMAIN RELATIONAL CALCULUS

In this section we develop domain relational calculus (DRC) for our model of data. We begin with definitions of database schemes and formulas on such schemes.

Definition 1. A *database scheme on Y* is any couple $\langle \mathbb{R}, \varrho \rangle$ where $\mathbb{R}$ is a nonempty set of relation symbols and ϱ is a map $\varrho\colon \mathbb{R} \to 2^Y$ assigning to each relation symbol $\mathbb{r} \in \mathbb{R}$ a relation scheme $\varrho(\mathbb{r}) \subseteq Y$. ■

Formulas of DRC will be introduced as first-order formulas of predicate fuzzy logic based on complete residuated lattices. In order to introduce formulas, we need to specify a predicate *language.* The language of DRC over database scheme $\langle \mathbb{R}, \varrho \rangle$ consists of relation symbols from $\mathbb{R}$, variables, constants for objects from domains, constants for truth degrees (for each $a \in L$ we consider a corresponding constant $\overline{a}$, see [11]), symbols for logical connectives (conjunctions $\otimes$ and $\wedge$, disjunction $\vee$, implication $\Rightarrow$, equivalence $\Leftrightarrow$, negation $\neg$, hedge Δ) and quantifiers (existential $\exists$, universal $\forall$), and auxiliary symbols such as parentheses.

For each attribute we assume that there is an infinite enumerable set $\mathrm{Var}(y)$ of *variables of attribute $y \in Y$.* The variables will be denoted $\mathbb{x}, \mathbb{x}', \mathbb{x}_1, \mathbb{x}_2, \ldots$ We assume that $\mathrm{Var}(y_1) \cap \mathrm{Var}(y_2) \neq \emptyset$ whenever $y_1 \neq y_2$. The set of all variables will be denoted by Var, i.e., $\mathrm{Var} = \bigcup_{y \in Y} \mathrm{Var}(y)$. For each $\mathbb{x} \in \mathrm{Var}$, we let $\mathrm{Attr}(\mathbb{x}) = y$ iff $\mathbb{x} \in \mathrm{Var}(y)$, i.e., iff $\mathbb{x}$ is variable of attribute y. In addition to variables, we consider an infinite and enumerable set Con containing *constants for values from domains* which are meant to denote fixed values from domains as we shall see later. Variables and constants from $\mathrm{Var} \cup \mathrm{Con}$ shall be called *terms.*

Formulas of Domain Relational Calculus

We now proceed by defining formulas of DRC. Consider a database scheme $\langle \mathbb{R}, \varrho \rangle$. We define *atomic formulas of* $\langle \mathbb{R}, \varrho \rangle$:

- (i) If $\mathbb{r} \in \mathbb{R}$ is relation symbol with $\varrho(\mathbb{r}) = \{y_1, \ldots, y_n\}$ and if $\mathbb{x}_1 \in \mathrm{Var}(y_1), \ldots, \mathbb{x}_n \in \mathrm{Var}(y_n)$ are variables then $\mathbb{r}(\mathbb{x}_1, \ldots, \mathbb{x}_n)$ is an atomic formula.
- (ii) If p and q are terms then $p \approx q$ is an atomic formula.
- (iii) Each constant $\overline{a}$ for truth degree is an atomic formula.

Notice that formulas of the form $p \approx q$ can contain arbitrary terms whereas in $\mathbb{r}(p_1, \ldots, p_n)$, all p_i's are variables of attributes specified by the relation scheme. The order in which variables appear in $\mathbb{r}(p_1, \ldots, p_n)$ is not essential. As it is usual, more complex formulas are built from other formulas using logical connectives and quantifiers as follows:

- (iv) If φ and ψ are formulas then $(\varphi \otimes \psi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, and $(\varphi \Leftrightarrow \psi)$ are formulas.
- (v) If φ is formula, then $\neg\varphi$ and $\Delta\varphi$ are formulas.
- (vi) If φ is a formula and $\mathbb{x}$ is variable then $(\forall \mathbb{x})\varphi$ and $(\exists \mathbb{x})\varphi$ are formulas.

All formulas of domain relational calculus of $\langle \mathbb{R}, \varrho \rangle$ result by application of (i)–(vi).

Remark 2. Formulas of DRC in our model should be read as similar formulas in other symbolic logic calculi. For instance, $\mathbb{r}(\mathbb{x}, \mathbb{z}) \otimes \mathbb{s}(\mathbb{x}, \mathbb{z})$ reads "$\mathbb{x}$ and $\mathbb{z}$ are related according to $\mathbb{r}$ and $\mathbb{x}$ and $\mathbb{z}$ are related according to $\mathbb{s}$" or, in other words, "$\mathbb{x}$ and $\mathbb{z}$ are related according to $\mathbb{r}$ and $\mathbb{s}$". Analogously, $(\exists \mathbb{x})\mathbb{r}(\mathbb{x}, \mathbb{z})$ can be read as "there is $\mathbb{x}$ which is related

to $\mathbb{z}$ according to $\mathbb{r}$". Taking into account the fact that $\mathbb{r}$ is supposed to denote a graded (fuzzy) relation, we might say that the formula expresses a "degree to which there is $\mathbb{x}$ which is related to $\mathbb{z}$ according to $\mathbb{r}$". Constants for truth degrees allow us to express formulas like $\overline{a} \Rightarrow \mathbb{r}(\mathbb{x}, \mathbb{z})$ which may be read "$\mathbb{x}$ is related to $\mathbb{z}$ at least to degree a", see [3, 10, 11], etc. The hedge Δ serves as a connective "very true" or "fully true", i.e., $\Delta\varphi$ reads "φ is very/fully true". Notice, however, that at this point, formulas are just syntactic notions and they do not have any "truth value" until we define their interpretation. Nevertheless, it should be obvious that working with and understanding of formulas on the symbolic level in our model is basically the same as in the ordinary predicate logic. Let us note that when writing formulas of DRC, we accept a common rule of omitting the outermost parentheses. ■

For each formula φ of DRC we may consider occurrences of variables and classify each occurrence as free or bound. These notions are defined as in the ordinary predicate logic. We denote by $\mathrm{free}(\varphi)$ the set of all variables which have free occurrences in φ. That is, if φ is an atomic formula, then $\mathrm{free}(\varphi)$ contains all variables that occur in φ; if φ equals $\psi \otimes \chi$ then $\mathrm{free}(\varphi) = \mathrm{free}(\psi) \cup \mathrm{free}(\chi)$ and analogously for $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\neg$, and Δ; if φ equals $(\exists \mathbb{x})\psi$ then $\mathrm{free}(\varphi) = \mathrm{free}(\psi) \setminus \{\mathbb{x}\}$ and analogously for $\forall$.

Semantics of Formulas of DRC

We turn our attention to the interpretation of formulas in instances of database schemes which will allow us to consider degrees (ranks) to which formulas are true in given instances. As a first step, we show a technical assertion showing that for a set $\{\langle D_y, \approx_y \rangle \,|\, y \in Y\}$ of disjoint domains with similarities, one can consider a common domain which is a union of all D_y ($y \in Y$) and that in addition, we can introduce a similarity relation on the common domain.

THEOREM 1. *Let $\{\langle D_y, \approx_y \rangle \,|\, y \in Y\}$ be a set of pairwise disjoint domains with similarity. Then a binary* $\mathbf{L}$*-relation $\approx_U$ defined on $D_U = \bigcup_{y \in Y} D_y$ by*

$$d \approx_U e = \begin{cases} d \approx_y e, & \text{if } \{d, e\} \subseteq D_y, \\ 0, & \text{otherwise,} \end{cases}$$

is a well-defined similarity on D_U. Moreover, if each $\approx_y$ satisfies (Tra) *then $\approx_U$ satisfies* (Tra)*; if each $\approx_y$ satisfies* (Sep) *then $\approx_U$ satisfies* (Sep).

PROOF. Omitted due to a limited scope of the paper. □

Similarity $\approx_U$ from Theorem 1 is called a similarity on a common domain of D_y ($y \in Y$) induced by $\approx_y$ ($y \in Y$). The previous theorem allows us to simplify several considerations about domains with similarity relations since we can consider a single universal domain with a universal similarity. If the assumption of Theorem 1 is not satisfied because domains in question are not pairwise disjoint, we can use a "labeling" so that with each $d \in D_y$ we associate a unique identifier of the domain D_y. Hence, the assumption of having disjoint domains is negligible.

Formulas of DRC will be evaluated in database instances:

Definition 2. A *database instance* (shortly, an instance) of database scheme $\langle \mathbb{R}, \varrho \rangle$ over domains $\{\langle D_y, \approx_y \rangle \,|\, y \in Y\}$ with similarities is a quadruple $\mathbf{D} = \langle D, \approx^{\mathbf{D}}, \mathbb{R}^{\mathbf{D}}, f^{\mathbf{D}} \rangle$ where

- (i) D is a universal domain containing all D_y ($y \in Y$),
- (ii) $\approx^{\mathbf{D}}$ is similarity on D induced by $\approx_y$ ($y \in Y$),
- (iii) $\mathbb{R}^{\mathbf{D}}$ is a set of RDTs such that for each $\mathbb{r} \in \mathbb{R}$ there is an RDT $\mathbb{r}^{\mathbf{D}} \in \mathbb{R}^{\mathbf{D}}$ on $\varrho(\mathbb{r})$ over $\{\langle D_y, \approx_y \rangle \,|\, y \in Y\}$.
- (iv) $f^{\mathbf{D}}$ is a surjective map assigning to each constant $c \in \mathrm{Con}$ an element $f^{\mathbf{D}}(c) \in D$. ■

The roles of D and $\approx^{\mathbf{D}}$ in $\mathbf{D}$ are obvious. Moreover, $\mathbb{R}^{\mathbf{D}}$ is a collection of RDTs which interpret relation symbols that may appear in formulas of $\langle \mathbb{R}, \varrho \rangle$. The role of $f^{\mathbf{D}}$ is to give interpretation of constants for domain values which may appear in formulas. Since the map is surjective, each value d from the common domain D has a corresponding constant c_d such that $d = f^{\mathbf{D}}(c_d)$. Thus, if there is no danger of confusion, we may identify constants for domain values with the values themselves. Still, one has to keep in mind the distinction between the constants (parts of formulas, i.e., syntactic notions) and values from D (i.e., semantic elements).

In order to define a value to which φ is true in given $\mathbf{D}$, we have to specify valuation of variables which occur free in φ. Thus, any map $\boldsymbol{v} \colon \mathrm{Var} \to D$ will be called a $\mathbf{D}$*-valuation of variables* (shortly, a *valuation*) if $\boldsymbol{v}(\mathbb{x}) \in D_{\mathrm{Attr}(\mathbb{x})}$, i.e., if each variable $\mathbb{x}$ is assigned a value $\boldsymbol{v}(\mathbb{x})$ from the domain of its attribute. For two $\mathbf{D}$-valuations $\boldsymbol{w}$ and $\boldsymbol{v}$ we write $\boldsymbol{w} =_{\mathbb{x}} \boldsymbol{v}$ whenever $\boldsymbol{w}(\mathbb{x}') = \boldsymbol{v}(\mathbb{x}')$ for each $\mathbb{x}'$ different from $\mathbb{x}$. In other words, $\boldsymbol{w}$ and $\boldsymbol{v}$ differ at most in the value which is assigned to $\mathbb{x}$. In addition, if p is a term (i.e., $p \in \mathrm{Var} \cup \mathrm{Con}$), we define a *value* $||p||_{\mathbf{D},\boldsymbol{v}}$ *of p in* $\mathbf{D}$ *under* $\boldsymbol{v}$ as follows: if $p \in \mathrm{Var}$ then $||p||_{\mathbf{D},\boldsymbol{v}} = \boldsymbol{v}(p)$; if $p \in \mathrm{Con}$ then $||p||_{\mathbf{D},\boldsymbol{v}} = f^{\mathbf{D}}(p)$.

Now, we can introduce degrees to which formulas of DRC are true in database instances:

Definition 3. Let $\mathbf{D} = \langle D, \approx^{\mathbf{D}}, \mathbb{R}^{\mathbf{D}}, f^{\mathbf{D}} \rangle$ be a database instance of database scheme $\langle \mathbb{R}, \varrho \rangle$ over $\{\langle D_y, \approx_y \rangle \,|\, y \in Y\}$. Furthermore, let φ be a formula of $\langle \mathbb{R}, \varrho \rangle$ and let $\boldsymbol{v} \colon \mathrm{Var} \to D$ be a valuation of variables. We define a *degree* $||\varphi||_{\mathbf{D},\boldsymbol{v}} \in L$ *to which φ is true in the database instance* $\mathbf{D}$ *under the valuation* $\boldsymbol{v}$ as follows:

- (i) if φ equals $\mathbb{r}(\mathbb{x}_1, \ldots, \mathbb{x}_n)$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = \mathbb{r}^{\mathbf{D}}(r)$ where $r \in \mathrm{Tupl}(\varrho(\mathbb{r}))$ is a tuple such that $r(\mathrm{Attr}(\mathbb{x}_i)) = \boldsymbol{v}(\mathbb{x}_i)$ for each $i = 1, \ldots, n$;
- (ii) if φ equals $p \approx q$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = ||p||_{\mathbf{D},\boldsymbol{v}} \approx^{\mathbf{D}} ||q||_{\mathbf{D},\boldsymbol{v}}$;
- (iii) if φ equals $\overline{a}$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = a$;
- (iv) If φ equals $\psi \otimes \chi$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = ||\psi||_{\mathbf{D},\boldsymbol{v}} \otimes ||\chi||_{\mathbf{D},\boldsymbol{v}}$; analogously for $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ and the corresponding truth functions $\wedge$, $\vee$, $\to$, $\leftrightarrow$;
- (v) if φ equals $\neg\psi$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = ||\psi||_{\mathbf{D},\boldsymbol{v}} \to 0$;
  if φ equals $\Delta\psi$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = \big(||\psi||_{\mathbf{D},\boldsymbol{v}}\big)^*$;
- (vi) if φ equals $(\forall \mathbb{x})\psi$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = \bigwedge_{\boldsymbol{w} =_{\mathbb{x}} \boldsymbol{v}} ||\psi||_{\mathbf{D},\boldsymbol{w}}$;
  if φ equals $(\exists \mathbb{x})\psi$ then $||\varphi||_{\mathbf{D},\boldsymbol{v}} = \bigvee_{\boldsymbol{w} =_{\mathbb{x}} \boldsymbol{v}} ||\psi||_{\mathbf{D},\boldsymbol{w}}$. ■

As in the ordinary first-order logic, one can show that $||\varphi||_{\mathbf{D},\boldsymbol{v}}$ does not depend of values $\boldsymbol{v}(\mathbb{x})$ assigned to variables $\mathbb{x}$ which do not have free occurrences in φ, see [3, 11].

Remark 3. (a) The notion $||\cdots||_{\mathbf{D},\boldsymbol{v}}$ of a degree to which a formula is true in $\mathbf{D}$ under $\boldsymbol{v}$ is naturally graded. That means, $||\cdots||_{\mathbf{D},\boldsymbol{v}}$ is a degree from L, not necessarily 0 (falsity) or 1 (full truth). For instance, if $||\varphi||_{\mathbf{D},\boldsymbol{v}} = 0.9$, we might say that φ is "almost true" in $\mathbf{D}$ and $\boldsymbol{v}$. From

the point of view of similarity-based databases, the degree $||\varphi||_{\mathbf{D},\boldsymbol{v}} = 0.9$ gives us a rank to which a tuple of values matches a query posed by φ. Recall that in the classic logic, φ is either true or not true in a structure (tuple either belongs to a data table of not) whereas in our model, φ can be true to degrees (which correspond to ranks).

(b) There is a clear connection of $||\cdots||_{\mathbf{D},\boldsymbol{v}}$ as we have introduced in Definition 3 and a notion of truth of formulas in first-order fuzzy structures as they appear in fuzzy logics in narrow sense, cf. [3, 10, 11]. Indeed, up to slight difference in formalization which is namely due to a specific nature of database systems (e.g., no order of attributes), $\mathbf{D}$ is a first-order fuzzy structure. This close connection to fuzzy logics is beneficial since it automatically gives us notions like semantic entailment and provability. Since formulas of DRC will represent queries in our model, we can explore, e.g., *entailment of queries* using well-established and explored notions from fuzzy logics [3, 10, 11]. Another strong point is that the relationship of our model and the first-order predicate fuzzy logic is the same as the relationship of the Codd's model of data and the classic first-order predicate logic. ■

Remark 4. Let us make a further note on the role of variables in atomic formulas of DRC. As it has been noted, atomic formulas $\mathbb{r}(\mathbb{x}_1, \ldots, \mathbb{x}_n)$ cannot contain constants for domain values which may seem odd. In fact, we have chosen this restriction to simplify the handling of variables and attributes. In our approach a relation scheme $\varrho(\mathbb{r})$ associated with $\mathbb{r}$ can be read directly from $\mathbb{r}(\mathbb{x}_1, \ldots, \mathbb{x}_n)$. Namely, $\varrho(\mathbb{r}) = \{\mathrm{Attr}(\mathbb{x}_1), \ldots, \mathrm{Attr}(\mathbb{x}_n)\}$. Thus, we are not forced to use formulas with variables containing attribute annotations such as $\mathbb{r}(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ which are used in classic DRCs [2]. This keeps our notation closer to that of first-order predicate logic. The price we pay is the inability to write constants directly into atomic formulas $\mathbb{r}(\mathbb{x}_1, \ldots, \mathbb{x}_n)$. On the other hand, we can write, e.g.,

$$\mathbb{r}(\mathbb{x}, \mathbb{z}) \otimes \Delta\mathbb{z} \approx c \quad \text{or} \quad (\exists \mathbb{z})(\mathbb{r}(\mathbb{x}, \mathbb{z}) \otimes \Delta\mathbb{z} \approx c)$$

To denote the fact that "$\mathbb{x}$ is related to value denoted c according to $\mathbb{r}$". Indeed, if $\approx^{\mathbf{D}}$ satisfies (Sep) and if * is globalization then $||(\exists \mathbb{z})(\mathbb{r}(\mathbb{x}, \mathbb{z}) \otimes \Delta\mathbb{z} \approx c)||_{\mathbf{D},\boldsymbol{v}} = \mathbb{r}^{\mathbf{D}}(\boldsymbol{v}(\mathbb{x}), f^{\mathbf{D}}(c))$.

Analogously, if $\varrho(\mathbb{r}) = \{y_1, y_2\}$ and $\varrho(\mathbb{s}) = \{y_3, y_4\}$, an expression $\mathbb{r}(\mathbb{x}, \mathbb{y}) \otimes \mathbb{s}(\mathbb{y}, \mathbb{z})$, which may represent a kind of a "join", is not a formula of DRC because $\mathbb{y}$ cannot be variable of two distinct attributes. In much the same way as in case of constants, we may use a formula

$$\mathbb{r}(\mathbb{x}, \mathbb{y}) \otimes (\exists \mathbb{y}')(\mathbb{r}(\mathbb{y}', \mathbb{z}) \otimes \Delta\mathbb{y} \approx \mathbb{y}')$$

which does the desired job and $\mathbb{x}, \mathbb{y}, \mathbb{z}$ are the only free variables in the formula. In fact, we may introduce a notation $\mathbb{r}(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ as in the ordinary DRCs and claim that each such expression is just an abbreviation for a formula

$$(\exists \mathbb{x}'_1) \cdots (\exists \mathbb{x}'_n)(\mathbb{r}(\mathbb{x}'_1, \ldots, \mathbb{x}'_n) \otimes \Delta(\mathbb{x}_1 \approx \mathbb{x}'_1 \otimes \cdots \otimes \mathbb{x}_n \approx \mathbb{x}'_n)),$$

where $\mathbb{x}'_1 \in \mathrm{Var}(y_1), \ldots, \mathbb{x}'_n \in \mathrm{Var}(y_n)$. ■

Ranked Data Tables Induced by Formulas of DRC

In this section we specify how formulas of DRC can be used as queries and what we mean by RDTs induced by formulas of DRC. Recall that in the classic DRC [2], one considers expressions of DRC of the form $\{y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n \,|\, \varphi\}$, where φ is a formula of DRC, $\mathbb{x}_1, \ldots, \mathbb{x}_n$ are variables, and $y_1, \ldots, y_n$ are pairwise disjoint attributes denoting a relation scheme. The sequence $y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n$ is called a target list. Tuples a data table which result by $\{y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n \,|\, \varphi\}$, consist of tuples t of values which make φ true in a database instance after a proper substitution of tuple values $t(y_i)$ for variables $\mathbb{x}_i$. We now introduce a way of inducing ranked data tables by formulas of DRC in our model:

Definition 4. Let $\mathbf{D}$ be a database instance of $\langle \mathbb{R}, \varrho \rangle$, let φ be a formula of $\langle \mathbb{R}, \varrho \rangle$ and let $y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n$ be a target list such that each y_i and $\mathrm{Attr}(\mathbb{x}_i)$ have the same domain with similarity in $\mathbf{D}$. Then, for each tuple t over $T = \{y_1, \ldots, y_n\}$, we define a *degree* $||\varphi||_{\mathbf{D},t(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$ *to which* φ *is true in* $\mathbf{D}$ *for tuple* t *and target list* $y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n$ by

$$||\varphi||_{\mathbf{D},t(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)} = \bigvee_{\boldsymbol{v} \in \mathbf{V}} ||\varphi||_{\mathbf{D},\boldsymbol{v}},$$

where $\mathbf{V}$ denotes a set of $\mathbf{D}$-valuations

$$\mathbf{V} = \{\boldsymbol{v} \colon \mathrm{Var} \to D \,|\, \boldsymbol{v}(\mathbb{x}_i) = t(y_i) \text{ for } i = 1, \ldots, n\}.$$

Moreover, define an RDT $\mathcal{D}^{\mathbf{D}}_{\varphi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$ on T by

$$\big(\mathcal{D}^{\mathbf{D}}_{\varphi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}\big)(t) = ||\varphi||_{\mathbf{D},t(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)},$$

for each tuple $t \in \mathrm{Tupl}(T)$. $\mathcal{D}^{\mathbf{D}}_{\varphi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$ is called a *result of query* $\varphi(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ *in database instance* $\mathbf{D}$. ■

Domain relational calculus should be seen as the *primary query system* in our model because it is directly derived from interpretation of first-order formulas according to common rules of predicate fuzzy logics. Each query is given by a formula and a target list. The execution of the query in a database instance is defined, again, according to common rules of evaluation of formulas in predicate fuzzy logics and the execution yields a new RDT. In addition to the logical clarity of our approach, the query system based on our DRC can be easily implemented into a *QUEL-like query language.* This issue is, however, outside the scope of this paper.

THEOREM 2. *Let* $\langle \mathbb{R}, \varrho \rangle$ *be a database scheme. For each query* $\varphi(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ *there is a query* $\psi(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ *so that the following are satisfied*

(i) $\mathrm{free}(\psi) = \{\mathbb{x}_1, \ldots, \mathbb{x}_n\}$, *and*

(ii) $\mathcal{D}^{\mathbf{D}}_{\varphi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)} = \mathcal{D}^{\mathbf{D}}_{\psi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$

is true for each instance $\mathbf{D}$ *of* $\langle \mathbb{R}, \varrho \rangle$.

PROOF SKETCH. It can be shown that ψ is a formula of the form $(\exists \mathbb{x}'_1) \cdots (\exists \mathbb{x}'_k)\varphi \otimes (\mathbb{z}_1 \approx \mathbb{z}_1 \otimes (\mathbb{z}_2 \approx \mathbb{z}_2 \otimes \cdots))$, where $\mathbb{x}'_i$ are free in φ but are not in its target list and $\mathbb{z}_i$ appear in the target list but do not appear free in φ. □

The following important assertion says that the domain relational calculus in our model is a generalization of the classic one in sense that if we use the classic two-valued structure of truth degrees in place of $\mathbf{L}$, our DRC is as expressive as the classic DRC:

THEOREM 3. *If* $\mathbf{L}$ *is a two-valued Boolean algebra then for each query* $\varphi(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ *there is an ordinary DRC expression* $\{y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n \,|\, \psi\}$ *whose value in* $\mathbf{D}$ *is* $\mathcal{D}^{\mathbf{D}}_{\varphi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$ *provided that* $\approx^{\mathbf{D}}$ *is identity on* D.

PROOF SKETCH. Consequence of the fact that $L = \{0, 1\}$ and all logical connectives have the same interpretation as the classic ones. Moreover, one can take ψ which results from φ by omitting all Δ's and replacing the constants $\overline{0}$ and $\overline{1}$ for truth degrees by any formulas of the form $\neg(\vartheta \Rightarrow \vartheta)$ and $\vartheta \Rightarrow \vartheta$, respectively. The rest is routine to check. □

Examples of Queries and Their Results

In the rest of this section we demonstrate how DRC can be used as a query language which is the main motivation for developing DRC in our model. We consider a database instance $\mathbf{D}$ containing two RDTs: a ranked data table $\mathbb{s}^{\mathbf{D}}$ of "houses for sale" from Section 1 and an RDT $\mathbb{r}^{\mathbf{D}}$ of "customers" which will be described later.

First, consider a DRC formula $\mathbb{s}(\mathbb{x}_1, \mathbb{x}_2, \mathbb{x}_3, \mathbb{x}_4) \otimes \mathbb{x}_3 \approx 3$ and a target list $\mathit{id}{:}\mathbb{x}_1, \mathit{price}{:}\mathbb{x}_2, \mathit{bdrm}{:}\mathbb{x}_3, \mathit{type}{:}\mathbb{x}_4$. The formula and the target list represent a query: "select tuples of attributes *id*, *price*, *bdrm*, and *type* from $\mathbb{s}^{\mathbf{D}}$ such that the number of bedrooms (the value of attribte *bdrm*) is similar to 3". The result of the query in $\mathbf{D}$ is

	id	*price*	*bdrm*	*type*
0.78	62	$195,000	2	Single Family
0.65	14	$230,000	2	Penthouse
0.33	23	$145,000	1	Log Cabin
0.20	59	$320,000	4	Ranch

The best match has rank 0.78. If the condition "number of bedrooms being similar to 3" appears too strict, we can relax the similarity condition. For instance, we can use the following formula $\mathbb{s}(\mathbb{x}_1, \mathbb{x}_2, \mathbb{x}_3, \mathbb{x}_4) \otimes (\overline{0.6} \Rightarrow \mathbb{x}_3 \approx 3)$ saying that "number of bedrooms is similar to 3 at least to degree 0.6". For the same target list as before, the query produces the following result:

	id	*price*	*bdrm*	*type*
1.00	14	$230,000	2	Penthouse
1.00	62	$195,000	2	Single Family
0.73	23	$145,000	1	Log Cabin
0.60	59	$320,000	4	Ranch

which says there are two exact matches, cf. the previous result. In order to consider "joins", we need to provide another table. Let $\mathbb{r}^{\mathbf{D}}$ be the following table of customers (persons willing to buy a house):

	name	*bdrm*	*budget*
1.00	Adams	3	$250,000
1.00	Black	4	$325,000
1.00	Chang	1	$300,000
1.00	Davis	2	$200,000

The previous RDT have all ranks equal to 1 because it is a table of customers prior to any querying. The attributes represent names of customers, number of bedrooms they require, and their budget (price willing to spend on a house). We can assume that *budget* has the same domains with similarity as *price*. A general join in our setting can be expressed as follows:

$$(\exists \mathbb{x}_3)(\exists \mathbb{x}_6)(\exists \mathbb{x}_7)(\mathbb{s}(\mathbb{x}_1, \mathbb{x}_2, \mathbb{x}_3, \mathbb{x}_4) \otimes \mathbb{r}(\mathbb{x}_5, \mathbb{x}_6, \mathbb{x}_7) \otimes \vartheta), \quad (2)$$

where ϑ is a formula representing an additional condition for the join. For instance, if ϑ equals $\mathbb{x}_3 \approx \mathbb{x}_6 \otimes \mathbb{x}_2 \approx \mathbb{x}_7$ and if the target list equals $\mathit{id}{:}\mathbb{x}_1, \mathit{name}{:}\mathbb{x}_5, \mathit{price}{:}\mathbb{x}_2, \mathit{type}{:}\mathbb{x}_4$ then (2) represents a query "select *id* of a house h (sold for approximatelly $200,000), *name* of a customer c, *price* of h, and *type* of h such that the number of berdooms of h is similar to the number of bedrooms required by c and the price of h is similar to the budget of c". Note that this query which involves similarity is more natural than a corresponding classic query involving strict equality because customers are not interested in the exact price but all prices which are suitably close to their budget. The result of the query is

	id	*name*	*price*	*type*
0.96	62	Davis	$195,000	Single Family
0.70	14	Davis	$230,000	Penthouse
0.55	14	Adams	$230,000	Penthouse
0.51	62	Adams	$195,000	Single Family
0.38	59	Black	$320,000	Ranch
0.30	14	Chang	$230,000	Penthouse
0.26	23	Davis	$145,000	Log Cabin
0.26	62	Chang	$195,000	Single Family

If we wish to display just tuples with ranks above 0.5, we can use a formula $\varphi \otimes \Delta(\overline{0.5} \Rightarrow \varphi)$, where φ is (2) and the target list remains as before. If Δ is interpreted by * which is globalization (1), we get:

	id	*name*	*price*	*type*
0.96	62	Davis	$195,000	Single Family
0.70	14	Davis	$230,000	Penthouse
0.55	14	Adams	$230,000	Penthouse
0.51	62	Adams	$195,000	Single Family

The DRC in our model is flexible and sensitive because we are allowed to put more/less emphasis on particular conditions in joins. For instance, if we wish to formalize a query "select *id*, *name*, *price*, and *type* such that the numbers of berdooms are similar to degree at lest 0.8 and the budget is similar to price at least to degree 0.7" then we can do it by using ϑ being $(\overline{0.8} \Rightarrow \mathbb{x}_3 \approx \mathbb{x}_6) \otimes (\overline{0.7} \Rightarrow \mathbb{x}_2 \approx \mathbb{x}_7)$ in (2). In addition to that, if we wish to display only tuples with ranks above 0.8, we can use $\varphi \otimes \Delta(\overline{0.8} \Rightarrow \varphi)$ where φ is (2). Altogether, we get a result

	id	*name*	*price*	*type*
0.98	62	Adams	$195,000	Single Family
0.98	62	Davis	$195,000	Single Family
0.85	14	Adams	$230,000	Penthouse
0.85	14	Davis	$230,000	Penthouse
0.80	14	Chang	$230,000	Penthouse

This short list of examples is far from being comprehensive. Because of the limited scope of the paper, we have shown only a fragment of possible "similarity-based selections" and "similarity-based joins" which can be formalized by DRC formulas in our model. Nevertheless, it should be apparent that DRC has a rich semantics and allows us to deal with nontrivial similarity-based issues and express ranked data tables as result of queries.

5. RELATIONAL ALGEBRA

It might be tempting to characterize the expressive power of the domain relational calculus introduced in our model in terms of elementary algebraic operations with RDTs over domains with similarities. This possibility will be explored in the rest of the paper. We begin by introducing operations of relational algebra in our model. Further description with examples can be found in [5].

Counterparts of Boolean Operations

The first group of relational operations contains our counterparts to the basic Boolean operations of Codd's model such as union and intersection. The operations emerge in our model as we replace the two-element Boolean algebra by a complete residuated lattice $\mathbf{L}$. For RDTs $\mathcal{D}_1$ and $\mathcal{D}_2$ on T, we put

$$(\mathcal{D}_1 \cap \mathcal{D}_2)(t) = \mathcal{D}_1(t) \wedge \mathcal{D}_2(t), \quad (3)$$

$$(\mathcal{D}_1 \cup \mathcal{D}_2)(t) = \mathcal{D}_1(t) \vee \mathcal{D}_2(t), \quad (4)$$

$$(\mathcal{D}_1 \otimes \mathcal{D}_2)(t) = \mathcal{D}_1(t) \otimes \mathcal{D}_2(t), \quad (5)$$

$$(\mathcal{D}_1 \to \mathcal{D}_2)(t) = \mathcal{D}_1(t) \to \mathcal{D}_2(t), \quad (6)$$

for each $t \in \mathrm{Tupl}(T)$. The resulting data tables $\mathcal{D}_1 \cap \mathcal{D}_2, \ldots, \mathcal{D}_1 \to \mathcal{D}_2$ are RDTs on T; $\mathcal{D}_1 \cap \mathcal{D}_2$ and $\mathcal{D}_1 \otimes \mathcal{D}_2$ are called the $\wedge$-*intersection* and $\otimes$-*intersection*, respectively; $\mathcal{D}_1 \cup \mathcal{D}_2$ is called the *union*; $\mathcal{D}_1 \to \mathcal{D}_2$ is called the *residuum* of RDTs $\mathcal{D}_1$ and $\mathcal{D}_2$.

Remark 5. Observe that the operation of residuum may produce an "infinite ranked data table" given two finite RDTs provided that at least one attribute has an infinite domain. This can be avoided using active domains, see [14]. Since for simplicity we do not deal with *domain independence* [2] in this paper, we allow "infinite RDTs" as results of the residuum operation performed with RDTs. ■

In addition to the previous operations, we may define a biresiduum $\mathcal{D}_1 \leftrightarrow \mathcal{D}_2$ of RDTs, a residuated negation $\neg\mathcal{D}$ of an RDT, an RDT $\mathcal{D}^*$ which results by applying hedge *, an a-shift $a{\to}\mathcal{D}$, and an a-multiple $a{\otimes}\mathcal{D}$ of an RDT:

$$(\mathcal{D}_1 \leftrightarrow \mathcal{D}_2)(t) = \mathcal{D}_1(t) \leftrightarrow \mathcal{D}_2(t), \tag{7}$$

$$(\neg\mathcal{D})(t) = \mathcal{D}(t) \to 0, \tag{8}$$

$$(\mathcal{D}^*)(t) = \mathcal{D}(t)^*, \tag{9}$$

$$(a{\to}\mathcal{D})(t) = a \to \mathcal{D}(t), \tag{10}$$

$$(a{\otimes}\mathcal{D})(t) = a \otimes \mathcal{D}(t). \tag{11}$$

Notice that $\mathcal{D}_1 \leftrightarrow \mathcal{D}_2$ and $\neg\mathcal{D}$ are derived operations because $\mathcal{D}_1 \leftrightarrow \mathcal{D}_2 = (\mathcal{D}_1 \to \mathcal{D}_2) \cap (\mathcal{D}_2 \to \mathcal{D}_1)$ and $\neg\mathcal{D} = \mathcal{D} \to \mathcal{D}_\emptyset$ where $\mathcal{D}_\emptyset$ is an empty RDT, i.e., $\mathcal{D}_\emptyset(t) = 0$ for all $t \in \mathrm{Tupl}(T)$. Obviously, if $\mathbf{L}$ is the two-element Boolean algebra then $\mathcal{D}^*$ equals $\mathcal{D}$. If $\mathcal{D}$ is a result of query Q then $(a{\to}\mathcal{D})(t)$ is a "degree to which t matches query Q at least to degree a", cf. [3, 11].

Cartesian Product and Projection

For RDTs $\mathcal{D}_1$ and $\mathcal{D}_2$ on disjoint relation schemes S and T we define a RDT $\mathcal{D}_1 \times \mathcal{D}_2$ on $S \cup T$, called a *Cartesian product of* $\mathcal{D}_1$ *and* $\mathcal{D}_2$, by

$$(\mathcal{D}_1 \times \mathcal{D}_2)(st) = \mathcal{D}_1(s) \otimes \mathcal{D}_2(t). \tag{12}$$

If $\mathcal{D}_1$ and $\mathcal{D}_2$ are results of queries Q_1 and Q_2, respectively, the rank of st in $\mathcal{D}_1 \times \mathcal{D}_2$ is a degree to which "s matches Q_1 and t matches Q_2". Obviously, if $\mathbf{L}$ is the two-element Boolean algebra, $\mathcal{D}_1 \times \mathcal{D}_2$ becomes the ordinary Cartesian product of relations which equals to $\mathcal{D}_1 \bowtie \mathcal{D}_2$.

Projection produces an RDT with tuples containing a subset of attributes. Since $\mathcal{D}$ can contain tuples which agree on the same subset of attributes but have various ranks, the rank of t in the projection of $\mathcal{D}$ onto A will be computed as a supremum of ranks of all tuples in $\mathcal{D}$ which agree with t on the attributes from A. Therefore, if $\mathcal{D}$ is an RDT on T, the *projection* $\pi_R(\mathcal{D})$ *of* $\mathcal{D}$ *onto* $R \subseteq T$ is defined by

$$(\pi_R(\mathcal{D}))(r) = \bigvee_{s \in \mathrm{Tupl}(T \setminus R)} \mathcal{D}(rs), \tag{13}$$

for each $r \in \mathrm{Tupl}(R)$. Again, if $\mathbf{L}$ is a two-valued Boolean algebra, $\pi_R(\mathcal{D})$ becomes the ordinary projection.

Similarity-Based Selection

The similarity-based selection introduced in this paragraph is a counterpart to ordinary selection which selects from a data table all tuples which approximately match a given condition: Let $\mathcal{D}$ be an RDT on T and let $y \in T$ and $d \in D_y$. Then, a *similarity-based selection* $\sigma_{y \approx d}(\mathcal{D})$ *of tuples in* $\mathcal{D}$ *matching* $y \approx d$ is defined by

$$\big(\sigma_{y \approx d}(\mathcal{D})\big)(t) = \mathcal{D}(t) \otimes t(y) \approx_y d. \tag{14}$$

Considering $\mathcal{D}$ as a result of query Q, the rank of t in $\sigma_{y \approx d}(\mathcal{D})$ can be interpreted as a degree to which "t matches the query Q and the y-value of t is similar to d". Hence, if $\mathcal{D}$ is a table where all rows have ranks 1, $(\sigma_{y \approx d}(\mathcal{D}))(t) = t(y) \approx_y d$, i.e. it is a degree to which "the y-value of t is similar to d". Obviously, (14) can be generalized to $\sigma_{p \approx q}(\mathcal{D})$ where p and q are either attributes (with a common domain), an attribute and a value (from its domain), or two values from the same domain.

Notice that using Cartesian products and similarity-based selections, we can introduce similarity-based θ-joins such as

$$\mathcal{D}_1 \bowtie_{p \approx q} \mathcal{D}_2 = \sigma_{p \approx q}(\mathcal{D}_1 \times \mathcal{D}_2). \tag{15}$$

Therefore, various types of joins can be considered derived relational operations in our model, see Section 6.

Division

The relational operation of division is introduced for technical reasons and its purpose will become apparent when considering relational completeness in Section 6.

Let $\mathcal{D}_1$ be an RDT on R and let $\mathcal{D}_2$ be an RDT on $S \subseteq R$. Then, a *division* $\mathcal{D}_1 \div \mathcal{D}_2$ *of* $\mathcal{D}_1$ *by* $\mathcal{D}_2$ *is an RDT on* $T = R \setminus S$, where

$$(\mathcal{D}_1 \div \mathcal{D}_2)(t) = \textstyle\bigwedge_{s \in \mathrm{Tupl}(S)} \big(\mathcal{D}_2(s) \to \mathcal{D}_1(st)\big),$$

for each $t \in \mathrm{Tupl}(t)$. In words, $(\mathcal{D}_1 \div \mathcal{D}_2)(t)$ is the degree to which the following proposition holds: "For each tuple s in $\mathcal{D}_2$, the concatenation st is in $\mathcal{D}_1$".

Relational Algebra Expressions

Algebraic operations can be consecutively applied for various RDTs so that we get new RDTs as results.

Consider a database scheme $\langle \mathbb{R}, \varrho \rangle$ on Y. A *relational algebra expression of* $\langle \mathbb{R}, \varrho \rangle$ (shortly, a *RA-expression*) is any expression formed legally from the relation symbols in $\mathbb{R}$ and symbols $\mathbb{d}_y$ ($y \in Y$) for constant relations, using (symbols for) relational operations $\cup$, $\cap$, $\otimes$, $\to$, *, $\times$, π_R, $\sigma_{p \approx q}$, $\div$, and ρ. Relation symbols and symbols for constant relations are used in place of RDTs. Relational algebra expressions shall be denoted $E, E', E_1, E_2, \ldots$

Remark 6. The definition of RA-expressions may be further formalized by precisely specifying what we mean by an expression "formed legally". We are not going into this because of the limited scope of this paper. Nevertheless, it should be clear, what we mean by legal expressions. For instance if $\mathbb{r} \in \mathbb{R}$ and $\mathbb{s} \in \mathbb{R}$ then $\mathbb{r} \times \mathbb{s}$ denoting a Cartesian product of ranked data tables denoted by relation symbols $\mathbb{r}$ and $\mathbb{s}$ is legal iff $\varrho(\mathbb{r}) \cap \varrho(\mathbb{s}) = \emptyset$, i.e., iff the relation schemes associated to symbols $\mathbb{r}$ and $\mathbb{s}$ are disjoint. Moreover, $\pi_A(\mathbb{r})$ is legal iff $A \subseteq \varrho(\mathbb{r})$, etc. In case of similarity-based selections, we assume that attribute names and constants for domain values may appear as qualifiers of the selection. For instance, $\sigma_{y \approx c}(\mathbb{d}_y)$ is a legal expression denoting a similarity-based selection of values from domain of attribute y which are similar to c. ■

Each relational algebra expression of $\langle \mathbb{R}, \varrho \rangle$ can be interpreted in a database instance of that database scheme. Namely, let $\mathbf{D} = \langle D, \approx^{\mathbf{D}}, \mathbb{R}^{\mathbf{D}}, f^{\mathbf{D}} \rangle$ be an instance of $\langle \mathbb{R}, \varrho \rangle$ over $\{\langle D_y, \approx_y \rangle \mid y \in Y\}$ and let E be a relation algebra expression of $\langle \mathbb{R}, \varrho \rangle$. Then $E(\mathbf{D})$ denotes the RDT which results by substituting RDTs $\mathbb{r}^{\mathbf{D}}, \mathbb{s}^{\mathbf{D}}, \ldots$ for relation symbols $\mathbb{r}, \mathbb{s}, \ldots$ and substituting values $f^{\mathbf{D}}(c), f^{\mathbf{D}}(d), \ldots$ for constants $c, d, \ldots$ which appear in E and evaluating the expres-

sion. $E(\mathbf{D})$ shall be called the *result of E in* $\mathbf{D}$.

6. RELATIONAL COMPLETENESS

Suppose that $\mathbf{L}$ is a complete residuated lattice with * being the globalization and that each similarity satisfies (Sep). Moreover, for each attribute $y \in Y$ we consider an additional relation symbol $\mathbb{d}_y$ which represents a ranked data table containing all elements in the domain of attribute y. Therefore, given a database instance $\mathbf{D}$, $\mathbb{d}_y$ is interpreted by an RDT $\mathbb{d}_y^{\mathbf{D}}$ on $\{y\}$ over $\{\langle D_y, \approx_y\rangle\}$ such that $\mathbb{d}_y^{\mathbf{D}}(t) = 1$ for all $t \in \mathrm{Tupl}(\{y\})$. Furthermore, for $R = \{y_1, \ldots, y_k\} \subseteq Y$ we let $\mathbb{d}_R$ denote the Cartesian product $\mathbb{d}_{y_1} \times \cdots \times \mathbb{d}_{y_k}$. Now, we can prove the following

THEOREM 4. *For each $\varphi(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ there is a RA-expression E_φ such that $E_\varphi(\mathbf{D}) = \mathcal{D}^{\mathbf{D}}_{\varphi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$.*

PROOF SKETCH. The claim is proved by structural induction over φ. First, let us introduce the following notation: $T = \{y_1, \ldots, y_n\}$; $X = \{\mathbb{x}_1, \ldots, \mathbb{x}_n\}$. If φ equals $\overline{a}$ and $n = 0$ (empty target list) then $E_\varphi = \pi_\emptyset(\mathbb{r})$ where $\mathbb{r}$ is any relation symbol. If the target list is nonempty, one can take $E_\varphi = a \otimes \mathbb{d}_T$. If φ equals $\mathbb{r}(\mathbb{z}_1, \ldots, \mathbb{z}_k)$ then $E_\varphi = \pi_T(\mathbb{r} \times \mathbb{d}_Z)$ where $Z = \{\mathrm{Attr}(\mathbb{x}) \mid \mathbb{x} \in X \text{ and } \mathbb{x} \notin \{\mathbb{z}_1, \ldots, \mathbb{z}_k\}\}$. If φ is $\mathbb{x} \approx c$, we distinguish two situations: if $\mathbb{x} \notin X$ then we put $E_\varphi = \mathbb{d}_T$; if $\mathbb{x} \in X$ then we put $E_\varphi = \sigma_{\mathrm{Attr}(\mathbb{x})\approx c}(\mathbb{d}_T)$. If φ is $(\psi \otimes \chi)$ then we put $E_\varphi = E_\psi \otimes E_\chi$ where, by induction, E_ψ and E_χ are RA-expressions such as $E_\psi(\mathbf{D}) = \mathcal{D}^{\mathbf{D}}_{\psi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$ and $E_\chi(\mathbf{D}) = \mathcal{D}^{\mathbf{D}}_{\chi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$. In a similar way, one can proceed for the other connectives. If φ equals $(\exists \mathbb{x})\psi$ then, by induction, for $\psi(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n, y{:}\mathbb{x})$ where $y \notin T$ there is E_ψ such that $E_\psi(\mathbf{D}) = \mathcal{D}^{\mathbf{D}}_{\psi(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n,y:\mathbb{x})}$. Therefore,

$$||(\exists \mathbb{x})\psi||_{\mathbf{D},t(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)} = \bigvee_{s\in \mathrm{Tupl}(\{y\})} ||\psi||_{\mathbf{D},st(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n,y:\mathbb{x})} = \bigvee_{s\in \mathrm{Tupl}(\{y\})} (E_\psi(\mathbf{D}))(st) = (\pi_T(E_\psi(\mathbf{D})))(t)$$

is true for each $t \in \mathrm{Tupl}(T)$. Analogously, one can prove that if φ is $(\forall \mathbb{x})\psi$ then $E_\psi \div \mathbb{d}_{\mathrm{Attr}(\mathbb{x})}$ does the job. □

THEOREM 5. *For each RA-expression E there is φ_E and $y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n$ such that $E(\mathbf{D}) = \mathcal{D}^{\mathbf{D}}_{\varphi_E(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$.*

PROOF SKETCH. The proof goes as in the ordinary case. For example, if $E = \sigma_{y\approx c}(E')$ then, by induction, there is $\varphi_{E'}(y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n)$ where $E'(\mathbf{D}) = \mathcal{D}^{\mathbf{D}}_{\varphi_{E'}(y_1:\mathbb{x}_1,\ldots,y_n:\mathbb{x}_n)}$. Therefore, we let φ_E be $\varphi_{E'} \otimes \mathbb{x}_i \approx c$ where $y_i = y$ and the target list remains $y_1{:}\mathbb{x}_1, \ldots, y_n{:}\mathbb{x}_n$. The remaining operations are handled in a similar way. □

COROLLARY 6. *Domain relational calculus and relational algebra in our model have the same expressive power.*

PROOF. Consequence of Theorem 5 and Theorem 4. □

Remark 7. (a) Let us stress that we have not discussed domain independence. Hence, our DRC and RA yield in general infinite ranked data tables. This issue can be overcome by introducing ranges like in the ordinary case [2].

(b) In addition to DRC, we can introduce a tuple relational calculus (TRC) which is equally expressive. This can be shown by translating formulas of DRC to equivalent formulas of TRC and *vice versa*. Details are postponed to the full version of this paper. ■

Notice that unlike the ordinary case, universal and existential quantifiers are independent. As a matter or fact, $||(\forall x)\varphi||_{\mathbf{D},v}$ may not be equal to $||\neg(\exists x)\neg\varphi||_{\mathbf{D},v}$. On the other hand, if $\mathbf{L}$ satisfies additional conditions, we can prove that the universal quantifier can be expressed using the existential one (and vice versa) which leads to a simplified set of the base operations of our relational algebra:

THEOREM 7. *Let $\mathbf{L}$ be a complete MV-algebra, i.e., a complete residuated lattice satisfying $a \vee b = (a \to b) \to b$. Then, DRC is as expressive as RA which uses residuum, globalization, projection, similarity-based selection, Cartesian product, and a-multiple as the only operations.*

PROOF. The claim follows from the fact that MV-algebras satisfy the law of double negation and that $\otimes$, $\wedge$, $\vee$ are expressible using $\to$ and 0 (falsity). Namely, $a \otimes b = (a \to (b \to 0)) \to 0$, and $a \wedge b = a \otimes (a \to b)$, see [3, 11]. □

7. ACKNOWLEDGMENTS

Supported by research plan MSM 6198959214.

8. REFERENCES

[1] S. Abiteboul *et al.* The Lowell database research self-assessment. *Comm. ACM* **48**(5)(2005), 111–118.

[2] P. Atzeni, V. De Antonellis: *Relational Database Theory.* Benjamin/Cummings, Redwood City (CA), 1993.

[3] Belohlavek R.: *Fuzzy Relational Systems: Foundations and Principles.* Kluwer, Academic/Plenum Publishers, New York, 2002.

[4] R. Belohlavek and V. Vychodil. Data tables with similarity relations: functional dependencies, complete rules and non-redundant bases. *DASFAA 2006*, LNCS 3882:644–658, 2006.

[5] R. Belohlavek, V. Vychodil: Logical foundations for similarity-based databases. *DASFAA 2009 Workshops*, LNCS 5667:137–151, 2009.

[6] R. Belohlavek and V. Vychodil. Codd's relational model from the point of view of fuzzy logic. *J. Logic and Computation* (to appear, doi: 10.1093/logcom/exp056).

[7] P. Bosc, D. Kraft, and F. Petry. Fuzzy sets in database and information systems: status and opportunities. *Fuzzy Sets and Syst.* 156:418–426, 2005.

[8] B. P. Buckles and F. E. Petry. Fuzzy databases in the new era. ACM SAC 1995, pages 497–502, Nashville, TN, 1995.

[9] R. Fagin. Combining fuzzy information: an overview. *ACM SIGMOD Record* 31(2):109–118, 2002.

[10] S. Gottwald. Mathematical fuzzy logics. *Bulletin for Symbolic Logic* Vol. **14**, No. 2, June 2008, 210–239.

[11] P. Hájek. *Metamathematics of Fuzzy Logic.* Kluwer, Dordrecht, 1998.

[12] P. Hájek. On very true. *Fuzzy Sets and Syst.* 124:329–333, 2001.

[13] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. RankSQL: Query Algebra and Optimization for Relational top-k queries. ACM SIGMOD 2005, pp. 131–142.

[14] D. Maier. *The Theory of Relational Databases.* Computer Science Press, Rockville, 1983.

[15] H. Prade and C. Testemale. Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Inf. Sci.* 34:115–143, 1984.

[16] K. V. S. V. N. Raju, and A. K. Majumdar. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Trans. Database Systems* Vol. 13, No. 2:129–166, 1988.

[17] Y. Takahashi. Fuzzy database query languages and their relational completeness theorem. *IEEE Trans. Knowledge and Data Engineering* 5:122–125, February 1993.