Relational similarity-based model of data part 1: foundations and query systems

Radim Belohlavek & Vilem Vychodil


To link to this article: https://doi.org/10.1080/03081079.2017.1357550
Relational similarity-based model of data part 1: foundations and query systems

Radim Belohlavek and Vilem Vychodil

Department of Computer Science, Palacky University, Olomouc, Czech Republic

ABSTRACT
We present a general rank-aware model of data which supports handling of similarity in relational databases. The model is based on the assumption that in many cases it is desirable to replace equalities on values in data tables by similarity relations expressing degrees to which the values are similar. In this context, we study various phenomena which emerge in the model, including similarity-based queries and similarity-based data dependencies. Central notion in our model is that of a ranked data table over domains with similarities which is our counterpart to the notion of relation on relation scheme from the classical relational model. Compared to other approaches which cover related problems, we do not propose a similarity-based or ranking module on top of the classical relational model. Instead, we generalize the very core of the model by replacing the classical, two-valued logic upon which the classical model is built by a more general logic involving a scale of truth degrees that, in addition to the classical truth degrees 0 and 1, contains intermediate truth degrees. While the classical truth degrees 0 and 1 represent nonequality and equality of values, and subsequently mismatch and match of queries, the intermediate truth degrees in the new model represent similarity of values and partial match of queries. Moreover, the truth functions of many-valued logical connectives in the new model serve to aggregate degrees of similarity. The presented approach is conceptually clean, logically sound, and retains most properties of the classical model while enabling us to employ new types of queries and data dependencies. Most importantly, similarity is not handled in an ad hoc way or by putting a “similarity module” atop the classical model in our approach. Rather, it is consistently viewed as a notion that generalizes and replaces equality in the very core of the relational model. We present fundamentals of the formal relational model and two equivalent query systems which are analogues of the classical relational algebra and domain relational calculus with range declarations. In the sequel to this paper, we deal with similarity-based dependencies.

1. Introduction
The success of the classic relational model of data (RM) whose investigation started with the influential paper by Codd (1970) is often attributed to its solid foundations based on

Onthecontrary,ifasubsetofpositiveintegersisusedtorepresentproductprices,thesituationchangesbecausedatabaseusersmaywanttoqueryforproductsprices\textit{similar}toagivenvalueinsteadofqueryingforproductspricesexactlymatchingaspecifiedvalue. Thisistypicalofdomainsthwosevaluesarenaturallybeingcomparedbyusersaccordingtotheirsimilarity. Forsuchdomains,similarity-basedqueries, ratherthan theirordinaryequality-basedcounterparts, areneeded.

Therearetwobasicapproachestoevaluatequeriesinvolvingconditionsbasedonsimilarityofelements:

First, a similarity-based query can be reduced to an ordinary range query whose result includes all data satisfying a given similarity-based condition up to a given tolerance level. Typically, metric approaches in databases, which can be seen as approaches to similarity via distance, consider all data with distance less than a prescribed threshold $\epsilon > 0$. The basic issue of this approach is obvious: $\epsilon$ is an additional input parameter whose selection is a matter of an arbitrary decision of a user. For a particular value of $\epsilon$, there may be interesting data whose distance is just slightly above $\epsilon$ and the data do not appear in the query result.

Second, we may accept the concept of \textit{partial} or \textit{imperfect matches}. That is, we depart from the yes/no matches and introduce degrees to which data match similarity-based conditions. Consequently, we need the tuples in relations to be annotated by suitable ranks (or scores), which are computed based on the similarity-based conditions and indicate the degrees to which tuples match queries. In this setting, the result of a similarity-based query contains all possible tuples annotated by such ranks.

We argue that the second approach is more suitable than the first because the results of queries can be shown to users sorted by the ranks with the best matches coming first with the possibility to stop listing (or computing) the results once the users have enough information, resolving thus the issue of the input tolerance level $\epsilon$. The most important facet of our model is its conceptual implementation of the second approach. Evidently, the approach can be implemented in various ways. One way is to propose an \textit{extension} of the classical model. That is, one can propose a similarity-based ranking module on top of the classical relational model. For some particular purposes, such a solution can be sufficient. In fact, present approaches which tackle the problems of ranking and similarity-based reasoning go along with these lines. What we find limiting in the approaches is that they focus mostly on isolated extensions of the classical model, typically rank-aware extensions of relational algebra which is used as the primary query system. It is by no means obvious
whether or how one can consider in such extensions other important issues like similarity-based dependencies in data and constraints. Also, such extensions are still based on the bivalent nature of the underlying model where only yes/no matches are allowed, i.e. the concept of partial matches has to be mimicked in the bivalent environment which we argue is cumbersome and logically not clean a solution.

We take a different approach and implement the concepts of similarity-based reasoning by generalization of the classical relational model. Thus, unlike the extension-based approaches, we do not build on top of the existing model, but we generalize the model so that the phenomena regarding similarity appear directly in the generalized model. From the theoretical viewpoint, we change the underlying logic of the relational model. In particular, we replace the classical predicate logic with a logic based on general residuated structures of degrees. We argue as follows. In the classical model, the key role is played by the two-element Boolean algebra which consists of 0 (falsity) and 1 (truth) and is implicitly used in the backbone of the relational model – the classical predicate logic. Then, the classical relations on relation schemes can be viewed as relations ranked by the values 0 and 1, only the ranks do not appear explicitly. Indeed, 1 stands for a match (each tuple in the relation has rank 1) and 0 stands for no match (tuples that do not belong to the relation have ranks 0). Analogously, each domain can be seen as equipped with equality which to any two values from the domain assigns 1 if the values are identical and 0 otherwise. Therefore, in addition to the interpretation of 0 and 1 as truth values, 0 and 1 can be interpreted as two borderline ranks or similarity degrees. Our generalization thus consists in replacing the two-element Boolean algebra by a more general structure of degrees, bounded by 0 and 1, and in general containing intermediate degrees 0 < a < 1 to express partial matches and similarity degrees. The classical model then becomes one particular instance of the generalized model when the structure of degrees becomes the two-element Boolean algebra. We show in the paper and in its sequel that such an approach is vital and is not limited just to querying.

A key step in developing the generalized model is the choice of the structures of degrees to replace the two-element Boolean algebra. We chose the class of (complete) residuated lattices for this purpose. Residuated lattices are certain generalizations of Boolean algebras and are utilized as structures of truth degrees in modern fuzzy logics (Gottwald 2008; Hájek 1998). As a consequence, our model has the same relationship to fuzzy predicate logic as the ordinary relational model has to the classical predicate logic. Indeed, we find the fact that there is a reasonably strong logic in the background of the model extremely important, basically for the same reasons as in case of the classical relational model. It allows us to consider, e.g. query systems based on evaluation of formulas (like analogues of domain and tuple calculi) and compare their expressive power with systems based on algebraic operations. Moreover, we can borrow the notions of syntactic and semantic entailment from the established logic and study, e.g. partial entailment of dependencies formalized in our model. Neither of these two topics seems to be discussed in the present literature (from the similarity and ranking point of view). In addition to the logical clarity of our approach, the close connection to residuated logics is beneficial from the user’s viewpoint. Namely, on the symbolic level, formulas and expressions that formalize queries or constraints have the same reading as in the classical model (with the notion of “similarity” substituted for “equality”) which simplifies their interpretation.
An important consequence of our approach via generalization is that we consider similarities as graded relations on domains and consider the ranked data tables as graded relations between domains. The graded relations play the same role in our model as the ordinary relations on relation schemes in the classical model. In particular, they represent both the base relations and results of queries. In the paper, we present two equivalent query systems, one based on the underlying logic and one based on algebraic manipulation with ranked data tables over domains with similarities. The set of algebraic operations we introduce in the paper is small but we show that a variety of reasonable and practically motivated similarity-based operations can be derived from the basic operations.

The paper is organized as follows. In Section 3, we present the fundamental concepts of our model. The section assumes that readers are acquainted with residuated lattices, we therefore present a survey of utilized notions in Section 2. In Sections 4 and 5, we introduce a relational algebra and a domain calculus in our model, and in Section 6 we prove their mutual equivalence. Furthermore, in Section 7, we present derived relational operations and show that various similarity-based operations can be derived from the basic ones from Section 4. Finally, in Section 8 we outline further extensions of the model which result by extending the underlying logic. In the rest of the introduction, we comment on related approaches.

1.1. Related work

Let us first note that our model should not be confused with probabilistic methods in databases which are currently gaining popularity, see Cavallo and Pittarelli (1987) for an early approach and Dalvi and Suciu (2007a), Dalvi and Suciu (2007b) for recent development. Indeed, our model is deterministic, truth-functional, and unlike probabilistic databases, does not deal with uncertain data. This is in contrast with probabilistic databases based on possible world semantics, cf. Dalvi, Ré, and Suciu (2009) for a current survey. Although conceptually different, our approach and probabilistic methods are complementary, both focusing at different aspects of data management, and their mutual relationship is worth further research.

Issues related to rank-aware querying in relational databases with the emphasis on top- \( k \) querying have been investigated in various extension-based approaches. The most mature is arguably the approach by Li et al. (2005) which extends the classic algebra by a new operator \textit{rank} and studies various aspects related to the physical model, including efficient query execution. A survey and classification of approaches can be found in Ilyas, Beskales, and Soliman (2008). Note that according to a taxonomy in the survey, our model would be classified as the mainstream – it uses \textit{exact methods over certain data} but proceeds by generalization, as discussed above. The papers concerned with rank-aware extensions deal almost exclusively with issues of querying and the underlying physical model. In contrast, we focus solely on the logical model, intentionally leaving the aspects of (possible) physical models and implementations. There are two good reasons for such treatment: (i) we adhere to the logical independence of the model and try to avoid premature optimization; (ii) there are indications that existing rank-aware methods can be exploited to be used for implementation of our model – this certainly deserves attention but is far beyond the scope of this paper.
From a conceptual point of view, the model is closely related to the seminal paper by Fagin (1999) on evaluation of monotone queries. Leaving aside technical details, the issues studied in Fagin (1999) have a rather similar motivation as our model and, interestingly, for a monotone fragment of our model (not all our operations are monotone, cf. Section 4.4), one can adopt algorithms from Fagin (1999) to efficiently execute queries provided that similarities on domains admit efficient implementation of the “sorted access”, which is a reasonable assumption. Related to that are various approaches which deal with similarity selections and joins in databases, see Jacox and Samet (2008), Augsten and Böhlen (2013) and the references therein for an extensive comparison. Most of them are based on distance-based similarity, i.e. similarities are formalized using metric methods. Importantly, there is little systematic connection between the logic and metric facets of the resulting models. That is, the models consist of a logic-based core with atop of which is added, as if “glued”, a metric-based “similarity module”. Compared to that, our approach is considerably more comprehensive because similarities, formalized as general graded relations, are an inherent part of the logic-based core of relational model and admit a purely logical/relational treatment in our approach. Nevertheless, the existing proposed methods can certainly be valuable for implementation of algorithms concerned with particular types of similarities in our model. The basic semantic structures used in our model, so-called ranked data tables, may be seen as relations where tuples have particular annotations. From this point of view, our approach is related to other approaches where annotated relations appear. For instance, Imieliński and Lipski (1984) consider relations with tuples annotated by boolean formulas and Green, Karvounarakis, and Tannen (2007) consider relations with tuples annotated by elements of semi-rings. Our query systems may be seen as domain-independent variants of the query systems outlined by Belohlavek and Vychodil (2010).

The survey of approaches we have given here is far from being complete or exhaustive. A detailed study on relationship of various approaches to our model and their possible use and/or adaptation is an important future goal. While the related approaches can help establish an efficient physical model for the logical model we present in this paper, the merit is reciprocal – we offer a sound and conceptually clear logical model which may serve as a theoretical foundation for ranking and similarity-based techniques in relational databases. To demonstrate this claim is an important aim of this paper.

2. Preliminaries: residuated structures

Our approach assumes that ranks and similarity degrees come from a set of elements (degrees) which is equipped with additional relations (e.g. a partial order to compare degrees) and operations (e.g. conjunction-like operations to aggregate degrees). Instead of using a fixed structure of degrees like a real unit interval and a fixed set of operations, we postulate conditions which must be satisfied by any reasonable sets of degrees and operations defined on the sets. This approach is general and theoretically clean – our reasoning involves only the assumed conditions and is not obstructed by a particular structure of degrees. In this section, we present our requirements on the structures of degrees.

We start by considering a nonempty set $L$ of degrees which is equipped with a partial order denoted $\leq$, i.e. $\leq$ is reflexive, antisymmetric, and transitive. The degrees from $L$ are interpreted comparatively according to $\leq$: $a \leq b$ means that (the rank/the similarity...
degree) \( b \) is at least as high as (the rank/the similarity degree) \( a \). Furthermore, we assume that the partially ordered set \( \langle L, \leq \rangle \) has its least and greatest elements denoted by 0 and 1, respectively. In terms of ranks, the element 0 represents no match whereas 1 represents a full match. Analogously, in terms of similarities, 0 represents no similarity whereas 1 represents full similarity.

In order to be able to correctly define relational operations like projection and division in our model, we further postulate that \( \leq \) is a lattice order (any two elements from \( L \) must have their least upper bound and greatest lower bound in \( L \)). Alternatively, we may have postulated that \( \leq \) is a linear order but this assumption seems to be too strict. Indeed, nonlinear scales of degrees may naturally arise when comprising degrees from multiple independent linear scales – imagine multiple experts defining similarities using their own linear scales which are then combined together as a direct product of the scales, resulting in a nonlinear scale of degrees. For technical reasons, we make even stronger assumption, namely that \( \leq \) is a complete lattice order: any subset \( K \subseteq L \) has its least upper bound in \( L \) (as a supremum) denoted by \( \bigvee K \) and its greatest lower bound in \( L \) (an infimum) denoted by \( \bigwedge K \). Note that \( a \leq b \) holds true iff \( a = a \land b \) (or dually, \( a \lor b = b \)); the existence of 0 and 1 follows from the fact that \( \leq \) is a complete lattice since \( \bigwedge \emptyset = \bigvee L = 1 \) and \( \bigvee \emptyset = \bigwedge L = 0 \), i.e. we may denote \( L \) together with the complete lattice order \( \leq \) by \( \langle L, \land, \lor, 0, 1 \rangle \).

For reasonable analogues of joins and restrictions in our model, it is desirable to equip \( L \) with an operation to aggregate degrees in a conjunctive manner. For instance, if \( a \in L \) and \( b \in L \) are degrees to which a tuple matches two individual queries, \( Q_1 \) and \( Q_2 \), respectively, then based on \( a \) and \( b \), we may want to express the degree to which the tuple matches the compound conjunctive query \( Q_1 \) and \( Q_2 \). In order to retain the truth-functional virtue on the classic relational model, the result should only depend on \( a \) and \( b \) and not on the structure of \( Q_1 \) and \( Q_2 \). Therefore, the result may be seen as a degree \( a \otimes b \) obtained by applying a suitable binary operation \( \otimes : L \times L \rightarrow L \) to \( a \) and \( b \). Again, we shall postulate conditions for \( \otimes \) in order to have a reasonable binary operation on \( L \) which can be regarded as a conjunctive aggregator (e.g. it is desirable to have \( \otimes \) order preserving). Note that in a similar spirit, Fagin (1999) motivates the use of monotone operations to aggregate degrees which originate from evaluating subqueries, possibly in multiple separate query subsystems.

Following the seminal observations of Goguen (1969) and later approaches dealing with logics based on residuated structures (Hájek 1998; Galatos et al. 2007), we argue that suitable properties of \( \otimes \) are guaranteed if there is a binary operation \( \rightarrow \) which is adjoint to \( \otimes \). Altogether, \( \langle L, \land, \lor, \otimes, \rightarrow, 0, 1 \rangle \) is a so-called complete residuated lattice. Formally, a complete residuated lattice is a general algebra denoted \( L = \langle L, \land, \lor, \otimes, \rightarrow, 0, 1 \rangle \) such that

(i) \( \langle L, \land, \lor, 0, 1 \rangle \) is a complete lattice (see above);
(ii) \( \langle L, \otimes, 1 \rangle \) is a commutative monoid (i.e. \( \otimes \) is a binary operation which is commutative, associative, and \( a \otimes 1 = 1 \otimes a = a \) for each \( a \in L \)); and
(iii) the operations \( \otimes \) (called a multiplication) and \( \rightarrow \) (called a residuum) satisfy the following adjointness property:

\[
a \otimes b \leq c \quad \text{iff} \quad a \leq b \rightarrow c
\]

for all \( a, b, c \in L \).
As a consequence of (i)–(iii), one obtains

\[ a \rightarrow b = \bigvee \{ c \in L \mid a \otimes c \leq b \}, \tag{2} \]

meaning that for \( \otimes \) there is a unique \( \rightarrow \) satisfying (1). Furthermore, \( \otimes \) is order preserving in the usual sense: if \( a_1 \leq b_1 \) and \( a_2 \leq b_2 \), then \( a_1 \otimes a_2 \leq b_1 \otimes b_2 \). While \( \otimes \) can be seen as a conjunction-like aggregation operation, \( \rightarrow \) can be seen as an implication-like aggregation operation and it has an important role in our model – it is used to define relational division and to express similarity-based dependencies in data based on a graded containment defined using \( \rightarrow \). Interestingly, conditions (i)–(iii) from the definition of residuated lattices are equivalent to postulating (i), (ii), and the following law of distributivity of \( \otimes \) over general \( \bigvee \):

\[ a \otimes \bigvee_{i \in I} b_i = \bigvee_{i \in I} (a \otimes b_i), \tag{3} \]

for all \( a \in L \) and \( b_i \in L \) \((i \in I) \). Condition (3) ensures reasonable properties of important notions in our model including semijoins. Altogether, we consider complete residuated lattices to be general enough and yet sufficiently powerful algebras to be taken as structures of degrees in our model.

Several desirable properties of operations of residuated lattices are direct consequences of the adjointness property. For instance, \( a \leq b \) holds true iff \( a \rightarrow b = 1 \) which is easy to see since \( a \leq b \) is true iff \( 1 \otimes a \leq b \) due to the neutrality of 1, thus, by adjointness, \( 1 \leq a \rightarrow b \), meaning \( a \rightarrow b = 1 \). Further important properties which we present here without proofs include

\[ a \rightarrow (b \rightarrow c) = (a \otimes b) \rightarrow c = b \rightarrow (a \rightarrow c), \tag{4} \]

\[ \bigwedge_{i \in I} (a \rightarrow b_i) = a \rightarrow \bigwedge_{i \in I} b_i, \tag{5} \]

\[ \bigwedge_{i \in I} (a_i \rightarrow b) = \bigvee_{i \in I} a_i \rightarrow b, \tag{6} \]

**Remark 2.1:** The adjointness property is crucial in multiple-valued logics and in particular in mathematical fuzzy logics (Hájek 1998) which are truth-functional and use \( \otimes \) and \( \rightarrow \) as truth functions of (fuzzy) conjunction and implication, respectively. That is, if \( e(\varphi) \) and \( e(\psi) \) are degrees to which formulas \( \varphi \) and \( \psi \) are true under some evaluation \( e \), then \( e(\varphi) \rightarrow e(\psi) \) is a truth degree of formula \( \varphi \Rightarrow \psi \) (\( \Rightarrow \) is a symbol of the implication connective) under the evaluation \( e \). Hence, we may write \( e(\varphi \Rightarrow \psi) = e(\varphi) \rightarrow e(\psi) \); analogously for the conjunction interpreted by \( \otimes \). In the same spirit, the general infima \( \bigwedge \) and suprema \( \bigvee \) are used as interpretations of the universal and existential quantifiers. Notice the same is true in the classical propositional/predicate logics with \( e \) being either a valuation of propositional variables which uniquely extends to all propositional formulas (in the propositional case); or with \( e \) given by a first-order structure together with valuations of object variables (in the predicate case). In the classic case, of course, \( L = \{0, 1\} \), and \( \rightarrow \) is the ordinary truth function of implication, i.e. \( 0 \rightarrow 0 = 0 \rightarrow 1 = 1 \rightarrow 1 = 1 \) and \( 1 \rightarrow 0 = 0 \). The important point is that the relationship between \( \otimes \) and \( \rightarrow \) imposed by (1) is motivated by graded version of the inference rule *modus ponens* saying that from \( \varphi \) valid to degree (at least) \( a \in L \) and \( \varphi \Rightarrow \psi \) valid to degree (at least) \( b \in L \), we can infer \( \psi \) valid to degree (at least) \( a \otimes b \). Indeed, (1) ensures its maximal inference power so that
the graded rule is still sound (observe that for \( a = b = 1 \) the classic *modus ponens* can be seen as a particular case of the graded rule). The correspondence of adjointness to *modus ponens* has been recognized by Goguen (1969) and later exploited in various logics based on residuated structures.

The class of complete residuated lattices is rich and includes infinite as well as finite structures. Popular infinite structures include linearly ordered complete residuated lattices (chains) defined on the real unit interval with \( \wedge \) and \( \vee \) being minimum and maximum, \( \otimes \) being a left-continuous (or a continuous) triangular norm with the corresponding \( \rightarrow \), see Klement, Mesiar, and Pap (2000). In the examples, we make use of the following pairs of adjoint operations on the unit interval:

\[
\begin{align*}
\text{Łukasiewicz:} & \quad a \otimes b = \max(a + b - 1, 0), \\
& \quad a \rightarrow b = \min(1 - a + b, 1), \\
& \quad a \otimes b = \min(a, b), \\
\text{Gödel:} & \quad a \rightarrow b = \begin{cases} 
1, & \text{if } a \leq b, \\
 b, & \text{otherwise}, 
\end{cases} \\
& \quad a \otimes b = a \cdot b, \\
\text{Goguen:} & \quad a \rightarrow b = \begin{cases} 
1, & \text{if } a \leq b, \\
 \frac{b}{a}, & \text{otherwise}. 
\end{cases}
\end{align*}
\]

It is well known that all multiplications \( \otimes \) from (7)–(9) are continuous and, moreover, each continuous triangular norm can be constructed as an ordinal sum of (isomorphic copies of) these three basic continuous triangular norms, cf. Klement, Mesiar, and Pap (2000).

Some observations in the paper apply only to residuated lattices satisfying additional conditions. \( L \) is called prelinear if \( (a \rightarrow b) \vee (b \rightarrow a) = 1 \) for all \( a, b \in L \); \( L \) is called divisible if \( a \wedge b = a \otimes (a \rightarrow b) \) for all \( a, b \in L \).

**Remark 2.2:** A particular case of a finite (and thus complete) residuated lattice is the two-element Boolean algebra \( \mathbb{L} = \langle \{0, 1\}, \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle \), which is the structure of truth degrees of the classical logic and thus the underlying structure of degrees in the classic relational model of data. That is, the operations \( \wedge, \vee, \otimes, \rightarrow \) of \( \mathbb{L} \) are the truth functions (interpretations) of the corresponding logical connectives of the classical logic (\( \otimes \) coincides with \( \wedge \)). More generally, (complete) Boolean algebras are (term equivalent to) particular (complete) residuated lattices which satisfy the law of the excluded middle (* tertium non datur *), i.e. \( a \vee (a \rightarrow 0) = 1 \). Obviously, there are residuated lattices which are not Boolean algebras, e.g. take any linearly ordered \( L \) with more than two elements and observe that \( a \vee (a \rightarrow 0) < 1 \) whenever \( 0 < a < 1 \). Therefore, if we prove an assertion for general \( L \) (i.e. over all complete residuated lattices taken as structures of degrees) then, as a particular case, we get a classic counterpart of this assertion for \( L \) being the two-valued Boolean algebra but the converse does not hold since residuated lattices are more general structures. As a consequence, some laws which hold in the classical relational model will no longer hold in our model (at least not in the same form). This is a natural trait of the more general model which allows us to handle (intermediate) degrees of similarity. Nevertheless, by generalizing the fundamental topics from query systems and dependency theory, we provide evidence that complete residuated lattices are suitable structures of similarity degrees and ranks.
For a complete residuated lattice \( L \) and a nonempty set \( U \) (call it a universe), we may consider maps of the form \( f : U \to L \), assigning to each \( u \in U \) the degree \( f(u) \in L \). Such maps are usually denoted by capital letters \( A, B, C, \ldots \) instead of \( f, g, h, \ldots \), and are called \( L \)-sets in \( U \). If \( A \) is an \( L \)-set in \( U \), then \( A(u) \) can be interpreted as “the degree to which \( u \) belongs to \( A \)”. Note that if \( L = \{0, 1\} \), then by a slight abuse of notation, we can identify \( L \)-sets in \( U \) with subsets of \( U \) (in the usual sense), i.e. with maps \( f : U \to \{0, 1\} \) (indicator functions of subsets of \( U \)). For \( L \)-sets \( A, B \) in \( U \), we write \( A \subseteq B \) whenever \( A(u) \leq B(u) \) holds for each \( u \in U \) and say that \( A \) is (fully) contained in \( B \). An \( L \)-set \( A \) in \( U \) is called finite if the set \( \{u \in U | A(u) > 0\} \) is finite; \( A \) is called crisp if \( A(u) \in \{0, 1\} \) for each \( u \in U \); \( A \) is called empty if \( A(u) = 0 \) for each \( u \in U \); the empty \( L \)-set in \( U \) is denoted by \( \emptyset_U \) (or just \( \emptyset \) if there is no danger of confusion). Obviously, crisp \( L \)-sets in \( U \) can be identified with ordinary subsets of \( U \). Therefore, for a crisp \( A \), we write \( u \in A \) to denote \( A(u) = 1 \) and \( u \notin A \) to denote \( A(u) = 0 \). If \( U \) is a Cartesian product of sets (say \( U_1, \ldots, U_n \)), we may call an \( L \)-set \( A \) in \( U \) an \( L \)-relation (between \( U_1, \ldots, U_n \)). In particular, a binary \( L \)-relation \( R \) between \( U \) and \( V \) is a map \( R : U \times V \to L \) assigning to each \( u \in U \) and each \( v \in V \) the degree \( R(u, v) \in L \) to which \( u \) and \( v \) are related by \( R \). As usual, if a symbol like \( \sim \) denotes a binary \( L \)-relation, we write \( u \sim v \) to denote the degree \( \sim(u, v) \). Since \( L \)-relations are particular \( L \)-sets, the notions like finiteness and containment apply to \( L \)-relations as well.

The survey of preliminaries provided here is sufficient for reading the paper. Readers interested in more details of the logical background and residuated structures can find details in the monographs by Hájek (1998), Gerla (2001), Belohlavek (2002), Galatos et al. (2007), and a series of edited handbooks (Cintula, Hájek, and Noguera 2011) covering recent results.

### 3. Fundamental concepts

This section introduces the basic concepts of our model, ranked data tables over domains with similarities which represent the counterpart of relations on relation schemes from the Codd model, and general notions related to relational querying. In this and the following sections, we assume that \( L \) is a complete residuated lattice, see Section 2 for a survey.

#### 3.1. Attributes, objects constants, and types

As in the classical model, we consider attributes which are given by their names and domains (types). We let \( Y \) denote a set of all attribute names and denote the names in \( Y \) by \( y, y', y_1, y_2, \ldots \) We assume that \( Y \) is denumerable (countably infinite) to have enough attribute names at our disposal. If there is no danger of confusion, we tacitly identify attributes with their names. Any finite subset \( R \subseteq Y \) is called a relation scheme. Relation schemes can be seen as specifications of (names of) columns in data tables. In particular, \( R = \emptyset \subseteq Y \) is an empty relation scheme.

In order to denote individual attribute values that can appear in a database, we introduce a set \( C \) of object constants, denoted \( c, d, c_1, c_2 \ldots \), which is also assumed to be denumerable.

The role of object constant will become important in Sections 4.8 and 5.

With each attribute and object constant, we consider its type. The type is a label annotating attributes and object constants to ensure that relational operations can only be performed using attributes and/or constants of the same type.
Definition 3.1: A type declaration for a set $Y$ of attributes and a set $C$ of object constants is a structure $\Lambda = \langle \Lambda, \lambda \rangle$, where $\Lambda$ is a nonempty set of elements called types, and $\lambda$ is a map $\lambda: Y \cup C \to \Lambda$ such that for each $\tau \in \Lambda$ there is an infinite subset $Y' \subseteq Y$ satisfying $\lambda(Y') = \{ \tau \}$.

As usual, if $\lambda(y) = \lambda(z)$, we say that under $\Lambda = \langle \Lambda, \lambda \rangle$, $y$ and $z$ have the same type. Definition 3.1 ensures that for each type there are infinitely many attributes of that type ($\Lambda$ is thus at most denumerable).

3.2. Domains, similarities, and tuples

For each attribute $y \in Y$ we consider its domain, usually denoted by $D_y$, which is a nonempty set of all possible values of the attribute $y \in Y$. Before we point out the relationship between domains and types, we formalize similarities on domains. We equip each domain $D_y$ with a binary $L$-relation $\approx_y: D_y \times D_y \to L$ (and use infix notation, $u \approx_y v$, etc.) satisfying

(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 $L$-relation $\approx_y$ on $D_y$ satisfying (Ref) and (Sym) is called a similarity. It is easily seen that (Ref) and (Sym) are the counterparts for $L$-relation of the ordinary reflexivity and symmetry. Namely, (Ref) says that each element is fully similar to (indistinguishable from) itself and (Sym) says that the degree to which $u$ is similar to $v$ is the same as the degree to which $v$ is similar to $u$.

Remark 3.2:

(a) While (Ref) is usually accepted without any reservation, it may be arguable whether a similarity is indeed symmetric (e.g. Tversky Index, see Tversky (1977), is a popular asymmetric similarity measure). We employ (Sym) for technical reasons. Namely, dropping (Sym) would have consequences we consider undesirable. For instance, similarity-based natural joins would depend on the order of the input arguments, violating one of the basic properties that are inherent to natural joins.

(b) Sometimes it is useful to require stronger properties for similarities. For instance, one may postulate:
(Sep) for each $u, v \in D_y: u \approx_y v = 1$ iff $u$ equals $v$;
(Tra) for each $u, v, w \in D_y: u \approx_y v \otimes v \approx_y w \leq u \approx_y w$.

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 if and only if $u$ and $v$ are identical (indistinguishable within its domain). Property (Tra) is called $\otimes$-transitivity and it represents a transitivity of $\approx_y$ with respect to the 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 “and” is given by $\otimes$.

For a domain $D_y$ and similarity $\approx_y$ on $D_y$, we call the pair $\langle D_y, \approx_y \rangle$ a domain with similarity. In database literature like (Date 2005), domains are often identified with types and considered as named sets of values. We distinguish the notions here. From the logic point of view, we treat types as syntactic notions (names) whereas domains (set of values) are considered as semantic notions and can be seen as interpretations of types, cf. Lacroix
and Pirotte (1980). The distinction has its benefits since various issues can be resolved syntactically based on types only without considering concrete sets of values. To keep types and the corresponding domains consistent, we introduce the following requirement.

**Definition 3.3:** A set \( \{ \langle D_y, \approx y \rangle \mid y \in Y \} \) of domains with similarities respects type declaration \( \Lambda = \langle \Lambda, \lambda \rangle \) if for all \( y_1, y_2 \in Y \), the following condition is satisfied: If \( \lambda(y_1) = \lambda(y_2) \), then \( \langle D_{y_1}, \approx y_1 \rangle \) coincides with \( \langle D_{y_2}, \approx y_2 \rangle \).

**Remark 3.4:** Our intention is to develop an independent logical model of data, we therefore do not discuss where the similarities came from because this depends on a particular application (the similarities can be computed using distance-based methods, can be given by experts, can be determined by a survey of customers, and can be obtained by other rules, e.g. the Leibniz similarity, see Belohlavek (2002) for a survey of methods).

Recall that a Cartesian product \( \prod_{i \in I} A_i \) of an \( I \)-indexed system \( \{ A_i \mid i \in I \} \) is a set of all maps \( f : I \to \bigcup_{i \in I} A_i \) such that \( f(i) \in A_i \) holds for each \( i \in I \). Let \( R \subseteq Y \) be a relations scheme and define \( \text{Tupl}(R) \) by

\[
\text{Tupl}(R) = \prod_{y \in R} D_y. \tag{10}
\]

Each \( r \in \text{Tupl}(R) \) is called a tuple over \( R \). By definition, \( r(y) \in D_y \) for each \( y \in R \); \( r(y) \) is called the \( y \)-value of \( r \). In the following sections, we prefer to use the abbreviated form \( \text{Tupl}(R) \) instead of \( \prod_{y \in R} D_y \) whenever the domains are clear from the context.

### 3.3. Ranked data tables over domains with similarities

We now introduce the notion of a ranked data table over domains with similarities which is our counterpart to the classic notion of a relation on a relation scheme:

**Definition 3.5:** Let \( R \subseteq Y \) be a relation scheme and let \( \langle D_y, \approx y \rangle \) be domains with similarities for attributes \( y \in R \). A ranked data table on \( R \) over \( \{ \langle D_y, \approx y \rangle \mid y \in R \} \) (shortly, an RDT) is any map

\[
\mathcal{D} : \prod_{y \in R} D_y \to L \tag{11}
\]

such that the set \( \{ r \in \prod_{y \in R} D_y \mid \mathcal{D}(r) > 0 \} \), called the answer set of \( \mathcal{D} \), is finite. The cardinality of the answer set of \( \mathcal{D} \) is called the size of \( \mathcal{D} \) and is denoted by \( |\mathcal{D}| \). If \( |\mathcal{D}| = 0 \), then \( \mathcal{D} \) is called the empty RDT on \( R \) and denoted by \( 0_R \). Furthermore, \( \mathcal{D} \) is called nonranked if \( \mathcal{D}(r) \in \{ 0, 1 \} \) for any \( r \). Each degree \( \mathcal{D}(r) \in L \) is called a rank of \( r \) in \( \mathcal{D} \).

Using the notion of an \( L \)-set Goguen (1967), RDTs are in fact finite \( L \)-sets in the Cartesian product \( \prod_{y \in R} D_y \), i.e. only finitely many tuples from the Cartesian product are assigned nonzero ranks. In addition, nonranked RDTs can be seen as finite crisp \( L \)-sets, i.e. finitely many tuples are assigned rank 1 (full match) and the others are assigned rank 0 (no match). Nonranked RDTs can be identified with the classic relations on relation schemes with \( \mathcal{D}(r) = 1 \) meaning that “tuple \( r \) belongs to (the answer set of) \( \mathcal{D} \)”. In particular, if \( L \) is the two-element Boolean algebra then all RDTs are nonranked, thus we have a one-to-one correspondence between the classic relations on relation schemes and RDTs.
Remark 3.6:

(a) The intended meaning of RDTs is to represent results of similarity-based queries, i.e. queries which involve similarities and which are naturally matched to degrees in $L$. Thus, the rank $D(r)$ of $r$ in $D$ is a degree to which the tuple $r$ satisfies requirements imposed by a similarity-based query. In addition, RDTs can also be interpreted as the basic data structures in the model. This is obvious in particular when $D$ is nonranked. Therefore, as in the classical relational model, RDTs represent both the results of queries and stored data (base relations).

(b) Analogously as in the classical model, RDTs on nonempty relation schemes can be depicted by two-dimensional tables with rows corresponding to tuples with nonzero ranks and with an additional column containing the ranks. If a table is nonranked, the column with ranks can be omitted and hence they appear as the usual data tables. Let us stress that although the ranks are written as values in a column, the column itself does not correspond to any attribute.

There are several RDTs in special forms which deserve our attention. First, if the relation scheme $R$ is empty, then from the set-theoretic definition of direct products as sets of maps, we get $\text{Tupl}(\emptyset) = \prod_{y \in \emptyset} D_y = \{\emptyset\}$. Therefore, the empty set $\emptyset$ (which in fact represents here the only map from $\emptyset$ to $\emptyset$) is the only (empty) tuple on $R = \emptyset$. Thus, each RDT $D$ on $R = \emptyset$ is uniquely given be the degree $D(\emptyset) \in L$. Therefore, it makes sense to introduce the following notation for all RDTs on the empty relation scheme:

**Definition 3.7:** For $a \in L$, we let $a_\emptyset$ denote the RDT on $\emptyset$ such that $a_\emptyset(\emptyset) = a$.

Among the particular cases of Definition 3.7 are the RDTs $0_\emptyset$ (with the empty answer set) and $1_\emptyset$ (with $\emptyset$ having rank 1). In the classical relational model, the analogues of $0_\emptyset$ and $1_\emptyset$ are the empty relation on the empty scheme (representing the truth value 0) and the relation on the empty scheme containing the empty tuple (representing the truth value 1). In the database language Tutorial D which is based on the principles of *The Third Manifesto (TTM)*, the relations are called TABLE_DUM and TABLE_DEE, see *Date and Darwen (2006, Chapter 5)*. Notice that the notation for empty RDTs $0_R$ from Definition 3.5 coincides with the notation from Definition 3.7 only in case of $a = 0$ (and $R = \emptyset$), i.e. only if $a_\emptyset$ is empty according to Definition 3.5, thus no ambiguity in the notation has been introduced.

The second important form of RDTs represents nonranked tables with a single attribute and a single tuple in the answer set:

**Definition 3.8:** Let $y \in Y$ and $d \in D_y$. A singleton RDT on $\{y\}$ over $D_y$ is the RDT denoted $[y:d]$ such that $[y:d](r) = 1$ if $r(y) = d$ and $[y:d](r) = 0$ otherwise.

**Remark 3.9:** Let us note that, as in the classical model, the set-theoretic nature of RDTs does not imply any particular order of tuples and/or attributes in the RDTs. In addition, (the table representation of) RDTs cannot contain “duplicate tuples” or “unknown values” – two concepts which are present in some query languages based on the relational model (e.g. in SQL) but which, strictly speaking, violate the basic principles of the relational model and hence we do not endorse them (*Date (1995)* presents good arguments against NULLs, see also recent paper by *Libkin (2015)*). We should also mention that our approach does not put any restriction on the notion of a domain. In particular, we do not require any form of “atomicity of values” (as often vaguely used to describe relations in 1NF), elements
of domains can be arbitrary. In this sense, we adhere to the true meaning of 1NF as it is presented by Date (2006).

### 3.4. Relational querying

As we outlined in the introduction, we propose in this paper two query systems: (i) a system based on a generalized relational algebra, and (ii) a system based on a generalized domain relational calculus. In this section, we introduce notions which are common to both of these query systems and can be utilized in general considerations on relational querying.

In general, we view queries as functions mapping collections of RDTs representing input data to an RDT representing the result. The input RDTs for a query can be seen as a snapshot of a database at a particular moment in time. We introduce database schemes to describe relation schemes of RDTs that can represent input data for a query.

**Definition 3.10:** A database scheme on $Y$ is any couple $\langle R, \varrho \rangle$ where $R$ is a finite nonempty set of relation symbols and $\varrho$ is a map $\varrho : R \rightarrow 2^Y$ assigning to each relation symbol $r \in R$ a relation scheme $\varrho(r) \subseteq Y$.

As in case of relations schemes, a database scheme is a syntactic notion. Database schemes can be interpreted by database instances, i.e. collections of RDTs interpreting relation symbols from the database scheme. A formal definition follows.

**Definition 3.11:** A database instance $D$ (shortly, an instance) of a database scheme $\langle R, \varrho \rangle$ on $Y$ and object constants $C$ is a triplet $D = \langle U_D, R_D, C_D \rangle$ where

1. $U_D = \{\langle D_y, \approx_y \rangle \mid y \in Y\}$ is a set of domains with similarities;
2. $R_D$ is a set of RDTs such that for each relation symbol $r \in R$ there is an RDT $r_D \in R_D$ on $\varrho(r)$ over $\{\langle D_y, \approx_y \rangle \mid y \in \varrho(r)\} \subseteq U_D$;
3. $C_D = \{c_D \in \bigcup_{y \in Y} D_y \mid c \in C\}$ is a set of values interpreting object constants.

If $C$ is clear from context, we say that $D$ is an instance of $\langle R, \varrho \rangle$. Moreover, we say that $D = \langle U_D, R_D, C_D \rangle$ respects type declaration $\Lambda = \langle \Lambda, \lambda \rangle$ if $U_D$ respects $\Lambda$ and if for each $c \in C$ and $y \in Y$ such that $\lambda(c) = \lambda(y)$, we have $c_D \in D_y$.

Database instances represent states of databases at given moments. Indeed, for each relation symbol $r \in R$, a database instance contains an RDT $r_D$ which interprets $r$ and for each object constant $c \in C$ it contains a value $c_D$ interpreting $c$ in $D$. From the logical point of view, database instance is a structure for a language which is determined by the database scheme and the set of object constants.

In general, a query on database scheme $\langle R, \varrho \rangle$ and object constants $C$ is a map from the set of all database instances of $\langle R, \varrho \rangle$, and $C$ to the set of all RDTs. Typically, queries correspond to evaluation of special expressions or formulas in database instances. In Section 4 and Section 5, we introduce two types of query systems which correspond to evaluation of relational algebra expressions and formulas of domain relational calculus.

### 4. Relational operations and relational algebra

In the same spirit as in the Codd model, we define a set of fundamental operations with RDTs that can be combined to form complex queries represented by terms. Such queries are evaluated in database instances, obtaining RDTs as results.
The set of relational operations we use in this section has been selected to ensure a sufficient expressive power of the query system. Our approach to relational operations is conceptually similar as in the classical relational model; however, it differs in several key aspects which are consequences of the presence of similarity relations and the structure of degrees, namely the residuated lattice which replaces the two-element Boolean algebra of the classical model.

A minor technical difference is the need to have a relational division in our model as a fundamental operation which is, in general, not expressible using the other operations. The most notable difference is the explicit manipulation with similarities and ranks that appear in our operations which take RDTs as input and produce other RDTs as output (query results). All the operations introduced in this section cope with similarities and ranks in a way that is logically sound and ensures that (i) results of operations are always finite and (ii) the operations conservatively extend the operations of the classical relational model, meaning that if one takes the two-element Boolean algebra for the structure of degrees and considers identity relations for similarities on domains, our relational operations have the same expressive power as the operations in the classical model.

In this section, we introduce relational operations which are grouped into several categories depending on their roles. Later, we formalize the notion of a relational query and its evaluation in database instances. Operations derived from the basic ones, including similarity-based joins, and extensions of the operations will be shown in Sections 7 and 8.

4.1. Counterparts to Boolean operations

The first category of operations contains our counterparts to the basic Boolean operations of Codd’s model – the union and the intersection. The operations emerge in our model by a componentwise application of operations of residuated lattices. For RDTs $D_1$ and $D_2$ on relation scheme $R$, we define

$$ (D_1 \cup D_2)(r) = D_1(r) \lor D_2(r), $$

$$ (D_1 \cap D_2)(r) = D_1(r) \land D_2(r), $$

$$ (D_1 \otimes D_2)(r) = D_1(r) \otimes D_2(r), $$

for all tuples $r$ on $R$. $D_1 \cup D_2$ is called a union of $D_1$ and $D_2$; $D_1 \cap D_2$ and $D_1 \otimes D_2$ are called the $\land$-intersection and $\otimes$-intersection of $D_1$ and $D_2$, respectively. RDT $D$ on relation scheme $R$ is called idempotent (with respect to $\otimes$) if $D \otimes D = D$.

Remark 4.1:

(a) Recall that the set-theoretical union of relations $D_1$ and $D_2$ from the classical model is $D_1 \cup D_2 = \{ r \mid r \in D_1 \text{ or } r \in D_2 \}$ where “or” stands for a non-exclusive Boolean disjunction. Thus, $r \in D_1 \cup D_2$ iff $r \in D_1$ or $r \in D_2$. Identifying sets with their indicator functions, we can write $(D_1 \cup D_2)(r) = 1$ iff $D_1(r) = 1$ or $D_2(r) = 1$. Hence, $(D_1 \cup D_2)(r) = D_1(r) \lor D_2(r)$, where $\lor$ is the truth function of the Boolean disjunction. In this sense, a union of RDTs (12) is obtained by replacing the Boolean disjunction by $\lor$ which is a supremum from a complete residuated lattice $L$. Therefore, the rank $(D_1 \cup D_2)(r)$ of tuple $r$ in the union $D_1 \cup D_2$ of RDTs is a supremum of ranks of $r$ in $D_1$ and $D_2$. Notice that on the symbolic level, (12)
has the same interpretation as the classic union only with explicit ranks involved: If \( \mathcal{D}_1 \) is a result of query \( Q_1 \) and \( \mathcal{D}_2 \) is a result of query \( Q_2 \), then \( (\mathcal{D}_1 \cup \mathcal{D}_2)(r) \) shall be interpreted as “a degree to which \( r \) matches \( Q_1 \) or \( r \) matches \( Q_2 \)”. If \( \mathbf{L} \) is linearly ordered, \( \lor \) coincides with maximum.

(b) Following the same arguments as in (a), we defined two kinds of intersections based on two choices of truth functions of conjunctions \( \land \) and \( \otimes \) in \( \mathbf{L} \). Note that the \( \land \)-intersection is idempotent, meaning \( \mathcal{D} \land \mathcal{D} = \mathcal{D} \). For the \( \otimes \)-intersection, we have \( \mathcal{D} \otimes \mathcal{D} \subseteq \mathcal{D} \) (i.e. \( (\mathcal{D} \otimes \mathcal{D})(r) \leq \mathcal{D}(r) \) for all \( r \)) but the converse inclusion does not hold in general, i.e. the \( \otimes \)-intersection is not an idempotent operation in general and therefore we call \( \mathcal{D} \) idempotent to denote that \( \mathcal{D} \otimes \mathcal{D} = \mathcal{D} \) for \( \mathcal{D} \) and \( \otimes \). All nonranked RDTs are idempotent, since \( 0 \otimes 0 = 0 \) and \( 1 \otimes 1 = 1 \) for any \( \otimes \). Using properties of residuated lattices, \( \mathcal{D}_1 \otimes \mathcal{D}_2 \subseteq \mathcal{D}_1 \cap \mathcal{D}_2 \), i.e. the \( \land \)-intersection can be seen as an upper bound for all possible \( \otimes \)-intersections. Having two operations of intersections and only a single operation of a union stems from the fact that both \( \land \) and \( \otimes \) and fundamental operations of \( \mathbf{L} \) which both generalize the two-valued conjunction, whereas \( \lor \) is the only fundamental operation of \( \mathbf{L} \) generalizing the two-valued disjunction. In Section 8.2, we show that one can extend \( \mathbf{L} \) by an additional non-idempotent operation \( \oplus \) (truth function of non-idempotent disjunction) and introduce a corresponding operation with RDTs whose lower bound is \( \lor \), meaning \( \mathcal{D}_1 \cup \mathcal{D}_2 \subseteq \mathcal{D}_1 \oplus \mathcal{D}_2 \).

(c) The existence of two intersections in our model can be justified by different requirements on handling ranks of conjunctive queries. Suppose \( \mathcal{D}_1, \ldots, \mathcal{D}_k \) are results of subqueries \( Q_1, \ldots, Q_k \). Then, both \( \mathcal{D}_1 \cap \cdots \cap \mathcal{D}_k \) and \( \mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_k \) represent results of a conjunctive query “\( Q_1 \) and \cdots and \( Q_k \)”. Assume that for a tuple \( r \), we have \( \mathcal{D}_1(r) = 0.5 \) and \( \mathcal{D}_2(r) = \cdots = \mathcal{D}_k(r) = 0.98 \) and the ranks are taken from the real unit interval with its natural ordering. Then, the rank of \( r \) in \( \mathcal{D}_1 \cap \cdots \cap \mathcal{D}_k \) is 0.5. The same holds if \( \mathcal{D}_2(r) = \cdots = \mathcal{D}_k(r) = 0.5 \) because the rank is computed as an infimum (a minimum of \( k \) ranks). Thus, the rank of \( r \) in \( \mathcal{D}_1 \cap \cdots \cap \mathcal{D}_k \) depends solely on the worst match. The \( \otimes \)-intersection can be seen as an alternative to the worst-match meaning of the \( \land \)-intersection. For instance, if \( \otimes \) is the Goguen multiplication on \([0, 1] \), then the rank of \( r \) in \( \mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_k \) equals to \( 0.5 \cdot 0.98^{k-1} \) if \( \mathcal{D}_2(r) = \cdots = \mathcal{D}_k(r) = 0.98 \) and \( 0.5^k \) if \( \mathcal{D}_2(r) = \cdots = \mathcal{D}_k(r) = 0.5 \). The rank \( 0.5 \cdot 0.98^{k-1} \) is considerably higher than \( 0.5^k \) and reflects the fact that in the first case, \( r \) matches \( Q_2, \ldots, Q_k \) almost fully. Thus, with general nonidempotent \( \otimes \), the ranks of all subqueries can contribute to the rank of \( r \) in \( \mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_k \). Hence, the \( \otimes \)-intersection has the ability to separate results (tuples) that match more subqueries to higher degrees from results that match less subqueries to higher degrees. This difference in interpreting the results of \( \land \)-intersections and \( \otimes \)-intersections further justifies the inclusion of both intersections in the model since both interpretations of combining conjunctive subqueries are reasonable and may be required.

(d) Operations (12)–(14) generalize the classic operations with relations in the following sense. If the underlying complete residuated lattice \( \mathbf{L} \) equals \( 2 \) (two-element Boolean algebra) then (12)–(14) become the Boolean operations in which case, the \( \otimes \)-intersection becomes the \( \land \)-intersection.
The following proposition shows that $\otimes$ is distributive over $\cup$. If in addition $L$ is prelinear (which in particular is true in case of linear scales of degrees) or divisible (Hájek 1998), $\otimes$ is distributive over $\cup$ and $\cap$ is distributive over $\cup$. In the same spirit as in the classic relational algebra, we use observations of this type to simplify compound expressions that represent queries – in this case, we can get a result by performing two operations instead of three.

**Proposition 4.2:** For any ranked data tables $D_1, D_2, D_3$ on $R$,

(i) $D_1 \otimes (D_2 \cup D_3) = (D_1 \otimes D_2) \cup (D_1 \otimes D_3)$.

If $L$ is prelinear or divisible, then

(ii) $D_1 \otimes (D_2 \cap D_3) = (D_1 \otimes D_2) \cap (D_1 \otimes D_3)$,

(iii) $D_1 \cap (D_2 \cup D_3) = (D_1 \cap D_2) \cup (D_1 \cap D_3)$.

**Proof:** Follows directly from properties of residuated lattices, see Belohlavek (2002, Theorems 2.25, 2.37, and 2.39).

4.2. Operations based on residuated implication

In order to express results of compound queries containing negation like “match query $Q_1$ but do not match query $Q_2$”, the ordinary relational algebra consists of a binary operation of difference or a unary operation of complement. In our model, we first introduce an operation with RDTs based on the operation of residuum $\rightarrow$ which serves as a truth function of “fuzzy implication”. In a particular case, the operation can be used to express compound negated queries. In addition, we introduce as derived operations so-called residuated shifts allowing to express queries like “match $Q$ at least to a prescribed degree $a$”. Interestingly, queries of this type do not have a nontrivial counterpart in the classical model but emerge in our model as users may want to emphasize or relax results of particular subqueries of a compound query.

**Remark 4.3:** The componentwise application of $\rightarrow$ is not straightforward as in case of the union and intersections where the results have always finite supports and thus are well-defined RDTs. If $D_1$ and $D_2$ are RDTs on relation scheme $R$ over domains $D_y (y \in R)$ and if at least one of the domains is infinite, then

$$D_1(r) \rightarrow D_2(r) = 1$$

for infinitely many tuples $r$ on $R$ over $D_y (y \in R)$. Indeed, there are infinitely many tuples $r$ such that $D_1(r) = 0$ and consequently $D_1(r) \rightarrow D_2(r) = 0 \rightarrow D_2(r) = 1$ for infinitely many tuples $r$. As a consequence, an operation with RDTs based on componentwise application of $\rightarrow$ will not be domain independent in the usual sense – a change of a domain without altering the data can produce different answers to the query which is not desirable. This problem is analogous to the problem of relational complements in the Codd model (Maier 1983) where infinite domain can also produce infinite relations.

The obstacle outlined in Remark 4.3 can be resolved in several ways. First, we can restrict ourselves to finite domains only – an approach we do not consider for the sake of generality. Second, we can introduce active domains as in the classical model, see Maier (1983). Third, we can introduce a new operation from which the residuum can be derived
and which does not yield an infinite RDT if applied componentwise to finite input RDTs. Although the approach via active domains is possible, we use the third one because it resolves the issue on the level of the structure of truth degrees and not on the level of the model itself.

For a complete residuated lattice $L = \langle L, \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$ and for arbitrary degrees $a, b, c \in L$, we define $b \rightarrow^a c \in L$ as follows:

$$b \rightarrow^a c = a \otimes (b \rightarrow c). \quad (16)$$

The degree $b \rightarrow^a c$ shall be called an $a$-residuum of $b \in L$ with respect to $c \in L$. The operation $\rightarrow^a$ as introduced in (16) can be seen as a ternary operation in which case the degrees $a, b, c \in L$ from (16) shall be called the first, the second, and the third arguments of $\rightarrow^a$, respectively.

**Proposition 4.4:** The following are true for $\rightarrow^a$ introduced by (16):

(i) $b \rightarrow^1 c = b \rightarrow c,$
(ii) $1 \rightarrow^a c = 1 \rightarrow^c a = a \otimes c,$
(iii) $0 \rightarrow^a c = b \rightarrow^a 1 = a,$
(iv) $b \rightarrow^0 c = b \rightarrow^b 0 = 1 \rightarrow^b 0 = 0,$
(v) $b \rightarrow^a c \leq b \rightarrow^1 (a \otimes c),$
(vi) $\rightarrow$ is monotone in the first and in the third argument,
(vii) $\rightarrow$ is antitone in the second argument,
(viii) $a \rightarrow^a b \leq a \wedge b,$
(ix) if $L$ is divisible, then $a \rightarrow^a b = a \wedge b,$
(x) if $b \leq c$, then $b \rightarrow^a c = a,$
(xi) if $L$ is a linear $\Pi$-algebra, then $b \leq c$ iff $b \rightarrow^a c = a$ for all $a > 0,$
(xii) $b \rightarrow^b c = c$ iff there is $x \in L$ such that $1 \rightarrow^x b = c,$
(xiii) $1 \rightarrow^a b \leq c$ iff $a \leq b \rightarrow^1 c.$

**Proof:** The properties can be derived from properties of $\otimes$ and $\rightarrow$ in complete residuated lattices, see Belohlavek (2002). \hfill \Box

Note that both $\otimes$ and $\rightarrow$ can be obtained from $\rightarrow^a$, see (i) and (ii) of Proposition 4.4. Hence, each (complete) residuated lattice $L = \langle L, \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$ induces a structure $\langle L, \wedge, \vee, \rightarrow^a, 0, 1 \rangle$ from which we can reconstruct all operations of $L$. The following assertion shows that one can go in the opposite direction as well:

**Theorem 4.5:** Let $L = \langle L, \wedge, \vee, \rightarrow, 0, 1 \rangle$ be a structure such that $(L, \wedge, \vee, 0, 1)$ is a bounded lattice and $\rightarrow$ be a ternary operation satisfying the following conditions:

$$1 \rightarrow^a 1 = a, \quad (17)$$
$$1 \rightarrow^a b = 1 \rightarrow^b a, \quad (18)$$
$$1 \rightarrow^a (1 \rightarrow^b c) = 1 \rightarrow^c (1 \rightarrow^a b), \quad (19)$$
$$1 \rightarrow^a b \leq c \text{ iff } a \leq b \rightarrow^1 c \quad (20)$$

for all $a, b, c \in L$. Then, $L' = \langle L', \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$, where $a \otimes b = 1 \rightarrow^a b$ and $a \rightarrow b = a \rightarrow^1 b$ for all $a, b \in L$, is a residuated lattice.
**Proof:** By a moment’s reflection, one can see that (17) encodes the fact that 1 is neutral w.r.t. $\otimes$, (18) and (19) yield that $\otimes$ is commutative and associative, and (20) is the adjointness of $\otimes$ and $\rightarrow$. The rest follows immediately. □

Obviously, $L$ from Theorem 4.5 is complete iff the corresponding $L'$ is complete. Since the class of all residuated lattices forms a variety (i.e. it is closed under homomorphic images, subalgebras, and direct products, see Wechler (1992)), we get the following characterization:

**Corollary 4.6:** The class of all bounded lattices with $\rightarrow$ satisfying (17)–(20) is a variety which is term equivalent to the variety of residuated lattices.

Thus, we obtained a class of algebras which are equivalent to residuated lattices up to the formalization of the adjoint operations. The reason behind introducing $\rightarrow$ is that now we can introduce a residuum of RDTs by a componentwise application of $\rightarrow$. For RDTs $D_1, D_2, D_3$ on $R$, we put

$$(D_1 \rightarrow D_3 D_2)(r) = D_1(r) \rightarrow D_3(r) D_2(r)$$

for all tuples $r$. $D_1 \rightarrow D_3 D_2$ is called a residuum of $D_1$ with respect to $D_2$ which ranges over $D_3$.

**Remark 4.7:**

(a) Obviously, $D_1 \rightarrow D_3 D_2$ is an RDT because there are only finitely many tuples $r$ such that $(D_1 \rightarrow D_3 D_2)(r) > 0$. In fact, we have

$$D_1 \rightarrow D_3 D_2 \subseteq D_3,$$

which is due to $a \otimes (b \rightarrow c) \leq a \otimes 1 = a$. Therefore, $D_3$ can be seen as a range for the componentwise application of the residuum. The role of $D_3$ as a range becomes even more explicit if $D_3$ is nonranked, i.e. if $D_3(r) \in \{0, 1\}$ for all tuples $r$. In that case, $D_1 \rightarrow D_3 D_2$ can be expressed as follows:

$$(D_1 \rightarrow D_3 D_2)(r) = \begin{cases} D_1(r) \rightarrow D_2(r), & \text{if } D_3(r) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Put in words, using nonranked $D_3$, one defines a range (a crisp set) of tuples that are used for computing the values of residuum $\rightarrow$ (the original operation of the residuated lattice $L$); ranks of tuples outside of the range will all be zero.

(b) The result of $D_1 \rightarrow D_3 D_2$ is interpreted as an answer to a query which involves an if–then condition. Namely, if the RDTs $D_1, D_2, D_3$ are considered results of the subqueries $Q_1, Q_2, Q_3$, then $(D_1 \rightarrow D_3 D_2)(r)$ is a degree to which “$r$ matches $Q_3$ and if it matches $Q_1$ then it matches $Q_2$”.

The ternary residuum of RDTs $\rightarrow$ has the following properties:

**Proposition 4.8:** For any ranked data tables $D_1, D_2, D_3$ on $R$ and a nonranked data table $D$ on $R$:

(i) $D_1 \rightarrow D (D_2 \rightarrow D D_3) = D_2 \rightarrow D (D_1 \rightarrow D D_3),$

(ii) $(D_1 \otimes D_2) \rightarrow D D_3 = D_1 \rightarrow D (D_2 \rightarrow D D_3),$
Proof: For (i)–(viii), we can use the following argument: Since \( \mathcal{D} \) is nonranked, it suffices to check that for each tuple \( r \) such that \( \mathcal{D}(r) = 1 \), the left-hand side is equal to (less than) the right-hand side of the equality (inclusion). But this follows from the corresponding identities (inequalities) that hold in \( \mathcal{L} \) if \( \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3 \) are replaced by arbitrary degrees from \( \mathcal{L} \). In particular case, if \( a = 0 \), we abbreviate \( \mathcal{D}_2 \boxdiag_a \mathcal{D}_1 \) by \( \mathcal{D}_2 \boxtimes \mathcal{D}_1 \) and call it a (residuated) negation of \( \mathcal{D}_1 \) which ranges over \( \mathcal{D}_2 \). Furthermore, we put

\[
(a \rightarrow_{\mathcal{D}_2} \mathcal{D}_1)(r) = a \rightarrow_{\mathcal{D}_2(r)} \mathcal{D}_1(r)
\]

for all tuples \( r \). A \( \rightarrow_{\mathcal{D}_2} \mathcal{D}_1 \) is called a (residuated) a-negation of \( \mathcal{D}_1 \) which ranges over \( \mathcal{D}_2 \).

Remark 4.9:

(a) The \( a \)-negations have the following interpretation. If \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) are results of queries \( Q_1 \) and \( Q_2 \), respectively, then \( (\mathcal{D}_2 \boxdiag_a \mathcal{D}_1)(r) \) is a degree to which “\( r \) matches \( Q_2 \) and \( r \) matches \( Q_1 \) at most to degree \( a \)”. This interpretation is due to the fact that in any residuated lattice, \( x \leq a \) is true iff \( x \rightarrow a = 1 \). Thus, \( a \in L \) in (22) can be seen as a threshold degree and for a tuple \( r \), we have \( (\mathcal{D}_2 \boxdiag_a \mathcal{D}_1)(r) = 1 \) (a full match) iff \( r \) fully matches \( Q_2 \) and \( r \) matches \( Q_1 \) at most to degree \( a \). In particular case, for \( a = 0 \), \( (\mathcal{D}_2 \boxtimes \mathcal{D}_1)(r) \) is a degree to which “\( r \) matches \( Q_2 \) and \( r \) matches \( Q_1 \) at most to degree 0”, i.e. “\( r \) matches \( Q_2 \) and \( r \) does not match \( Q_1 \)”. Therefore, \( \mathcal{D}_2 \boxtimes \mathcal{D}_1 \) can be seen as a particular case of a (residuated) difference. Section 8.2 shows another approach to a difference in our model. Altogether, with \( \boxdiag_a \) and in particular with \( \boxdiag_0 = \boxtimes \), we have the ability to express negated compound queries.

(b) Residuated negations satisfy the following laws that can be derived from the properties of \( \rightarrow_{\mathcal{D}} \): \( \mathcal{D}_2 \boxdiag_1 \mathcal{D}_1 = \mathcal{D}_2, 0_R \boxdiag_a \mathcal{D} = 0_R, \mathcal{D} \boxdiag_a 0_R = \mathcal{D}, (\mathcal{D} \boxdiag_a \mathcal{D})(r) \leq a \) for all tuples \( r \), from which we get \( \mathcal{D} \boxtimes \mathcal{D} = 0_R \). If \( \mathcal{L} \) is the two-element Boolean algebra, then \( \boxdiag_0 = \boxtimes \) is the usual difference of relations. For \( a \)-shifts, we can easily show
that $0 \rightarrow_{D_2} D_1 = D_2$ and $1 \rightarrow_{D_2} D_1 = D_1 \otimes D_2$, i.e. the \( \otimes \)-intersection of RDTs can be seen as a special case of a 1-shift.

(c) The \( a \)-shifts can be interpreted analogously as \( a \)-negations in (a), except that \( a \in L \) acts as a threshold prescribing that \( Q_2 \) shall be matched

\[
\text{at least to degree } a \in L. \tag{24}
\]

Indeed, if \( D_1 \) and \( D_2 \) are results of queries \( Q_1 \) and \( Q_2 \), respectively, then \( (a \rightarrow_{D_2} D_1)(r) \) is a degree to which \( r \) matches \( Q_2 \) and \( r \) matches \( Q_1 \) at least to degree \( a \)”. The purpose of \( a \)-shifts is to put emphasis on results that are matched at least to a prescribed degree. Exploiting the fact that \( x \leq a \) is true iff \( x \rightarrow a = 1 \), \( (a \rightarrow_{D_2} D_1)(r) = 1 \) means that \( r \) fully matches \( Q_2 \) and matches \( Q_1 \) at least to degree \( a \).

(d) Note that \( D_2 \boxtimes_a D_1 = D_1 \rightarrow_{D_2} D_3 \) and \( a \rightarrow_{D_2} D_1 = D_3 \rightarrow_{D_2} D_1 \), where \( D_3 \) is any RDT (on the same relation scheme as \( D_1 \) and \( D_2 \)) such that \( D_3(r) = a \) whenever \( D_2(r) > 0 \). As a result, \( a \)-negations and \( a \)-shifts can be seen as particular applications of the ternary residuum introduced by (21). In particular, \( D_2 \boxtimes D_1 = D_1 \rightarrow_{D_2} 0_R \), where \( 0_R \) is the empty RDT on relation scheme \( R \).

Let us emphasize that residuated \( a \)-shifts are new relational operations that emerge in our model and do not have nontrivial counterparts in the classical model where the only possible shifts (considering the two-element Boolean algebra as the structure of degrees) are 0-shifts and 1-shifts which are derivable from the ordinary relational operations, cf. Remark 4.9 (b).

### 4.3. Natural join and cross join

In our model, we can introduce various types of join-like operations which generalize natural joins and general theta-joins from the Codd model. In this section, we introduce the basic type of a join which we call the (equality-based) natural join and in Section 7.3 we show that various similarity-based joins may be obtained from the basic join combined with other relational operations.

As usual, we call tuples \( r \) and \( s \) on relation schemes \( R \) and \( S \), respectively, joinable if \( r(y) = s(y) \) for all \( y \in R \cap S \). Note that if \( r \) and \( s \) are joinable, we may consider their set-theoretical union \( r \cup s \), denoted shortly by \( rs \) and called a join of tuples \( r \) and \( s \), which is a well-defined tuple on relation scheme \( R \cup S \). Therefore, \( rs(y) = r(y) \) if \( y \in R \) and \( rs(y) = s(y) \) if \( y \in S \). By its definition, the join of tuples is commutative, associative, and idempotent operation with \( \emptyset \) (the empty tuple) being its neutral element.

If \( D_1 \) is an RDT on relation scheme \( R \cup S \) and \( D_2 \) is an RDT on relation scheme \( S \cup T \) such that \( R \cap S = R \cap T = S \cap T = \emptyset \) (i.e. \( R \), \( S \), and \( T \) are pairwise disjoint), then the (equality-based) natural join of \( D_1 \) and \( D_2 \) is an RDT \( D_1 \bowtie D_2 \) on relation scheme \( R \cup S \cup T \) defined by

\[
(D_1 \bowtie D_2)(rst) = D_1(rs) \otimes D_2(st), \tag{25}
\]

for each \( r \in \text{Tupl}(R) \), \( s \in \text{Tupl}(S) \), and \( t \in \text{Tupl}(T) \). If \( S = \emptyset \), the natural join defined by (25) is called a cross join.
Remark 4.10:

(a) As in case of the ordinary natural join, $D_1 \bowtie D_2$ represents joins of all joinable tuples from $D_1$ and $D_2$ and, in addition, the ranks of the joined tuples in $D_1 \bowtie D_2$ are computed by aggregating the ranks in $D_1$ and $D_2$ by $\times$. This agrees with the basic interpretation of joins. If $D_1$ and $D_2$ are results of queries $Q_1$ and $Q_2$, respectively, then the rank $(D_1 \bowtie D_2)(rst)$ is a degree to which "rs matches $Q_1$ and st matches $Q_2"$. Obviously, if $D_1$ and $D_2$ are nonranked (e.g. if $L$ is 2), $D_1 \bowtie D_2$ becomes the ordinary natural join.

(b) Natural join of RDTs is obviously a monoidal operation – it is associative since $\times$ is associative, and $1\emptyset$ is its neutral element (see Definition 3.7). Indeed, $1\emptyset$ is an RDT on the relations scheme $\emptyset$, i.e. for any RDT $D$ on $R$, we have

$$(1\emptyset \bowtie D)(r) = 1\emptyset(\emptyset) \times D(r) = 1 \times D(r) = D(r)$$

for all $r \in \text{Tupl}(R)$. In general, for $a\emptyset$, we have

$$(a\emptyset \bowtie D)(r) = a\emptyset(\emptyset) \times D(r) = a \times D(r),$$

i.e. the result of $a\emptyset \bowtie D$ is an RDT of the same scheme as $D$ where each rank is multiplied by degree $a \in L$ (we can call this operation an $a$-contraction). Thus, $0\emptyset$ is an annihilator since $0\emptyset \bowtie D = 0_R$. In addition, natural join is also commutative. This allows us to keep the usual simplified notation for a natural join of several RDTs based on omitting parentheses and disregarding the order of arguments. Hence, we denote the result of the natural join of RDTs $D_i$ ($i \in I$ for a finite $I$) by $\bowtie_{i \in I} D_i$. If in addition $I = \{1, \ldots, n\}$, we write $\bowtie_{i=1}^n D_i$.

(c) If $R = T = \emptyset$, then (25) becomes the $\times$-intersection of RDTs on the relation scheme $S$, see (14). As a consequence, natural joins are not idempotent in general. Namely, $D \bowtie D \subseteq D$ holds but the converse inclusion is true iff $D$ is idempotent, i.e. iff $D \times D = D$. Hence, when considering joins $\bowtie_{i \in I} D_i$ of multiple RDTs, we cannot remove duplicities among all $D_i$ ($i \in I$) before performing the join unless the RDTs are idempotent. In addition, $|D_1 \bowtie D_2| \leq |D_1| \cdot |D_2|$ but the converse inequality does not hold in general because we can have $D_1(rs) = a > 0$, $D_2(st) = b > 0$, and $a \times b = 0$ if $\times$ has zero divisors.

(d) The equality-based natural join introduced by (25) can be used to express an equality-based restriction. In particular, for RDT $D$ on $R$, $y \in R$ and $d \in D_y$, we can consider a natural join $D \bowtie [y:d]$, see Definition 3.8. Then, (25) yields

$$(D \bowtie [y:d])(r) = \begin{cases} D(r), & \text{if } r(y) = d, \\ 0, & \text{otherwise} \end{cases}$$

for all $r \in \text{Tupl}(R)$. Hence, $D \bowtie [y:d]$ can be seen as an equality-based restriction (selection) resulting from $D$ by taking tuples with $y$-values equal to $d$. The original ranks of those tuples in $D$ are preserved.
4.4. Projection and residuated division

In this section, we describe relational operations that can be utilized in expressing compound queries which involve existential and universal quantification like “there is… satisfying Q” and “for all… satisfying Q”. The operations presented here represent our counterparts of the classic projection and division operations.

Recall that the classic projection produces a data table whose tuples are projected onto a subset of the original attributes, meaning that the tuples consist of a subset of attributes of the original relations scheme together with the corresponding values. In this section, we use the usual notion of a projection of a tuple: for a tuple $t \in \text{Tupl}(T)$ and $R \subseteq T$, we let $t(R)$ denote the projection of tuple $t$ onto $R$, i.e. $t(R) \in \text{Tupl}(R)$ such that $(t(R))(y) = t(y)$ for all $y \in R$. An important aspect of the ordinary projection as a relational operation is that it allows to express existential compound queries since $r \in \pi_R(D)$ is interpreted so that “$D$ contains a tuple $t$ which agrees with $r$ on all attributes from $R$”. In terms of projections of tuples, $r \in \pi_R(D)$ means that “there exists $t$ in $D$ such that $r = t(R)$.”

The projection of RDTs we introduce here has the same existential meaning but we have to deal with ranks. Unlike the classical model, we may have an RDT $D$ on relation scheme $T$ and tuples $t$ and $t'$ with common values on all attributes form $R \subseteq T$, i.e. $t(R) = r = t'(R)$ for some $r \in \text{Tupl}(R)$, such that both $t$ and $t'$ have nonzero ranks $D(t) \neq D(t')$. Thus, in general, we may define a rank of $r$ in the projection of $D$ onto $R$ based on ranks of all tuples $t \in \text{Tupl}(T)$ in $D$ that are projected onto $r$ (since $D$ is an RDT, only finitely many of them have nonzero ranks in $D$). In order to keep the existential interpretation of projections, it is desirable to define the degree using the supremum $\lor$ in $L$. Therefore, if $D$ is an RDT on $T$, the projection $\pi_R(D)$ of $D$ onto $R \subseteq T$ is defined by

$$\pi_R(D)(r) = \bigvee_{s \in \text{Tupl}(T \setminus R)} D(rs), \quad (27)$$

for each $r \in \text{Tupl}(R)$.

**Remark 4.11:**

(a) Projections of RDTs defined by $(27)$ can be equivalently restated as follows:

$$\pi_R(D)(r) = \bigvee \{ D(rs) \mid s \in \text{Tupl}(T \setminus R) \}$$

$$= \bigvee \{ D(t) \mid t \in \text{Tupl}(T) \text{ such that } t(R) = r \}. \quad (28)$$

Observe that if $D$ is non-ranked (which is true in particular if $L$ is a two-element Boolean algebra), then $\pi_R(D)$ introduced by $(27)$ becomes the ordinary projection.

(b) There are two special cases of projections. First, if $D$ is an RDT on $T$, for $\pi_T(D)$ and arbitrary $t \in \text{Tupl}(T)$, we get

$$\pi_T(D)(t) = \bigvee_{s \in \text{Tupl}(T \setminus T)} D(ts) = \bigvee_{s \in \emptyset} D(ts) = D(t\emptyset) = D(t).$$

Therefore, $\pi_T(D) = D$. Second, putting $R = \emptyset$ in $(27)$ yields

$$\pi_{\emptyset}(D)(\emptyset) = \bigvee_{t \in \text{Tupl}(T \setminus \emptyset)} D(t\emptyset) = \bigvee_{t \in \text{Tupl}(T)} D(t).$$
In words, $\pi_\emptyset(D)$ is an RDT on $\emptyset$ such that the empty tuple (represented by $\emptyset$) has a rank which is the supremum of ranks of all tuples in $D$, in short

$$\pi_\emptyset(D) = (\bigvee_{t \in \text{Tupl}(T)} D(t)) \emptyset.$$ 

Following the comments before Definition 3.7, $\pi_\emptyset(D)$ can be seen as a single degree (a truth value) from $L$.

As in the classical model, if two (or more) projections are applied in a row, the last one subsumes the previous ones. The following proposition shows properties of projections and their relationship to other operations.

**Proposition 4.12:** For any $D_1, D_2, D$ on $R$ and $a \in L$,

(i) If $R_1 \subseteq R_2$, then $\pi_{R_1}(\pi_{R_2}(D)) = \pi_{R_1}(D)$,

(ii) $\pi_R(D_1 \cup D_2) = \pi_R(D_1) \cup \pi_R(D_2)$,

(iii) $\pi_R(D_1 \cap D_2) \subseteq \pi_R(D_1) \cap \pi_R(D_2)$,

(iv) $\pi_R(D_1 \otimes D_2) \subseteq \pi_R(D_1) \otimes \pi_R(D_2)$,

(v) $\pi_R(a \rightarrow D_2, D_1) \subseteq a \rightarrow \pi_R(D_2) \pi_R(D_1)$.

**Proof:**

(i) If $R_1 \subseteq R_2$, using the fact that $(T \setminus R_2) \cup (R_2 \setminus R_1) = T \setminus R_1$,

$$\left(\pi_{R_1}(\pi_{R_2}(D))\right)(r_1) = \bigvee_{s_1 \in \text{Tupl}(R_2 \setminus R_1)} \left(\pi_{R_2}(D)\right)(r_1 s_1) = \bigvee_{s_1 \in \text{Tupl}(R_2 \setminus R_1)} \bigvee_{s_2 \in \text{Tupl}(T \setminus R_2)} D(r_1 s_1 s_2) = \bigvee_{s \in \text{Tupl}(T \setminus R_1)} D(r_1 s) = (\pi_{R_1}(D))(r_1).$$

(ii) For arbitrary $r \in \text{Tupl}(R)$, we get

$$\left(\pi_R(D_1 \cup D_2)\right)(r) = \bigvee_{s \in \text{Tupl}(T \setminus R)} (D_1(rs) \vee D_2(rs)) = \bigvee_{s \in \text{Tupl}(T \setminus R)} D_1(rs) \vee \bigvee_{s \in \text{Tupl}(T \setminus R)} D_2(rs).$$

Hence $\left(\pi_R(D_1 \cup D_2)\right)(r) = \left(\pi_R(D_1)\right)(r) \vee \left(\pi_R(D_2)\right)(r) = (\pi_R(D_1) \cup \pi_R(D_2))(r)$.

(iii) A consequence of the monotony of projection.

(iv) Using the distributivity of $\otimes$ over $\vee$, for any $r \in \text{Tupl}(R)$,

$$\left(\pi_R(D_1 \otimes D_2)\right)(r) = \bigvee_{s \in \text{Tupl}(T \setminus R)} (D_1 \otimes D_2)(rs) = \bigvee_{s \in \text{Tupl}(T \setminus R)} (D_1(rs) \otimes D_2(rs)) \leq \bigvee_{s_1 \in \text{Tupl}(T \setminus R)} \bigvee_{s_2 \in \text{Tupl}(T \setminus R)} (D_1(r s_1) \otimes D_2(r s_2)) = \bigvee_{s_1 \in \text{Tupl}(T \setminus R)} (D_1(r s_1) \otimes \bigvee_{s_2 \in \text{Tupl}(T \setminus R)} D_2(r s_2)) = \bigvee_{s_1 \in \text{Tupl}(T \setminus R)} (D_1(r s_1) \otimes D_2(r s_2)) = (\pi_R(D_1))(r) \otimes (\pi_R(D_2))(r) = (\pi_R(D_1 \otimes D_2))(r).$$
(v) Using the monotony of $\otimes$ and the monotony of $\rightarrow$ in the second argument,

\[
(\pi_R(a \rightarrow \mathcal{D}_2 \mathcal{D}_1))(r) = \bigvee_{s \in \text{Tupl}(T \setminus R)} (a \rightarrow \mathcal{D}_2(rs) \mathcal{D}_1(rs)) \\
= \bigvee_{s \in \text{Tupl}(T \setminus R)} (\mathcal{D}_2(rs) \otimes (a \rightarrow \mathcal{D}_1(rs))) \\
\leq \bigvee_{s \in \text{Tupl}(T \setminus R)} (\mathcal{D}_2(rs) \otimes (a \rightarrow \bigvee_{s \in \text{Tupl}(T \setminus R)} \mathcal{D}_1(rs))) \\
= \bigvee_{s \in \text{Tupl}(T \setminus R)} \mathcal{D}_2(rs) \otimes (a \rightarrow \bigvee_{s \in \text{Tupl}(T \setminus R)} \mathcal{D}_1(rs)),
\]

the last term is equal to $a \rightarrow (\pi_R(\mathcal{D}_2))(r) (\pi_R(\mathcal{D}_1))(r)$, finishing the proof.

\[\square\]

The following theorem shows the basic relationship between projections and joins which are analogous to that from the Codd model. In the theorem, we use the following notion: for an RDT $\mathcal{D}$, we define its $n$th power, denoted $\mathcal{D}^n$ by

\[
\mathcal{D}^n = \begin{cases} 
1\emptyset, & \text{if } n = 0, \\
\mathcal{D} \triangledown \mathcal{D}^{n-1}, & \text{otherwise.} 
\end{cases} \tag{29}
\]

for all nonnegative integers $n$. Clearly, the $n$th power of $\mathcal{D}$ for $n \geq 1$ can equivalently be defined using the $\otimes$-intersection instead of the natural join. On the other hand, definition (29) is more general because $\mathcal{D}^0 = 1\emptyset$ which is beneficial to keep for technical purposes.

**Theorem 4.13:** Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be RDTs on relation schemes $R \cup S$ and $S \cup T$ such that $R \cap S = R \cap T = S \cap T = \emptyset$. Furthermore, let $\{\mathcal{D}_i | i \in I\}$ be a finite set of RDTs on $R_i (i \in I)$, and let $\mathcal{D}$ be an RDT on $R = \bigcup_{i \in I} R_i$. Then,

(i) $\pi_{R \cup S}(\mathcal{D}_1 \triangledown \mathcal{D}_2) = \mathcal{D}_1 \triangledown \pi_S(\mathcal{D}_2)$,
(ii) $\pi_{R_i}(\triangledown_{j \in I} \mathcal{D}_j) \subseteq \mathcal{D}_i$ for all $i \in I$,
(iii) $\mathcal{D}^{[I]} \subseteq \triangledown_{i \in I} \pi_{R_i}(\mathcal{D})$,
(iv) if $\mathcal{D}$ is idempotent, then $\mathcal{D} \subseteq \triangledown_{i \in I} \pi_{R_i}(\mathcal{D})$.

**Proof:**

(i) For all $r \in \text{Tupl}(R)$ and $s \in \text{Tupl}(S)$, we have

\[
(\pi_{R \cup S}(\mathcal{D}_1 \triangledown \mathcal{D}_2))(rs) = \bigvee_{t \in \text{Tupl}(T)} (\mathcal{D}_1 \triangledown \mathcal{D}_2)(rst) \\
= \bigvee_{t \in \text{Tupl}(T)} (\mathcal{D}_1(rs) \otimes \mathcal{D}_2(st)) \\
= \mathcal{D}_1(rs) \otimes \bigvee_{t \in \text{Tupl}(T)} \mathcal{D}_2(st) = \mathcal{D}_1(rs) \otimes (\pi_S(\mathcal{D}_2))(s) \\
= (\mathcal{D}_1 \triangledown \pi_S(\mathcal{D}_2))(rs),
\]

which proves that $\pi_{R \cup S}(\mathcal{D}_1 \triangledown \mathcal{D}_2) = \mathcal{D}_1 \triangledown \pi_S(\mathcal{D}_2)$.

(ii) Using (i) together with the fact that $\triangledown$ is associative and commutative, for arbitrary $i \in I$, we can write

\[
\pi_{R_i}(\triangledown_{j \in I} \mathcal{D}_i) = \pi_{R_i}(\mathcal{D}_i \triangledown (\triangledown_{j \in I \setminus \{i\}} \mathcal{D}_j)) = \mathcal{D}_i \triangledown \pi_S(\triangledown_{j \in I \setminus \{i\}} \mathcal{D}_j),
\]
where $S = \bigcup_{j \in \Gamma \setminus \{i\}} (R_i \cap R_j)$. Since $S \subseteq R_i$, it follows that $D_i \bowtie \pi_S( \bowtie_{j \in \Gamma \setminus \{i\}} D_j)$ is an RDT on $R_i$ and in consequence, $D_i \bowtie \pi_S( \bowtie_{j \in \Gamma \setminus \{i\}} D_j) \subseteq D_i$, finishing the proof of (ii).

(iii) Note that $D^{[i]}$ denotes the $n$th power of $D$, where $n$ is the number of elements in $I$. Also note that for $I = \emptyset$, the claim is trivial since both sides of the inequality are $1_\emptyset$ (a “join of no RDTs” equals $1_\emptyset$ since it is the neutral element). Using the fact that if $(\bowtie_{i \in I} \pi_{R_i}(D))(r) > 0$, then $r$ is a join of tuples $r_i \in \text{Tupl}(R_i)$ ($i \in I$), we may write

$$(\bowtie_{i \in I} \pi_{R_i}(D))(r) = \bigotimes_{i \in I} (\pi_{R_i}(D))(r_i) = \bigotimes_{i \in I} \bigvee_{s_i \in \text{Tupl}(R \setminus R_i)} D(r_i s_i),$$

where $\bigotimes_{i \in I} \cdots$ denote $\bowtie$-multiplications of degrees indexed by elements from $I$. Now, take $t_i \in \text{Tupl}(R \setminus R_i)$ so that $r = r_i t_i$ for all $i \in I$ and observe that the previous equality yields

$$D^{[I]}(r) = \bigotimes_{i \in I} D(r_i t_i) \leq \bigotimes_{i \in I} \bigvee_{s_i \in \text{Tupl}(R \setminus R_i)} D(r_i s_i) = (\bowtie_{i \in I} \pi_{R_i}(D))(r)$$

for $r \in \text{Tupl}(R)$, i.e. $D^{[I]} \subseteq \bowtie_{i \in I} \pi_{R_i}(D)$.

(iv) A direct consequence of (iii) and $D^n = D$ $(n \geq 1)$ if $D$ is idempotent.

□

**Remark 4.14:**

(a) Theorem 4.13 (i) represents mutual relationship between natural joins and projections that is related to the notion of a semijoin. Recall that the ordinary (natural) semijoin can be seen as a natural join of two arguments projected onto attributes of the first argument. In the same way, one can express the ordinary semijoin as a join of the first argument and projection of the second argument onto the common set of attributes of both the arguments. In our setting, the same law applies due to Theorem 4.13 (i). Thus, for RDTs $D_1$ and $D_2$ on $R \cup S$ and $S \cup T$ such that $R \cap S = R \cap T = S \cap T = \emptyset$, we may define

$$D_1 \bowtie D_2 = \pi_{R \cup S}(D_1 \bowtie D_2) = D_1 \bowtie \pi_S(D_2),$$

which is called a *(natural equality-based) semijoin* of $D_1$ and $D_2$ (in this order, as in the ordinary case, the operation is not commutative). Following the interpretation of natural joins and projections, if $D_1$ and $D_2$ are results of queries $Q_1$ and $Q_2$, respectively, then $(D_1 \bowtie D_2)(rs)$ may be interpreted as “a degree to which $rs$ matches $Q_1$ and $st$ matches $Q_2$ for some $t$”, i.e. analogously as for the ordinary semijoins, $D_1 \bowtie D_2$ may be regarded as an RDT expressing *joinability degrees* of tuples from $D_1$: $(D_1 \bowtie D_2)(rs)$ is “a degree to which $rs$ matches $Q_1$ and is joinable with some tuple matching $Q_2$”, cf. Date (2005).

(b) In general, the inclusion in Theorem 4.13 (iv) does not hold for nonidempotent RDTs. For instance, $\pi_{0.5}(0.5 \bowtie 0.5) = 0.25 \bowtie 0.5 \not\approx 0.5_\bowtie$ if $\bowtie$ is the Goguen multiplication. Note that the Codd model, the converse inclusions to that from Theorem 4.13 (ii) and (iv) are used to define notions of a complete join and a nonloss decomposition. We can introduce analogous notions in our model, but we
take a more general approach and introduce approximate joins and decompositions based on partial containment introduced later in this subsection.

We now turn our attention to residuated division that can be utilized in expressing compound queries which involve universal quantification. In the classical model, there is no need to include the division among the fundamental operations since it can be expressed by means of projections, joins, and differences. This is a consequence of the well-known fact that in the classical predicate logic the universal quantifier can be expressed in terms of the existential one and negation. Logics based on residuated lattices do not have this property, i.e. the quantifiers are not mutually definable in general. We therefore consider division as a fundamental operation and define it using the derived ternary operation \( \rightarrowtriangle \) on \( L \) and aggregate the ranks using general infimum \( \land \) in \( L \).

Let \( D_1 \) be an RDT on \( R \), let \( D_2 \) be an RDT on \( S \subseteq R \), and let \( D_3 \) be an RDT on \( T = R \setminus S \). Then, a division \( D_1 \div D_3 D_2 \) of \( D_1 \) by \( D_2 \) which ranges over \( D_3 \) is an RDT on \( T \) defined by

\[
(D_1 \div D_3 D_2)(t) = \bigwedge_{s \in \text{Tupl}(S)} (D_3(s) \rightarrowtriangle D_1(st)) ,
\]

for each \( t \in \text{Tupl}(T) \).

**Remark 4.15:**

(a) Residuated division is well defined. Indeed, using (16), we can rewrite (31) as follows

\[
(D_1 \div D_3 D_2)(t) = \bigwedge_{s \in \text{Tupl}(S)} (D_3(s) \land (D_2(s) \rightarrow D_1(st))),
\]

from which it follows that only finitely many tuples \( t \in \text{Tupl}(T) \) are assigned nonzero degrees because \( \text{Tupl}(S) \) is nonempty and \( D_3 \) is finite. In addition, the degree (31) can always be computed by performing finitely many applications of operations of \( L \) because the infimum in (31) which goes over all tuples over \( S \) can be reduced to an infimum going over all tuples over \( S \) that have nonzero ranks in \( D_2 \), see Proposition 4.4 (iii). Hence,

\[
(D_1 \div D_3 D_2)(t) = D_3(t) \land \bigwedge_{D_2(s) > 0} (D_3(s) \land (D_2(s) \rightarrow D_1(st))).
\]

If \( L \) is 2 and if \( D_3 \) is the projection of \( D_1 \) onto \( T \), then (31) becomes the ordinary division from the Codd model.

(b) Note that results of relational division are sometimes called quotients [Maier (1983)], i.e. \( D_1 \div D_3 D_2 \) defined by (31) can be called a quotient of \( D_1 \) divided by \( D_2 \) (over \( D_3 \)). The interpretation of division is analogous to its ordinary counterpart. For instance, if \( D_2 \) defines reliable suppliers, \( D_3 \) defines solvent customers, and \( D_1 \) defines which suppliers are frequently used by customers, then \( D_1 \div D_3 D_2 \) represents solvent customers frequently using all reliable suppliers. It is worth mentioning that in this example, all three arguments to the division can naturally be considered as ranked data tables which either resulted by similarity-based queries or were created by an expert (e.g. assessing reliability of suppliers) because all tuples in “suppliers”, “customers”, and the record of “supplies-to-customer” can be considered as ranked by degrees from \( L \) according to how they satisfy the conditions of “reliability”, “solvency”, and “frequency”.
Note that in the classic relational model of data, there does not seem to be an agreement on the right definition of relational division. Indeed, there exist multiple definitions, some of which are easily reducible to other. In our setting, there are also possibilities to start with a different definition. The recent paper Vaverka and Vychodil (2016) studies the relationship between the existing approaches to the classic as well as the similarity-based operations of division.

Consider RDTs $D_1$, $D_2$, and $D_3$ on relation schemes $R$, $S \subseteq R$, and $T = R \setminus S$, respectively. We can consider the following borderline cases of division. If $S = R$ and thus $T = \emptyset$, then using (16), we can write (31) as follows:

$$
(D_1 \div D_3 D_2)(\emptyset) = \bigwedge_{r \in \text{Tupl}(R)} (D_2(r) \rightarrow D_3(\emptyset) D_1(r\emptyset)) = \bigwedge_{r \in \text{Tupl}(R)} (a \otimes (D_2(r) \rightarrow D_1(r))),
$$

where $a = D_3(\emptyset)$, i.e. $D_3 = a_\emptyset$. Thus, in this case, we may write $D_1 \div^a D_2$ instead of $D_1 \div D_3 D_2$, where $a = D_3(\emptyset)$. In particular, for $a = 1$, i.e. if $D_3$ is $1_\emptyset$, we obtain

$$
(D_1 \div^1 D_2)(\emptyset) = \bigwedge_{r \in \text{Tupl}(R)} (D_2(r) \rightarrow D_1(r)),
$$

which is in fuzzy relational systems known as the *subsethood degree* of L-sets. In this case, the subsethood degree of $D_2$ in $D_1$. Note, however, then the result is not the degree of subsethood but rather, an RDT (on the empty relation scheme) representing the degree. Nevertheless, residuated division enables us to formalize two important concepts: subsethood (inclusion) degrees and similarity degrees of RDTs in our model. Indeed, for any RDTs $D_1$ and $D_2$ on the same relation scheme, we may introduce

$$
S(D_1, D_2) = (D_1 \div D_2)(\emptyset),
$$

(33)

$$
E(D_1, D_2) = S(D_1, D_2) \land S(D_2, D_1).
$$

(34)

In what follows, (33) is called a *subsethood degree* of $D_1$ in $D_2$; (34) is called a *similarity degree* of $D_1$ and $D_2$. If $D_1$ and $D_2$ are interpreted as answers to similarity-based queries $Q_1$ and $Q_2$, respectively, then (33) is the degree to which each answer to $Q_1$ is an answer to $Q_2$ and (34) is a degree to which queries $Q_1$ and $Q_2$ have similar answers (in a particular database instance).

The second borderline case of residuated division results if $S = \emptyset$ and hence $T = R \setminus \emptyset = R$. In this case:

$$
(D_1 \div^D D_2)(r) = \bigwedge_{s \in [\emptyset]} (D_2(s) \rightarrow D_3(r) D_1(rs)) = D_3(r) \otimes (D_2(\emptyset) \rightarrow D_1(r)) = a \rightarrow D_3 D_1,
$$

where $a = D_2(\emptyset)$. Hence, the previous equality shows that residuated shifts (23) introduced before can be seen as particular cases of residuated divisions.

The notions of subsethood and similarity degrees (33) and (34) are crucial in our model because they can be used to define exact and approximate integrity constraints. More on this issue will be presented in Section 4.8 and the sequel of this paper. In addition, we
may use (33) and (34) to introduce properties of RDTs related to projections and joins. Namely, in the classical model, the notions of complete joins and nonloss decompositions are defined in terms of the ordinary subsethood. We show that analogous notions can be introduced in our model using subsethood and similarity degrees.

Let $D_i$ be RDTs on relation schemes $R_i$ ($i \in I$ for finite $I$). We say that RDTs $D_i$ ($i \in I$) have an (approximate) $a$-join if

$$S(D_i, \pi_{R_i}((\triangleright j \in I \ D_j))) \geq a$$

for all $i \in I$. Furthermore, a 1-join is called a complete join (in other words, RDTs $D_i$ join completely). In addition, the degree

$$Jnd((D_i \mid i \in I)) = \bigwedge_{i \in I} S(D_i, \pi_{R_i}((\triangleright j \in I \ D_j)))$$

is called a degree of joinability of RDTs $D_i$ ($i \in I$).

**Remark 4.16:**

(a) A moment’s reflection shows that $Jnd((D_i \mid i \in I))$ is the greatest degree $a \in L$ such that $D_i$ ($i \in I$) are $a$-joinable and that any RDTs are trivially 0-joinable. Furthermore, if $D_i$ ($i \in I$) are $a$-joinable, then $D_i$ ($i \in I$) are $b$-joinable for all $b \leq a$. The 1-joinability corresponds to the classic notion of a complete joinability since due to Theorem 4.13 (ii), it means that $D_i$ is equal to $\pi_{R_i}((\triangleright j \in I \ D_j))$ for all $i \in I$, i.e. each $D_i$ can be retrieved from the join by the corresponding projection.

(b) The general joinability degrees in our model are capable of expressing that RDTs $D_i$ ($i \in I$) are almost completely joinable, e.g. $a$-joinable for a high degree $a \in L$. In other words, from the join $\triangleright j \in I \ D_j$ one can retrieve $\pi_{R_i}((\triangleright j \in I \ D_j))$ which is similar to $D_i$ to degree at lest $a \in L$. If $a$ is sufficiently high, we may interpret $\pi_{R_i}((\triangleright j \in I \ D_j))$ as a sufficiently close approximation of the original data table $D_i$. Although not directly applicable when a complete join is required, the approximate join can be interesting from data analytical point of view – instead of all $D_i$ one may store just the join $\triangleright j \in I \ D_j$ and still be able to get sufficiently close approximations of the original ranked data tables.

Let $D$ be an RDT on relation schemes $R = \bigcup_{i \in I} R_i$ where $I$ is finite. We define a degree of decomposability of $D$ with respect to $R_i$ ($i \in I$) as follows:

$$Dcd(D, \{R_i \mid i \in I\}) = E(D, \triangleright j \in I \pi_{R_i}(D)).$$

In addition, we say that $D$ has an (approximate) $a$-decomposition with respect to $R_i$ ($i \in I$) if $Dcd(D, \{R_i \mid i \in I\}) \geq a$. Furthermore, a 1-decomposition is called a nonloss decomposition.

**Remark 4.17:** Similar comments as in Remark 4.16 can be made in case of approximate decomposability. Instead of storing (a large) RDT $D$, one may store its (smaller) projections $\pi_{R_i}(D)$ provided that their join is sufficiently close to the original data $D$. Let us note that, once again, the concepts of approximate joinability and decomposability have naturally appeared in our model just by considering a general structure of ranks L and replacing the bivalent subsethood relation by its graded counterpart (33) which appears in our model as an instance of division. Therefore, the necessary framework for exploring approximate joinability and decompositions is inherent in the model from the very beginning.
The following assertion provides an alternative characterization of divisions if the RDT which serves as a range is nonranked.

**Theorem 4.18:** Let \( D_1, D_2, \) and \( D_3 \) be RDTs on relation schemes \( R, S \subseteq R, \) and \( T = R \setminus S, \) respectively. Then the following hold:

(i) \( D_1 \div D_3 D_2 \subseteq D_3; \)

(ii) \( (D_1 \div D_3 D_2) \bowtie D_2 \subseteq D_1; \)

(iii) if \( D_3 \) is nonranked, then \( D_1 \div D_3 D_2 \) is the greatest element of

\[
S_{D_1,D_2,D_3} = \{ D \subseteq D_3 | D \bowtie D_2 \subseteq D_1 \}. \tag{38}
\]

**Proof:**

(i) follows immediately from (31) using the fact that both \( \otimes \) and \( \wedge \) are monotone. Observe that the join in (ii) is a cross join. Thus,

\[
((D_1 \div D_3 D_2) \bowtie D_2)(st) = (D_1 \div D_3 D_2)(t) \otimes D_2(s)
\]

\[
= \wedge_{s' \in \text{Tupl}(S)} (D_3(t) \otimes (D_2(s') \rightarrow D_1(s't))) \otimes D_2(s)
\]

\[
\leq (D_2(s) \rightarrow D_1(st)) \otimes D_2(s) \leq D_1(st)
\]

for all \( s \in \text{Tupl}(S) \) and \( t \in \text{Tupl}(T). \) Now, from (i) and (ii), we conclude that the system \( S_{D_1,D_2,D_3} \) of RDTs is nonempty. Suppose now that \( D_3(t) \in \{0,1\} \) for all \( t \in \text{Tupl}(T) \) and take any \( D \in S_{D_1,D_2,D_3}. \) In order to prove (iii), it suffices to check that \( D \subseteq D_1 \div D_3 D_2. \) Take any \( t \in \text{Tupl}(T). \) If \( D_3(t) = 0, \) then by \( D \subseteq D_3 \) we get \( D(t) = 0, \) which is a trivial case. If \( D_3(t) = 1, \) from \( D \bowtie D_2 \subseteq D_1, \) we have \( D(t) \otimes D_2(s) \leq D_1(st) \) for all \( s \in \text{Tupl}(S). \) Using adjointness, for all \( s \in \text{Tupl}(S), \)

\[
D(t) \leq D_2(s) \rightarrow D_1(st) = 1 \otimes (D_2(s) \rightarrow D_1(st)) = D_3(t) \otimes (D_2(s) \rightarrow D_1(st)) = (D_1 \div D_3 D_2)(t),
\]

which concludes the proof of (iii).

\[\square\]

### 4.5. Similarity-based restrictions

The relational operations considered in the previous subsections do not utilize similarities of values on domains. In fact, all the operations, when applied to nonranked RDTs, yield nonranked RDTs. We now turn our attention to the similarity-based restriction which is an interesting counterpart to the ordinary restriction (selection). This operation involves similarities on domains and is the primary operation producing ranked data tables from nonranked ones by imposing similarity-based restrictions on data. In fact, our restriction can be seen as an operation which selects from a data table all tuples which approximately match a given condition. In the basic setting, we consider atomic conditions written as

\[
y \approx z
\]

(39)
and saying that the value of the attribute \( y \) should be similar (i.e. approximately equal) to the value of \( z \), where \( z \) is allowed to be either (i) an attribute of the same type as \( y \), or (ii) a value which belongs to the domain of \( y \). Evidently, (i) is intended to be used for comparing similarity of values of two attributes (e.g. queries like “Houses whose last renovation date is approximately equal to their sale date”) whereas (ii) serves for comparing values of attribute values with a fixed element of a domain (e.g. queries like “Houses located approximately in Palo Alto”). As we shall see later in this section, (ii) can be expressed by (i), projections, and natural joins. We therefore begin with the formalization of (i).

Let \( \Lambda = (\Lambda_1, \lambda) \) be a type declaration for \( Y \) and \( C \) and let \( D \) be an RDT on relation scheme \( R \subseteq Y \) over domains with similarities \( \{ (D_y, \approx_y) \mid y \in Y \} \) which respect \( \Lambda \). We keep this assumption throughout the rest of this subsection. Now, for attributes \( y_1, y_2 \in R \) such that \( \lambda(y_1) = \lambda(y_2) \), we define the *similarity-based restriction* \( \sigma_{y_1 \approx y_2}(D) \) of \( D \) by \( y_1 \approx y_2 \) which is an RDT on \( R \) defined by

\[
(\sigma_{y_1 \approx y_2}(D))(r) = D(r) \otimes r(y_1) \approx_{y_1} r(y_2),
\]

for all \( r \in \text{Tupl}(R) \).

**Remark 4.19:**

(a) Obviously, (40) is well defined due to the fact that the domain similarities \( \approx_{y_1} \) and \( \approx_{y_2} \) are the same because \( y_1 \) and \( y_2 \) have the same types and \( D \) respects \( \Lambda \), cf. Definition 3.11.

(b) Considering \( D \) as a result of query \( Q \), the rank of \( r \) in \( \sigma_{y_1 \approx y_2}(D) \) can be interpreted as a degree to which “\( r \) matches the query \( Q \) and in addition the \( y_1 \)-value of \( r \) is similar to the \( y_2 \)-value of \( r \)”. 

(c) As for the ordinary restriction, \( \sigma_{y_1 \approx y'_1}(\sigma_{y_2 \approx y'_2}(D)) = \sigma_{y'_2 \approx y'_1}(\sigma_{y_1 \approx y'_1}(D)) \) which can be extended to arbitrary many restrictions. Indeed, due to the associativity and commutativity of \( \otimes \), we may write

\[
\sigma_{y_1 \approx y'_1 \ldots y_n \approx y'_n}(D) = \sigma_{y_1 \approx y'_1}(\ldots(\sigma_{y_n \approx y'_n}(D))\ldots).
\]  

Directly from (40) and using the associativity of \( \otimes \),

\[
(\sigma_{y_1 \approx y'_1 \ldots y_n \approx y'_n}(D))(r) = D(r) \otimes r(y_1) \approx_{y_1} r(y'_1) \otimes \ldots \otimes r(y_n) \approx_{y_n} r(y'_n). \tag{42}
\]

Since \( \otimes \) is not idempotent in general, duplicate restrictions cannot be eliminated without altering the query result. That means, \( \sigma_{y_1 \approx y_2}(\sigma_{y_1 \approx y_2}(D)) \subseteq \sigma_{y_1 \approx y_2}(D) \) but the converse inclusion does not hold in general, cf. Remark 4.10(c).

We now turn our attention to restrictions which compare \( y \)-values of tuples with a fixed value \( d \) from a domain. If \( d \in D_y \), we may introduce a restriction of \( D \) by \( y \approx d \) as follows:

\[
(\sigma_{y \approx d}(D))(r) = D(r) \otimes r(y) \approx_{y} d.
\]  

Clearly, the rank of each tuple in such \( \sigma_{y \approx d}(D) \) is well defined and \( \sigma_{y \approx d}(D) \) generalizes the ordinary equality-based restriction. Let us see that (43) is a derived operation. First, we may take \( y' \notin R \) such that \( \lambda(y) = \lambda(y') \) because there are infinitely many attributes having
the same type as \( y \) and \( R \) is finite. Then, it is immediate that

\[
\sigma_{y \approx d}(\mathcal{D}) = \pi_R(\sigma_{y \approx y'}(\mathcal{D} \bowtie [y':d])).
\] (44)

Put in words, \( \mathcal{D} \) is extended by a new attribute whose \( y' \)-value is \( d \) in all tuples (the result of \( \mathcal{D} \bowtie [y':d] \)), then we perform the similarity-based restriction as in (40), and we project the result back onto \( R \) to get rid of the auxiliary attribute \( y' \). Therefore, (40) is indeed fundamental while (43) shall be considered derived.

The following assertion shows selected properties of the similarity-based restriction and its relationship to the other relational operations (in the assertion we abbreviate \( y_1 \approx y_2 \) by \( \theta \) if the names of attributes are not further used).

**Proposition 4.20:** The following conditions are all true provided that both left-hand and right-hand sides of the conditions exist:

(i) \( \pi_S(\sigma_{y \approx z}(\mathcal{D})) = \sigma_{y \approx z}(\pi_S(\mathcal{D})) \) if \( \mathcal{D} \) is an RDT on \( R \) and \( R \cap \{y, z\} \subseteq S \),

(ii) \( \sigma_{y \approx z}(\mathcal{D}_1 \bowtie \mathcal{D}_2) = \sigma_{y \approx z}(\mathcal{D}_1) \bowtie \sigma_{y \approx z}(\mathcal{D}_2) \) if \( \mathcal{D}_1 \bowtie \mathcal{D}_2 \) is an RDT on \( R_1 \) and \( \{y, z\} \cap R_2 = \emptyset \),

(iii) \( \sigma_\theta(\mathcal{D}_1 \cup \mathcal{D}_2) = \sigma_\theta(\mathcal{D}_1) \cup \sigma_\theta(\mathcal{D}_2) \),

(iv) \( \sigma_\theta(\mathcal{D}_1 \cap \mathcal{D}_2) \subseteq \sigma_\theta(\mathcal{D}_1) \cap \mathcal{D}_2 \),

(v) \( \sigma_\theta(\mathcal{D}_1 \otimes \mathcal{D}_2) = \sigma_\theta(\mathcal{D}_1) \otimes \mathcal{D}_2 \),

(vi) \( \mathcal{D}_1 \rightarrow_\theta \mathcal{D}_3 \mathcal{D}_2 = \sigma_\theta(\mathcal{D}_1 \rightarrow_\mathcal{D}_3 \mathcal{D}_2) \).

If \( L \) is prelinear or divisible, then

(vii) \( \sigma_\theta(\mathcal{D}_1 \cap \mathcal{D}_2) = \sigma_\theta(\mathcal{D}_1) \cap \sigma_\theta(\mathcal{D}_2) \),

(viii) \( \mathcal{D}_1 \rightarrow_\theta \mathcal{D}_3 \mathcal{D}_2 = \sigma_\theta(\mathcal{D}_1 \rightarrow_\mathcal{D}_3 \mathcal{D}_2) \).

**Proof:** Properties (i)–(vi) follow directly from the monotony of \( \cap \) and associativity and commutativity of \( \otimes \) together with the fact that \( \otimes \) is distributive over \( \cup \). In addition, if \( L \) is prelinear or divisible, (vii) and (viii) are consequences of the fact that \( (a \land b) \otimes c = (a \otimes c) \land (b \otimes c) \) holds for all \( a, b, c \in L \).

**Remark 4.21:**

(a) Typical use of similarity-based restrictions is to obtain ranked RDTs from nonranked ones as we have mentioned. Indeed, if \( \mathcal{D} \) is a nonranked RDT and \( d \in D_y \), then under the notation of (43),

\[
(\sigma_{y \approx d}(\mathcal{D}))(r) = \begin{cases} r(y) \approx_y d, & \text{if } \mathcal{D}(r) = 1, \\ 0, & \text{otherwise}, \end{cases}
\]

i.e. the rank of \( r \) in \( \sigma_{y \approx d}(\mathcal{D}) \) is a degree to which “\( r \) from \( \mathcal{D} \) matches condition posed by \( y \approx d \)”, cf. the meaning of equality-based restrictions (26).

(b) As in Remark 4.10, it is tempting to express similarity-based restrictions via natural joins. For given \( \mathcal{D} \), this can be done by materializing the similarity \( L \)-relation involved in the query. Indeed, for \( \sigma_{y_1 \approx y_2}(\mathcal{D}) \), we may consider an RDT \( \mathcal{D}_{y_1 \approx y_2} \) on \( \{y_1, y_2\} \) by putting

\[
\mathcal{D}_{y_1 \approx y_2}(r(\{y_1, y_2\})) = \begin{cases} r(y_1) \approx_{y_1} r(y_2), & \text{if } \mathcal{D}(r) > 0, \\ 0, & \text{otherwise}, \end{cases}
\]
for all \( r \in \text{Tup}(R) \), where \( r([y_1, y_2]) \) denotes the projection of \( r \) onto \([r_1, r_2]\), see Section 4.4. Thus, \( D_{y_1\approx y_2} \) can be seen as an RDT which materializes a “finite part” of \( \approx_{y_1} \). It is easy to see that \( \sigma_{y_1\approx y_2}(D) = D \Join D_{y_1\approx y_2} \). Hence, to some extent, similarities on domains can be represented by ranked data tables and similarity-based restrictions are expressible as natural equality-based joins of RDTs. Note, however, that in general, domains are infinite and we can only materialize their finite subsets (\( D_{y_1\approx y_2} \) depends on \( D \)). Therefore, the domain similarity is not a redundant notion and it requires a special treatment in our model.

Note that we introduced the similarity-based restrictions as restrictions using a simple atomic conditions of the form (39) because our intention is to present a minimal relational algebra in our model. In Section 7.1, we present more general restrictions, where the similarity-based condition may be given by more complex formulas involving similarity-based conditions. In addition, in Section 7.3 we show that similarity-based restrictions can be used to define various types of similarity-based joins. An important aspect is that all the operations from Section 7 are derivable from the basic relational operations, i.e. the present similarity-based restriction together with the other relational operations is powerful enough to express the other types of restrictions and joins.

4.6. Kernel and support

We now introduce a couple of operations which produce nonranked RDTs from general ones. In this sense, the operations play an opposite role to similarity-based restrictions and appear if users require yes/no answers. Although we introduced the model to support imperfect matches, it is desirable to have operations in our algebra that can force only exact matches whenever it is required by a situation. Utilizing the operations of kernel and support introduced in this section, one can flexibly combine similarity-based and equality-based querying within one formal model.

Intuitively, there are two immediate ways to convert an RDT \( D \) into a nonranked one. First, one may want to see only the “full matches in \( D \)”, which means that all tuples with ranks lower than 1 are disregarded. Second, one may want to convert all nonzero ranks in \( D \) to 1s, i.e. to treat all partial matches as full matches.

Formally, for any RDT \( D \) on relation scheme \( R \), the kernel \( \Delta(D) \) and support \( \nabla(D) \) of \( D \) are RDTs on \( R \) defined by

\[
(\Delta(D))(r) = \begin{cases} 
1, & \text{if } D(r) = 1, \\
0, & \text{otherwise},
\end{cases} \quad (45)
\]

\[
(\nabla(D))(r) = \begin{cases} 
1, & \text{if } D(r) > 0, \\
0, & \text{otherwise},
\end{cases} \quad (46)
\]

for all \( r \in \text{Tup}(R) \). If \( D \) is interpreted as a result of query \( Q \), the kernel \( \Delta(D) \) of \( D \) is an RDT which consists only of the full matches of \( Q \) whereas \( \nabla(D) \) is an RDT which consists of all tuples that match \( Q \) to a nonzero degree, i.e. at least partially. Clearly, \( \Delta(D) \subseteq D \subseteq \nabla(D) \) for any \( D \).
Remark 4.22:

(a) The kernel is in fact an interior operator on the set of all RDTs on R because it is contractive, monotone, and idempotent. In addition, $\Delta(D)$ is the greatest nonranked RDT such that $\Delta(D) \subseteq D$. Dually, the support is a closure operator (it is extensive, monotone, and idempotent) and $\nabla(D)$ is the least nonranked RDT such that $D \subseteq \nabla(D)$. Naturally, there are other ways to express nonranked RDTs from general ones (e.g. convert all ranks above $a \in L$ to 1 and disregard the rest) but the kernel and support are two borderline cases of all such operations which satisfy monotony and idempotency – two basic conditions that may be accepted for any reasonable (and truth-functional) operation of this type.

(b) Besides being useful if one wants to “remove ranks” either for the purpose of presentation of results with no distraction caused by the ranks (one may not want to confront the end users or application client with the explicit ranks) or for the sake of compatibility with database or application subsystems that are not rank aware, the kernel and support play an important technical role in the algebra. For instance, for any RDT $D$ on $R$ and attribute $y \in R$, the result of $\nabla(\pi_y(D))$ can be seen as a nonranked RDT representing the active domain of $y$ in $D$, i.e. the collection of all $y$-values in $D$, represented by a nonranked table which, by a slight abuse of notation, can be seen as a finite subset of $D_y$. The notations $\Delta(\pi_y(D))$ and $\nabla(\pi_y(D))$ are inspired by unary connectives on Gödel chains proposed by Baaz (1996).

Proposition 4.23: The following conditions are all true provided that both left-hand and right-hand sides of the conditions exist:

\begin{enumerate}
  \item[(i)] $\Delta(D_1 \otimes D_2) = \Delta(D_1) \otimes \Delta(D_2)$, $\nabla(D_1 \otimes D_2) \subseteq \nabla(D_1) \otimes \nabla(D_2)$,
  \item[(ii)] $\Delta(D_1 \cap D_2) = \Delta(D_1) \cap \Delta(D_2)$, $\nabla(D_1 \cap D_2) \subseteq \nabla(D_1) \cap \nabla(D_2)$,
  \item[(iii)] $\Delta(D_1 \cup D_2) \supseteq \Delta(D_1) \cup \Delta(D_2)$, $\nabla(D_1 \cup D_2) = \nabla(D_1) \cup \nabla(D_2)$,
  \item[(iv)] $\Delta(D_1 \rightarrow D_2) \subseteq \Delta(D_1) \rightarrow \Delta(D_2)$, $\nabla(D_1 \rightarrow D_2) \subseteq \nabla(D_1) \rightarrow \nabla(D_2)$,
  \item[(v)] $\Delta(D_1 \mathbin{\bowtie} D_2) = \Delta(D_1) \mathbin{\bowtie} \Delta(D_2)$, $\nabla(D_1 \mathbin{\bowtie} D_2) \subseteq \nabla(D_1) \mathbin{\bowtie} \nabla(D_2)$,
  \item[(vi)] $\Delta(\pi_R(D)) \supseteq \pi_R(\Delta(D))$, $\nabla(\pi_R(D)) = \pi_R(\nabla(D))$,
  \item[(vii)] $\Delta(D_1 \div D_2) \subseteq \Delta(D_1) \div \Delta(D_2)$, $\nabla(D_1 \div D_2) \subseteq \nabla(D_1) \div \nabla(D_2)$,
  \item[(viii)] $\Delta(\sigma_\theta(D)) \subseteq \sigma_\theta(\Delta(D))$, $\Delta(\sigma_\theta(D)) = \Delta(\sigma_\theta(\Delta(D)))$.
\end{enumerate}

If $L$ is linear, then

\begin{enumerate}
  \item[(ix)] $\nabla(D_1 \cap D_2) = \nabla(D_1) \cap \nabla(D_2)$, $\Delta(D_1 \cup D_2) = \Delta(D_1) \cup \Delta(D_2)$,
  \item[(x)] $\Delta(\pi_R(D)) = \pi_R(\Delta(D))$.
\end{enumerate}

Proof: All (i)–(x) follow by elementary checking. For instance, in case of (vi), observe that $(\pi_R(\Delta(D)))(r) \in \{0, 1\}$. Thus, if $(\pi_R(\Delta(D)))(r) = 1$, there is tuple $s$ such that $(\Delta(D))(rs) = 1$, i.e. $D(rs) = 1$. As a consequence, $(\pi_R(D))(r) = 1$ and hence $(\Delta(\pi_R(D)))(r) = 1$. The rest can be shown analogously.

Interestingly, neither $\nabla(\sigma_\theta(D)) \subseteq \sigma_\theta(\nabla(D))$ nor $\nabla(\sigma_\theta(D)) \supseteq \sigma_\theta(\nabla(D))$ holds in general which can be shown by simple counterexamples considering $L$ as a three-element Łukasiewicz chain.
4.7. Renaming Attributes

As in the ordinary model, we introduce renaming of attributes which allows to change names of attributes without otherwise altering RDTs (i.e. the data or ranks). For an RDT $\mathcal{D}$ on $R$ and an injective map $h : R \to Y$ such that $\lambda(y) = \lambda(h(y))$ holds for each $y \in R$, we define a renaming $\rho_h(\mathcal{D})$ of $\mathcal{D}$ by $h$ as an RDT on $h(R) = \{ h(y) \mid y \in R \}$ by $(\rho_h(\mathcal{D}))(h(r)) = \mathcal{D}(r)$, where $h(r) \in \text{Tupl}(h(R))$ such that $(h(r))(h(y)) = r(y)$ for each attribute $y \in R$. We also adopt a common notation $\rho_{h(y_1),...,h(y_n)}(\mathcal{D})$ for $\rho_h(\mathcal{D})$ if \{ $y_1, \ldots, y_n$ \} $\subseteq R$ and $h(y) = y$ for each $y \in R$ not appearing among $y_1, \ldots, y_n$.

4.8. Relational algebra expressions and queries

We now introduce queries that correspond to consecutive application of relational operations to RDTs. In the same spirit as in the Codd model, the queries result by evaluation of relational algebra expressions. From the logical point of view, relational algebra expressions are terms and the induced queries are (partial) term functions. Relational algebra expressions are thus syntactic notions and are defined recursively as follows.

**Definition 4.24:** Let $\langle R, \rho \rangle$ be a database scheme on $Y$ and let $\Lambda = \langle \Lambda, \lambda \rangle$ be a type declaration for $Y$ and $\mathcal{C}$. The relational algebra expressions over $\langle R, \rho \rangle$ (shortly, RA-expressions) are defined as follows:

(i) If $r \in \mathbb{R}$, then $r$ is RA-expression on $\rho(r)$;
(ii) if $a \in L$, then $\overline{a}_h$ is RA-expression on $\emptyset$;
(iii) if $d \in \mathcal{C}$, $y \in Y$, and $\lambda(d) = \lambda(y)$, then $[y:d]$ is RA-expression on $\{y\}$;
(iv) if $E_1$ and $E_2$ are RA-expressions on $R$, then $(E_1 \cap E_2)$ and $(E_1 \cup E_2)$ are RA-expressions on $R$,
(v) if $E_1$, $E_2$, and $E_3$ are RA-expressions on $R$, then $(E_1 \rightarrow E_3 \ E_2)$ is RA-expression on $R$;
(vi) if $E_1$ is RA-expression on $R_1$ and $E_2$ is RA-expression on $R_2$ then $(E_1 \bowtie E_2)$ is RA-expression on $R_1 \cup R_2$;
(vii) if $E$ is RA-expression on $T$ and $R \subseteq T$, then $\pi_R(E)$ is RA-expression on $R$;
(viii) if $E_1$ is RA-expression on $R$, $E_2$ is RA-expression on $S \subseteq R$, and $E_3$ is RA-expression on $T = R \setminus S$, then $(E_1 \rightarrow E_3 \ E_2)$ is RA-expression on $T$;
(ix) if $E$ is RA-expression on $R$, $y \in R$, and $z \in R \cup \mathcal{C}$ such that $\lambda(y) = \lambda(z)$, then $\sigma_{y\rightarrow z}(E)$ is RA-expression on $R$;
(x) if $E$ is RA-expression on $R$, then $\Delta(E)$ and $\nabla(E)$ are RA-expressions on $R$;
(xi) if $E$ is RA-expression on $R$, and $h : R \to Y$ is an injective map such that for each $y \in R$ we have $\lambda(y) = \lambda(h(y))$, then $\rho_h(E)$ is RA-expression on $h(R)$.

All RA-expressions result by application of (i)–(xi). In addition, if $E$ is RA-expression on $R$, we call $R$ the relation scheme of $E$ and denote it by $\text{sch}(E)$.

**Remark 4.25:**

(a) RA-expressions can be seen as terms written in a predicate language which is given by (i) database scheme $\langle R, \rho \rangle$; (ii) set of object constants $\mathcal{C}$; (iii) type declaration $\Lambda$; and (iv) the set $L$ or degrees. Note that the first three items of Definition 4.24 define atomic RA-expressions: relation symbols, expressions corresponding to degrees from $L$ (note here that we distinguish a degree $a \in L$ and its syntactic representation $\overline{a}$, i.e. the constant for degree $a$, cf. Hájek (1998)), and singleton
the rules in Definition 4.26 yield query results.

(b) We omit parentheses in RA-expressions to simplify notation unless the omission introduces ambiguity. For instance, instead of \((\Delta(r) \bowtie \sigma_{y \approx z}(s \cup t))\) we write \(\Delta(r) \bowtie \sigma_{y \approx z}(s \cup t)\), etc.

By definition, each RA-expression \(E\) has its relation scheme \(\text{sch}(E)\) but does not have a value on its own right (it is a syntactic notion). In order to evaluate RA-expressions, we supply a database instance and define results of RA-expressions in the database instance. If defined, the result of \(E\) in a database instance is an RDT on the relation scheme \(\text{sch}(E)\).

**Definition 4.26:** Let \(D = \langle U^D, \mathcal{R}^D, \mathcal{C}^D \rangle\) be an instance of database scheme \(\langle \mathcal{R}, \varrho \rangle\) which respects a type declaration \(\Lambda = \langle \Lambda, \lambda \rangle\) for \(Y\) and \(C\). Then, for each RA-expression \(E\) over \(\langle \mathcal{R}, \varrho \rangle\), we define an RDT \(E^D\), called the result of \(E\) in \(D\) as follows:

1. If \(E\) is \(r \in \mathbb{R}\), then \(E^D = r^D\);
2. If \(E\) is \(a\), then \(E^D = a^D\);
3. If \(E\) is \([y:a]\), then \(E^D = [y:a]^D\);
4. If \(E\) is \(E_1 \cap E_2\), then \(E^D = E_1^D \cap E_2^D\);
5. If \(E\) is \(E_1 \sigma E_2\), then \(E^D = E_1^D \sigma E_2^D\);
6. If \(E\) is \(\pi_Y(F)\), then \(E^D = \pi_Y(F^D)\);
7. If \(E\) is \(\pi_D(F)\), then \(E^D = \pi_D(F^D)\);
8. If \(E\) is \(\sigma_{y \approx z}(F)\) and \(z \in Y\), then \(E^D = \sigma_{y \approx z}(F^D)\);
9. If \(E\) is \(\sigma_{y \approx a}(F)\) and \(a \in C\), then \(E^D = \sigma_{y \approx a}(F^D)\);
10. If \(E\) is \(\Delta(F)\), then \(E^D = \Delta(F^D)\);
11. If \(E\) is \(\rho_Y(F)\), then \(E^D = \rho_Y(F^D)\).

Restricting to database instances respecting type declarations allows us to have all results of queries defined. For general database instances, results of queries involving singleton expressions, restriction expressions, and renaming expressions may not be well defined due to type incompatibility. As in the Codd model, the relational queries are expressed by arbitrarily complex RA-expressions whose evaluation in database instance according to the rules in Definition 4.26 yields query results.

**Remark 4.27:** Let us note that using \([y:a], a\), union and join expressions, we are capable of expressing an arbitrary RDT whose values (in database instances) are given by values of object constants. Indeed, for a relation scheme \(R = \{y_1, \ldots, y_n\}\), an RDT with a single tuple with rank 1 can be expressed as \(\bowtie_{i=1}^n [y_i:a_i]\). In addition to that, by \(a\bowtie (\bowtie_{i=1}^n [y_i:a_i])\), we express an RDT with the same tuple and rank equal to \(a\) instead of 1. Then, by taking a union of finitely many of such expressions, we express an RDT with all values given by values of object constants. Hence, arbitrarily large “constant RDTs” can be expressed by our relational algebra expressions.

Although this paper is focused primarily on query systems, it is worth mentioning that RA-expressions can be used as general similarity-based integrity constraints. Recall that in the ordinary relational model, a general integrity constraint is usually defined as an atomic formula of the form \(E \subseteq F\), where both \(E\) and \(F\) are RA-expressions in the ordinary sense. The constraint is satisfied by an ordinary database instance if the value of \(E\) in the database
instance is a subset of the value of $F$ in the database instance. The usual and widely used constraints including keys, functional dependencies, and referential integrity constraints can be expressed this way. In our setting, it is natural to consider integrity constraints based on subsethood degrees (33) which can be satisfied to general degrees not just 0 (not satisfied) and 1 (satisfied). For instance, we may introduce the constraints as atomic inequalities of the from

$$S(E, F) \geq a,$$

(47)

with $E$ and $F$ being RA-expressions such that $\text{sch}(E) = \text{sch}(F)$ and $a \in L$. Given a database instance $D$, we may call (47) valid in $D$ if $S(E^D, F^D) \geq a$, i.e. if the degree to which $E^D$ is included in $F^D$ exceeds the threshold $a \in L$, see (33). Another approach is to consider constraints as formulas $S(E, F)$, (48), and define a degree of validity of (48) in $D$ by $S(E^D, F^D)$. Clearly, both (47) and (48) represent a new and nontrivial type of data dependency that cannot be captured in the classical model since it relies on ranks taken from complete residuated lattices, relational operations in the generalized model, and the concept of general subsethood degrees which is based on residuated implication.

### 5. Domain relational calculus

We now introduce a query system which is in many aspects close to the ordinary domain relational calculus of Lacroix and Pirotte (1977) with range declarations (DRC-RD). Our calculus differs from the ordinary DRC-RD in that its underlying structure of degrees is a complete residuated lattice instead of the two-element Boolean algebra. Technically, formulas of our calculus will be introduced as first-order formulas of predicate fuzzy logic based on complete residuated lattices. The formulas, when evaluated in database instances, are allowed to have general degrees of satisfaction, not just 1 and 0, and the general degrees of satisfaction define ranks in query results (i.e. in RDTs). Despite this principal difference, the symbolic level of both the calculi is the same: Users acquainted with the ordinary DRC-RD (and/or predicate logic) may read formulas appearing in our calculus the same way as ordinary formulas (e.g. an existentially quantified formula $(\exists x)\varphi$ means “there is $x$ such that $\varphi$” as in the ordinary predicate logic even though its real meaning takes general, intermediate truth degrees into account, in addition to 0 and 1).

This feature is a strong point of our model. It enables us to generalize notions from the classical model to our model via DRC-RD. Namely, if one expresses an ordinary notion (e.g. a derived operation with relations) via an ordinary DRC-RD formula, by considering the same formula in our calculus (and evaluating the formula in a database instance in sense of Definition 3.11), one introduces a generalization of the ordinary notion to similarity-based databases. Thus, besides having a declarative query system the calculus we present here is important from methodological point of view since it offers a sound and conceptually clear way to fully exploit and possibly extend the similarity-based model.

Before we delve into technical details, let us note that formulas we introduce in this section are considered with respect to a predicate language given by (i) a database scheme
\( (\mathbb{R}, \varrho); \) (ii) a set of object constants \( \mathcal{C}; \) and (iii) constants for truth degrees (for each \( a \in L \) we consider a corresponding constant denoted \( \overline{a} \)). The language further consists of object variables, symbols for logical connectives (binary connectives: conjunctions \( \otimes \) and \( \land \), disjunction \( \lor \), implication \( \Rightarrow \); unary connective: hedge \( \triangle \)), quantifiers (existential \( \exists \), universal \( \forall \)), and auxiliary symbols (parentheses). All the utilized notions are explained in the following subsections.

### 5.1. Variables, range declarations, and types

We assume that there is a denumerable set \( X \) of object variables (shortly, variables), which are denoted \( x, x', x_1, x_2, \ldots \) As in the predicate logic, the variables denote elements from domains in database instances. Unlike object constants from \( \mathcal{C} \) which are assigned their values by database instances (i.e. in \( \mathcal{D} \), each \( c \in \mathcal{C} \) has a fixed value \( c^\mathcal{D} \)), object variables are assigned values independently on database instances and are allowed to vary in their domains. In order to make sure that results of queries in our calculus are proper RDTs (namely, that they are finite), values of a variable can only vary within a predefined range. We ensure this by introducing range declarations:

**Definition 5.1:** A range component (over \( \langle \mathbb{R}, \varrho \rangle \) and \( \mathcal{C} \)) is (i) any object constant from \( \mathcal{C} \) and (ii) any expression \( r(y) \), where \( r \in \mathbb{R} \), and \( y \in \varrho(r) \). A range declaration (over \( \langle \mathbb{R}, \varrho \rangle \) and \( \mathcal{C} \)) is any finite set \( \mathbb{R} \) of range components. A range declaration \( \mathbb{R} \) is compatible with type declaration \( \Lambda = \langle \Lambda, \lambda \rangle \) if all attributes and constants appearing in \( \mathbb{R} \) have the same type \( \tau \in \Lambda \) which is called the type of \( \mathbb{R} \) and denoted \( \lambda(\mathbb{R}) \). The set of all range declarations over \( \langle \mathbb{R}, \varrho \rangle \) and \( \mathcal{C} \) compatible with \( \Lambda \) is denoted by \( \text{Rd}(\mathbb{R}, \varrho, \mathcal{C}, \Lambda) \). Any map \( \text{rd}: X \to \text{Rd}(\mathbb{R}, \varrho, \mathcal{C}, \Lambda) \) is called a range declaration for variables from \( X \). The value \( \text{rd}(x) \) is called a range declaration for \( x \).

As a consequence of Definition 5.1, each map \( \text{rd}: X \to \text{Rd}(\mathbb{R}, \varrho, \mathcal{C}, \Lambda) \) can be seen as introducing types of variables. Indeed, the type of \( x \) under \( \text{rd} \) can be defined as the type of \( \text{rd}(x) \). We further introduce values of range declarations as follows.

**Definition 5.2:** Let \( \mathcal{D} = \langle U^\mathcal{D}, R^\mathcal{D}, \mathcal{C}^\mathcal{D} \rangle \) be an instance of database scheme \( \langle \mathbb{R}, \varrho \rangle \) and let \( \mathbb{R} \in \text{Rd}(\mathbb{R}, \varrho, \mathcal{C}, \Lambda) \) be a range declaration. The set

\[
\mathbb{R}^\mathcal{D} = \{ c^\mathcal{D} | c \in \mathbb{R} \} \cup \{ d \in D_y | r(y) \in \mathbb{R} \quad \text{and} \quad (\pi_{\{y\}}(r^\mathcal{D}))(\{(y, d)\}) > 0 \}
\]

is called the value of \( \mathbb{R} \) in database instance \( \mathcal{D} \).

Since \( \mathbb{R}^\mathcal{D} \) are always finite, the value of a range declaration in a database instance can be expressed by a nonranked RDTs with a single attribute. In fact, for each \( R \), one can find an RA-expression whose value in the database instance is exactly such a nonranked RDT as it is shown by the following proposition.

**Proposition 5.3:** Let \( \mathbb{R} \) be a range declaration which is compatible with type declaration \( \Lambda = \langle \Lambda, \lambda \rangle \). Then, for each attribute \( y \) such that \( \lambda(y) = \lambda(\mathbb{R}) \) there is an RA-expression \( E_{\mathbb{R}, y} \) on \( \{y\} \) such that for each \( r \in \text{Tupl}(\{y\}) \), we have

\[
(E_{\mathbb{R}, y}^\mathcal{D})(r) = \begin{cases} 1, & \text{if } r(y) \in \mathbb{R}^\mathcal{D}, \\ 0, & \text{otherwise}, \end{cases}
\]

with \( \mathcal{D} \) being arbitrary database instance.
Remark 5.5: All formulas result by application of (i)–(vi) with range declarations and $C_1$ are in a one-to-one correspondence with the values from $R$.

Definition 5.4: Let $\text{R} \in \text{R}$ be a database scheme on $Y$, let $\Lambda$ be a type declaration for $Y$ and $\mathcal{C}$, and let $X$ be a set of object variables. Formulas of the domain relational calculus with range declarations (shortly, DRC-RD formulas or formulas) over $(\text{R}, \varnothing), \mathcal{C}$, and $\Lambda$ are defined as follows:

(i) If $r \in \text{R}$ is a relation symbol such that $\varnothing(r) = \{y_1, \ldots, y_n\}$ and $x_1, \ldots, x_n \in X$ are arbitrary variables, then $\varnothing(y_1;x_1, \ldots, y_n;x_n)$ is formula;
(ii) if $a \in L$, then $a$ is formula;
(iii) if $x \in X$ and $z \in X \cup \mathcal{C}$, then $x \approx z$ is formula;
(iv) if $\varnothing_i$ and $\psi$ are formulas then $(\varnothing_i \otimes \psi), (\varnothing_i \land \psi), (\varnothing_i \lor \psi), (\varnothing_i \Rightarrow \psi)$ are formulas;
(v) if $\varnothing_i$ is formula, then $\varnothing_i$ is formula;
(vi) if $\varnothing_i$ is formula, $x \in X$, and $\text{R} \in \text{Rd}(\text{R}, \varnothing_i, \mathcal{C}, \Lambda)$, then $(\forall x \in \text{R})\varnothing_i$ and $(\exists x \in \text{R})\varnothing_i$ are formulas and $\varnothing_i$ is called the scope of the quantifiers $(\forall x \in \text{R})$ and $(\exists x \in \text{R})$, respectively.

All formulas result by application of (i)–(vi).

Remark 5.5:

(a) As in the ordinary DRC, the order in which variables appear in atomic formulas $\varnothing_i(\cdots)$ is not essential since they are labeled by attributes from the relation scheme of $r$. Thus, $\varnothing(y_1;x_1, y_2;x_2)$ is equivalent to $\varnothing(y_2;x_2, y_1;x_1)$ which corresponds with the unordered nature of attributes in relation schemes and RDTs. We also keep the usual rules on omission of outermost parentheses in formulas. In quantified formulas, in addition to the variable bound by the quantifier, we specify its range declaration. The symbol “$\in$” appearing in $(\forall x \in \text{R})$ and $(\exists x \in \text{R})$ is used only to separate variables from their ranges (we do not interpret it as “set membership”). Further derivable logical connectives can be introduced as abbreviations. For instance, $\neg \varnothing_i$ (a negation) stands for $\varnothing_i \Rightarrow \varnothing_i$; $\varnothing_i \leftrightarrow \psi$ (a logical biconditional) stands for $(\varnothing_i \Rightarrow \psi) \land (\psi \Rightarrow \varnothing_i)$.

(b) As we have outlined in the introduction to Section 5, formulas can be read the usual way. For instance, $\varnothing(y_1;x_1, y_2;x_2) \otimes \varnothing(y_1;x_1, y_2;x_2)$ reads “the value of $y_1$ denoted by $x_1$ and the value of $y_2$ denoted by $x_2$ are related according to $r$ and the value of $y_1$ denoted by $x_1$ and the value of $y_2$ denoted by $x_2$ are related according to $s$”. Analogously, $(\exists x \in \text{R})\varnothing(y_1;x_1, y_2;x_2)$ can be read as “there is a value of $y_1$ denoted by $x$ (which ranges over $R$) which is according to $r$ related to the value of $y_2$ denoted by $x$”, cf. Section 4.4 and the existential meaning of projections. Constants for
truth degrees (atomic formulas of the form \( \bar{a} \)) allow us to express formulas like 
\( \bar{a} \Rightarrow r(y_1 : x, y_2 : z) \) which may be read “the value of \( y_1 \) denoted by \( x \) is according to 
\( r \) related to the value of \( y_2 \) denoted by \( z \) at least to degree \( a \)”, cf. Section 4.2 and the 
meaning of residuated \( a \)-shifts. The hedge \( \triangle \) is a unary connective which does not 
have a nontrivial counterpart in the classic logic and serves as a connective “fully 
true” (\( \triangle \varphi \) can be read “\( \varphi \) is fully true”).

We use the usual notion of a subformula. We also adopt the notions of free and 
bound occurrences of variables in formulas depending on whether they lie within a 
scope of a quantifier. Now we can explain the role of range declarations given by 
\( rd : X \rightarrow Rd(\mathbb{R}, \varrho, \xi, \Lambda) \) and the range declarations that appear in quantified formulas. Each 
ocurrence of a variable in a formula from Definition 5.4 is supposed to have its range 
whereas range declarations of bound variables are specified in quantifiers 
\( \forall x \in \mathbb{R} \) and \( \exists x \in \mathbb{R} \) and are assumed to be valid through the whole scope of the quantifiers.

In relational queries, we are going to use formulas that together with range declarations 
for (free) variables make sense from the point of view of the type system. These formulas 
shall be called type safe:

**Definition 5.6:** Formula \( \varphi \) is called safe with respect to \( rd : X \rightarrow Rd(\mathbb{R}, \varrho, \xi, \Lambda) \) if the 
following conditions hold for any variable \( x \) that occurs in \( \varphi \):

(i) If the occurrence of \( x \) in a subformula \( \psi \) of \( \varphi \) is free in \( \varphi \) and if 
   (a) \( \psi \) is \( r(y : x, \ldots) \), then \( \lambda(y) = \lambda(rd(x)) \); or if 
   (b) \( \psi \) is \( x \approx c \), then \( \lambda(c) = \lambda(rd(x)) \); or if 
   (c) \( \psi \) is \( x \approx y \) and this occurrence of \( y \) is free in \( \varphi \), then \( \lambda(rd(x)) = \lambda(rd(y)) \).

(ii) For any subformula \( (Q \forall x \in \mathbb{R}) \psi \) of \( \varphi \) with \( Q \) being \( \forall \) or \( \exists \), the formula \( \psi \) is safe with 
    respect to \( rd' : X \rightarrow Rd(\mathbb{R}, \varrho, \xi, \Lambda) \), where \( rd'(y) = rd(y) \) for each \( y \neq x \), and 
    \( rd'(x) = R \).

Safe formulas of the domain relational calculus with range declarations serve as descriptions 
of relational queries introduced in the next section.

**5.3. Semantics of formulas, queries, and query results**

We now turn our attention to interpretation of formulas in database instances which 
defines queries in our domain relational calculus. First, we need a valuation which assigns 
values to variables so that we can interpret free variables in formulas.

**Definition 5.7:** Let \( D = (U^D, R^D, \xi^D) \) be a database instance of \( (\mathbb{R}, \varrho) \) where \( U^D = \{ \langle d_y, \approx_y \rangle \mid y \in Y \} \) and let \( rd : X \rightarrow Rd(\mathbb{R}, \varrho, \xi, \Lambda) \) be a range declaration for variables. A 
map \( v : X \rightarrow \bigcup_{y \in Y} D_y \) such that \( v(x) \in rd(x)^D \) is called a \( D \)-valuation (of variables from 
\( X \) with respect to \( rd \)) and \( v(x) \) is called the value of \( x \) under \( v \). For two \( D \)-valuations \( w \) and 
\( v \), we write \( w =_x v \) whenever \( w(x') = v(x') \) for each \( x' \in X \) different from \( x \).

According to Definition 5.7, \( w =_x v \) means that \( w \) and \( v \) differ at most in the value 
which is assigned to the variable \( x \). Now, we can introduce degrees to which formulas are 
true in database instances under \( D \)-valuations considering range declaration for variables 
\( rd : X \rightarrow Rd(\mathbb{R}, \varrho, \xi, \Lambda) \).
Definition 5.8: Let $\mathcal{D} = \langle U^{\mathcal{D}}, \mathbb{R}^{\mathcal{D}}, \mathcal{C}^{\mathcal{D}} \rangle$ be an instance of database scheme $\langle \mathbb{R}, \mathcal{C} \rangle$ which respects a type declaration $\Lambda = \langle \Lambda, \lambda \rangle$ for $\mathcal{Y}$ and $\mathcal{C}$, let $\nu$ be a $\mathcal{D}$-valuation with respect to $\mathcal{Y}$, and let $\varphi$ be a formula which is safe with respect to $\mathcal{D}$. Then, we define a degree $||\varphi||^{\mathcal{D}, \nu}$ of a degree to which $\varphi$ is true in $\mathcal{D}$ under $\nu$ and a degree as follows:

(i) If $\varphi$ is $\pi(y_1, x_1, \ldots, y_n, x_n)$, then $||\varphi||^{\mathcal{D}, \nu} = \pi^{\mathcal{D}}(t)$, where $t$ is a tuple on $\pi(\nu)$ such that $t(y_i) = \nu(x_i)$ for all $i = 1, \ldots, n$;

(ii) if $\varphi$ is $\bar{a}$, then $||\varphi||^{\mathcal{D}, \nu} = a$;

(iii) if $\varphi$ is $\approx y$, then $||\varphi||^{\mathcal{D}, \nu} = \nu(y) \approx y$ where $\lambda(y) = \lambda(\mathcal{R}(y))$;

(iv) if $\varphi$ is $\psi \otimes \chi$, then $||\varphi||^{\mathcal{D}, \nu} = ||\psi||^{\mathcal{D}, \nu} \otimes ||\chi||^{\mathcal{D}, \nu}$;

(v) if $\varphi$ is $\Delta\psi$, then $||\varphi||^{\mathcal{D}, \nu} = \left\{ \begin{array}{ll} 1, & \text{if } ||\psi||^{\mathcal{D}, \nu} = 1, \\ 0, & \text{otherwise}; \end{array} \right.$

Remark 5.9:

(a) The notion $||\cdots||^{\mathcal{D}, \nu}$ of a degree to which a formula is true in $\mathcal{D}$ under $\nu$ and $\mathcal{D}$ is naturally graded. That is, $||\cdots||^{\mathcal{D}, \nu}$ is a degree from $\mathcal{L}$, not necessarily 0 (falsity) or 1 (full truth). For instance, if $||\varphi||^{\mathcal{D}, \nu} = 0.95$, we might say that $\varphi$ is “almost true” in $\mathcal{D}$ under $\nu$ and $\mathcal{D}$. From the point of view of similarity-based databases, the degree can be viewed as a rank to which a tuple of values matches a query posed by $\nu$. Further in this section, we formalize queries following this interpretation.

(b) Interpretation of atomic formulas of the form $\pi(y_1, x_1, \ldots, y_n, x_n)$ is fully analogous to the interpretation in the ordinary DRC except for $\pi^{\mathcal{D}}$ (the interpretation of $\pi$ in $\mathcal{D}$) is an RDT with general ranks. From Definition 5.8 (ii) we can see that constants for truth degrees are interpreted by the corresponding degrees. Degrees to which formulas of the form $\approx z$ are true in $\mathcal{D}$ are determined from domain similarities $\approx$ of attributes of the same type as $\lambda(\mathcal{R}(y))$ (which is the same as $\lambda(\mathcal{R}(y))$ because $\varphi$ is considered safe). Analogous situation applies to $\approx x \approx c$. In (iv), the truth degree of compound formulas is defined based on truth degrees of simpler formulas using operations of residuated lattices $\otimes, \land, \lor, \rightarrow$ interpreting the corresponding logical connectives $\otimes, \land, \lor, \rightarrow$. As in classical logic, this corresponds to the truth-functionality principle, except that truth functions from a general complete residuated lattice $\mathcal{L}$ are used in place of the usual Boolean truth functions. Formulas $\Delta \varphi$ are interpreted according to the intended meaning of $\Delta$ as a unary connective “fully true”. Since only 1 denotes the full truth, $||\Delta\varphi||^{\mathcal{D}, \nu} = 1$ if $||\varphi||^{\mathcal{D}, \nu} = 1$ and $||\Delta\varphi||^{\mathcal{D}, \nu}$ is zero otherwise. Note that $\Delta$ has been extensively studied in multiple-value logics (fuzzy logics in the narrow sense) by Baaz (1996) and Hâjek (1998) and in linearly ordered structures it is known under the name
Baaz’s Delta. In nonlinear structures, a function with this interpretation has been studied in context of intuitionistic set theory by Takeuti and Titani (1987) and is known as globalization. Finally, (vi) defines interpretation of quantified formulas by general infima and suprema of degrees $||\psi||^r_d,D,v$ ($\psi$ is the scope of a quantifier and $rd'$ is a range declaration which redefines range for the bound variable for $\psi$). As in case of connectives, the technical difference in our interpretation of quantifiers and the classic one is using $\wedge$ and $\vee$ for general $L$ instead of the two-element Boolean algebra (recall that for general $L$, quantifiers are not mutually definable as in case of the classical predicate logic).

An important methodological remark is in order. There is a clear connection of $||\cdot\cdot\cdot||^r_d,D,v$ from Definition 5.8 and the notion of the truth of formulas in first-order fuzzy structures as they appear in fuzzy logics in the narrow sense, cf. Belohlavek (2002), Gottwald 2008 and Hájek 1998. Indeed, up to a difference in formalization which is namely due to a specific nature of database systems (e.g. no explicit order of attributes, range declarations, type safety), $D$ 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 our formulas are intended to represent queries in our model, we can explore, e.g. entailment of queries using well-established and explored notions from fuzzy logics. Another important 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 classical first-order predicate logic.

**Definition 5.10:** Let $D$ be an instance of $\langle R, \rho \rangle$ which respects type declaration $\Lambda = \langle \Lambda, \lambda \rangle$ for $Y$ and $C$ and let $\varphi$ be a formula which is safe with respect to range declaration $rd : X \rightarrow Rd(\mathbb{R}, \rho, C, \Lambda)$ for variables from $X$. A **target component** (based on $rd$) is any expression $y:x$ such that $y \in Y, x \in X$, and $\lambda(y) = \lambda(rd(x))$. A **target set** (based on $rd$)

$$\mathcal{T} = \{y_1:x_1, \ldots, y_n:x_n\} \quad (50)$$

is any finite set of target components where all $y_1, \ldots, y_n$ and all $x_1, \ldots, x_n$ are pairwise distinct. For (50) and $\varphi$, we define an RDT $\mathcal{T} \varphi^{rd,D}$ on the relation scheme $R = \{y_1, \ldots, y_n\}$, called the result of query $\mathcal{T} \varphi$ in $D$ under $rd$ as follows:

$$(\mathcal{T} \varphi^{rd,D})(r) = \left\{ \begin{array}{ll}
\bigvee\{||\varphi||^r_d,D,v | \text{for each } i: \nu(x_i) = r(y_i)\}, & \text{if } r \in \prod_{y_i \in R} rd(x_i)^D, \\
0, & \text{otherwise.} \end{array} \right. \quad (51)$$

Moreover, if $\nu(x_i) = r(y_i)$ for each $i$, we say that $\nu$ is induced by $r$.

According to Definition 5.10, $((y_1:x_1, \ldots, y_n:x_n) \varphi^{rd,D})(r)$ is a rank assigned so that one goes over all $D$-valuations which assign to variables $x_i$ appearing in the target set the corresponding $y_i$-values from the tuple $r$ and the supremum of $||\varphi||^r_d,D,v$ going over all such valuations is the resulting rank. In words, (51) is a degree to which “the variables from the target set can be assigned values so that $\varphi$ is true in $D$”. Naturally, if $\varphi$ together with $\{y_1:x_1, \ldots, y_n:x_n\}$ and $rd$ prescribe the query, then $\{y_1:x_1, \ldots, y_n:x_n\} \varphi^{rd,D}$ is the result for database instance $D$. The following proposition shows conditions related to queries that can be simplified:
Proposition 5.11: The following are true for each formula $\varphi$ which is safe with respect to $rd: X \rightarrow Rd(\mathbb{R}, \varrho, \mathfrak{C}, \Lambda)$:

(i) If for each $x$ which is free in $\varphi$ we have $v(x) = w(x)$, then $||\varphi||^{rd}_{D,v} = ||\varphi||^{rd}_{D,w}$;

(ii) If for each $x$ which is free in $\varphi$ we have $rd(x) = rd'(x)$, then for any $v$ which is a $D$-valuation with respect to both $rd$ and $rd'$, we have $||\varphi||^{rd}_{D,v} = ||\varphi||^{rd'}_{D,v}$;

(iii) For each target set $T$, there is a safe formula $\psi$ so that the variables in $T$ are exactly the free variables in $\varphi$ and for each $D$, we have $T\varphi^{rd,D} = T\psi^{rd,D}$ and

$$\left(T\psi^{rd,D}\right)(r) = ||\varphi||^{rd}_{D,v}$$

for all $D$-valuations $v$ induced by $r$.

Proof: The proofs of (i) and (ii) go by structural induction on the complexity of $\varphi$ and are routine to check (the proofs are omitted). In order to prove (iii), consider $T$ and denote by $x_1, \ldots, x_m$ variables which are free in $\varphi$ but do not occur in $T$ and denote by $z_1, \ldots, z_n$ variables from $T$ which are not free in $\varphi$. Let $\psi$ be

$$(\exists x_1 \in rd(x_1)) \cdots (\exists x_m \in rd(x_m)) \varphi \otimes (z_1 \approx z_1 \otimes (z_2 \approx z_2 \otimes \cdots))$$

Clearly, free variables in $\psi$ are exactly the variables from $T$. Now, $T\varphi^{rd,D} = T\psi^{rd,D}$ is a consequence of the facts that $||z_i \approx z_i||^{rd}_{D,v} = 1$ and 1 is a neutral element with respect to $\otimes$ and applying the following general observation $m$ times:

$$||\left(\exists x \in rd(x)\right)\chi||^{rd}_{D,v} = \sqrt{||\chi||^{rd}_{D,w} | w = v \text{ and } w(x) \in rd(x)^D}\}$$

and substituting inside (51). Now, (52) follows directly from (i) and (51) because $||\varphi||^{rd}_{D,v}$ depends only on values assigned to free variables in $\varphi$. \qed

The previous proposition enables us to simplify the notation of queries. Instead of specifying the target set, the formula, and the range declaration for variables, applying Proposition 5.11 (ii) and (iii), we may specify each query by

$$\{y_1:x_1 \in R_1, \ldots, y_n:x_n \in R_n\}\varphi,$$

meaning that we consider $\{y_1:x_1, \ldots, y_n:x_n\}\varphi$ under a range declaration of variables $rd: X \rightarrow Rd(\mathbb{R}, \varrho, \mathfrak{C}, \Lambda)$ such that $rd(x_i) = R_i$ for all $i = 1, \ldots, n$ and all free variables in $\varphi$ are exactly $x_1, \ldots, x_n$. We conclude this section by observations on the relationship to the ordinary domain relational calculus with range declarations, highlighting the facts that ordinary DRC-RD queries can be seen as particular queries in our model when the ranks and similarities are degenerate.

Theorem 5.12: If $L$ is a two-element Boolean algebra then each $T\varphi^D$ can be obtained as a result of a query in the ordinary DRC-RD calculus provided that all similarity relations in $D$ are identities.

If $L$ is arbitrary complete residuated lattice, all $r^D$ in $D$ are nonranked, all similarities in $D$ are identities, and the only constants of truth degrees appearing in $\varphi$ are $\overline{0}$ or $\overline{1}$, then $T\varphi^D$ is nonranked and a corresponding ordinary relation can be obtained as a result of a query in the ordinary DRC-RD calculus.
Proof: The first claim is a consequence of the fact that if \( L = \{0, 1\} \), truth functions in \( L \) become the truth functions of ordinary conjunction, disjunction, and implication. Thus, one can take \( \psi \) which results from \( \varphi \) by omitting all \( \Delta \)'s and replacing the constants \( \bar{0} \) and \( \bar{1} \) for truth degrees by any formulas of the form \( \bar{\neg}(\vartheta \Rightarrow \vartheta) \) and \( \bar{\vartheta} \Rightarrow \vartheta \), respectively. The rest is obvious. The second claim can be justified by similar arguments. In this case, \( L \) is arbitrary, but all operations of complete residuated lattices restricted to \( \{0, 1\} \) behave as the ordinary truth functions and no RDT with ranks other than 0 and 1 can result, cf. Definition 5.8 (i)–(iii).

\[ \square \]

6. Relational completeness

The domain relational calculus we introduced in the previous sections shall be viewed as the primary query system in our model because it is directly derived from the interpretation of first-order formulas according to common rules of predicate logics based on residuated structures of degrees – each query is given by a formula, a target set, and a range declaration for free variables, and the query evaluation in a database instance yielding the query result (an RDT) is defined in the usual truth-functional fashion. This section contains two theorems which together show that the DRC-RD in our model is equivalent to RA, i.e. the relational operations we introduced have the same expressive power in terms of query results as the primary query system.

Theorem 6.1: Let \( \varphi \) be a safe formula with respect to rd : \( X \rightarrow \text{Rd}(\mathbb{R}, \varrho, \mathcal{C}, \Lambda) \) and let \( \mathcal{T} \) be a target set based on rd. Then, there is an RA-expression \( E_{\varphi} \) over \( \langle \mathbb{R}, \varrho \rangle \) such that \( \mathcal{T} \mathcal{F}_{\varphi, \psi, \rho, \varrho, \mathcal{C}, \Lambda} = E_{\varphi} \) for any instance \( \mathcal{D} \) of \( \langle \mathbb{R}, \varrho \rangle \) which respects \( \Lambda \).

Proof: By Proposition 5.11, assume that all variables that appear in \( \mathcal{T} \) are exactly the variables which appear freely in \( \varphi \). In order to prove Theorem 6.1, we first show by structural induction that for each subformula \( \psi \) of \( \varphi \) there are

(a) a target set \( \mathcal{T}_{\psi} = \{x_1:x_1, \ldots, x_n:x_n\} \) where the names of attributes \( x_i \) are given by the corresponding variables \( x_i, \{x_1, \ldots, x_n\} \) is the set of free variables in \( \psi \), and for each \( i = 1, \ldots, n \) if \( x_i \) is free in \( \varphi \), then \( \lambda(x_i) = \lambda(y) \) for \( y:x_i \in \mathcal{T} \);
(b) a range declaration \( rd_{\psi} : X \rightarrow \text{Rd}(\mathbb{R}, \varrho, \mathcal{C}, \Lambda) \) for variables;
(c) an RA-expression \( F_{\psi} \)

such that \( \mathcal{T}_{\psi} \mathcal{F}_{\varphi, \psi, \rho, \varrho, \mathcal{C}, \Lambda} = F_{\psi} \mathcal{D} \) for any database instance \( \mathcal{D} \) of \( \langle \mathbb{R}, \varrho \rangle \) which respects \( \Lambda \). That is, the result of the DRC-RD query given by \( \mathcal{T}_{\psi} \), \( \varphi \), and \( rd_{\psi} \) in \( \mathcal{D} \) is equal to the value of \( F_{\psi} \) in \( \mathcal{D} \). For the particular case of \( \varphi \) (which is trivially a subformula of itself) we then get that \( \mathcal{T}_{\psi} \mathcal{F}_{\varphi, \psi, \rho, \varrho, \mathcal{C}, \Lambda} = F_{\psi} \mathcal{D} \). Hence, the desired \( E_{\varphi} \) can be obtained as \( \rho_{h}(F_{\psi}) \), where for all \( i = 1, \ldots, n \), \( h(x_i) = y \) whenever \( y:x_i \in \mathcal{T} \). The renaming is correct, due to (a), thus \( \mathcal{T}_{\psi} \mathcal{F}_{\varphi, \psi, \rho, \varrho, \mathcal{C}, \Lambda} = \rho_{h}(F_{\psi}) \mathcal{D} = F_{\psi} \mathcal{D} \).

Let us note that each variable \( x \) which is free in \( \psi \) (a subformula of \( \varphi \)) is either free in \( \varphi \) or bound in \( \varphi \) by a quantifier \( (Qx \in \mathbb{R}) \) and lies within its scope. Thus, for \( \psi \) and its free variable \( x \), we may introduce

\[ R_x = \begin{cases} \text{rd}(x), & \text{if } x \text{ is free in } \varphi, \\ R, & \text{if } x \text{ is bound in } \varphi \text{ by } (Qx \in \mathbb{R}). \end{cases} \]  

(54)
Consequently, for each \( y \in Y \) of the same type as \( R_x \), we may denote the corresponding RA-expression \( ER_{y,y} \) from Proposition 5.3 simply by \( E_{x,y} \). In the rest of the proof, we assume that \( r_{\psi} \) required in (b) is given by ranges (54), i.e. \( r_{\psi}(x) = R_x \) for each \( x \) which is free in \( \psi \) (ranges for other variables can be arbitrary, cf. Proposition 5.11). Moreover, each attribute \( x \) from \( \mathcal{F}_\psi \) which corresponds to free variable \( x \) appearing in \( \psi \) is considered as a fresh attribute (unused so far) so that it has the same type as \( R_x \) (this can be done since \( Y \) contains infinitely many attributes for each type).

Thus, it suffices to show (c), i.e. the existence of RA-expressions \( F_\psi \) satisfying the required condition. We proceed by cases based on the form of the subformula \( \psi \).

**Case (i)** Assume that \( \psi \) is \( \tau(y_1:x_1, \ldots, y_n:x_n) \). Then, we may put

\[
F_\psi = \rho_{x_1, x_2, \ldots, x_n} \circ \tau(y_1, \ldots, y_n)(r) \gg E_{x_1, x_1} \gg \cdots \gg E_{x_n, x_n},
\]

(55)

Observe that \( F_\psi^D \) is an RDT on the relation scheme \( R = \{x_1, \ldots, x_n\} \). Take any tuple \( r \) on \( R \) and any \( D \)-valuation \( v \) such that \( r \in \prod_{y_i \in R} r_{\psi}(x_i)^D \). Then, according to (52) and Definition 5.8, we get

\[
(\mathcal{T}_\psi \psi r_{\psi}^D, D)(r) = ||\tau(y_1:x_1, \ldots, y_n:x_n)||_{D,v} = F_\psi^D(r)
\]

because \( (E_{x_1, x_1} \gg \cdots \gg E_{x_n, x_n})^D(r) = 1 \), hence \( \mathcal{T}_\psi \psi r_{\psi}^D \subseteq F_\psi^D \). The converse inclusion can be shown analogously: For any tuple \( r \) such that \( F_\psi^D(r) > 0 \), we must have \( (E_{x_1, x_1} \gg \cdots \gg E_{x_n, x_n})^D(r) = 1 \), i.e. any \( v \) such that \( v(x_i) = r(x_i) \) is a \( D \)-valuation for which \( ||\psi||_{D,v}^r = F_\psi^D(r) \), showing \( F_\psi^D \subseteq \mathcal{T}_\psi \psi r_{\psi}^D \).

**Case (ii)** Assume that \( \psi \) is \( \overline{a} \). Then, we put

\[
F_\psi = \overline{a}_\emptyset.
\]

(56)

Notice that \( \mathcal{T}_\psi = \emptyset \), i.e. the only tuple under our consideration is \( \emptyset \) (the empty tuple). Hence, we clearly have \( (\mathcal{T}_\psi \psi r_{\psi}^D)(\emptyset) = a = a_\emptyset(\emptyset) = F_\psi^D(\emptyset) \).

**Case (iii)** Here we distinguish two subcases. First, let \( \psi \) be \( \varphi_1 \approx \varphi_2 \) and put

\[
F_\psi = \sigma_{\varphi_1 \approx \varphi_2}(E_{\varphi_1, \varphi_1} \gg E_{\varphi_2, \varphi_2}).
\]

(57)

Obviously, \( ||\varphi_1 \approx \varphi_2||_{D,v}^r = a > 0 \) iff \( v(\varphi_1) \in r_{\psi}(\varphi_1)^D \), \( v(\varphi_2) \in r_{\psi}(\varphi_2)^D \), and

\[
v(\varphi_1) \approx v(\varphi_2) = a > 0 \quad \text{for an attribute } y \text{ of the type } \lambda(r_{\psi}(x_i)) = \lambda(r_{\psi}(x_2)).
\]

The latter is true iff \( (E_{\varphi_1, \varphi_1} \gg E_{\varphi_2, \varphi_2})(r) = 1 \) for tuple \( r \) on \( \{x_1, x_2\} \) such that \( r(x_1) = v(\varphi_1), r(x_2) = v(\varphi_2), \) and \( (\sigma_{\varphi_1 \approx \varphi_2}(E_{\varphi_1, \varphi_1} \gg E_{\varphi_2, \varphi_2}))(r) = a > 0 \), showing that \( ||\psi||_{D,v}^r = a > 0 \) iff \( F_\psi^D(r) = a > 0 \).

Second, if \( \psi \) is \( \varphi \approx c \) for \( c \in C \), we proceed analogously using

\[
F_\psi = \sigma_{\varphi \approx c}(E_{\varphi, \varphi}).
\]

(58)

This concludes the proof for \( \psi \) being atomic. Now, we distinguish the following cases assuming that the claim holds for all subformulas.

**Case (iv)** We distinguish two subcases. First, assume that \( \psi \) is \( (\chi \odot \eta) \), where \( \odot \) is any of \( \otimes, \wedge, \lor \) and let \( X_\chi = \{x_1, \ldots, x_n, y_1, \ldots, y_p\} \) and \( X_\eta = \{x_1, \ldots, x_n, z_1, \ldots, z_q\} \) be the
sets of variables which appear freely in $\chi$ and $\vartheta$, respectively, i.e. $x_1, \ldots, x_n$ are all free variables common to both $\chi$ and $\vartheta$ (with the possibility of $n = 0$, $p = 0$, and $q = 0$). By induction hypothesis, we assume that there are RA-expressions $F_\chi$ and $F_\vartheta$ corresponding to $\chi$ and $\vartheta$ and satisfying assumptions (a)–(c). Clearly, the set of all free variables in $\chi \oplus \vartheta$ is exactly $X_\chi \cup X_\vartheta$. Moreover, we let $rd_\vartheta(x_i) = rd_\chi(x_i)$ (which is also equal to $rd_\vartheta(x_i)$) for each $i = 1, \ldots, n$; $rd_\vartheta(y_j) = rd_\chi(y_j)$ for each $j = 1, \ldots, p$; and $rd_\vartheta(z_k) = rd_\vartheta(z_k)$ for each $k = 1, \ldots, q$. For $\oplus$ being any of $\otimes$, $\wedge$, or $\lor$, we put

$$F_\psi = (F_\chi \otimes E_{x_1 z_1} \otimes \cdots \otimes E_{x_n z_n}) \circ (F_\vartheta \otimes E_{y_1 y_1} \otimes \cdots \otimes E_{y_p y_p}),$$  \hspace{1cm} (59)$$

where $\circ$ is the corresponding operation with RDTs, i.e. $\otimes$, $\cap$, and $\cup$, see (12)–(14). Now, the claim follows by observing that $F_\psi^D(r) > 0$ iff $r$ consists of values from the ranges of the corresponding variables (taking into account the joins in (59) and the induction hypothesis), and $F_\psi^D(r) = F_\chi^D(r) \circ F_\vartheta^D(r)$ is the corresponding operation of $L$, i.e. $\otimes$, $\land$, or $\lor$ which is using the induction hypothesis equal to $||\chi \oplus \vartheta||_{D,v}^{rd_\vartheta}$ for a $D$-valuation $v$ induced by $r$.

Second, assume that $\psi$ is ($\chi \Rightarrow \vartheta$) and, under the same notation as in the first subcase of (iv), put

$$F_\psi = (F_\chi \circ E_{x_1 z_1} \circ \cdots \circ E_{x_n z_n}) \rightarrow^G (F_\vartheta \circ E_{y_1 y_1} \circ \cdots \circ E_{y_p y_p}),$$  \hspace{1cm} (60)$$

where $G = E_{x_1 z_1} \circ \cdots \circ E_{x_n z_n} \circ E_{y_1 y_1} \circ \cdots \circ E_{y_p y_p} \circ E_{z_1 z_1} \circ \cdots \circ E_{z_n z_n}$. Again, $F_\psi^D(r) > 0$ iff $r$ consists of values from the ranges of the corresponding variables (this is now ensured by $G$) and

$$F_\psi^D(r) = F_\chi^D(r) \rightarrow^{G^D(r)} F_\vartheta^D(r).$$

Since $G^D$ is nonranked, the latter yields

$$F_\psi^D(r) = F_\chi^D(r) \rightarrow F_\chi^D(r) = ||\chi \Rightarrow \vartheta||_{D,v}^{rd_\vartheta},$$

for $v$ induced by $r$ and $r$ consisting of values from ranges of the corresponding variables, proving the case of $\Rightarrow$.

**Case (v)** Assume that $\psi$ is $\Delta \chi$. Then, we put

$$F_\psi = \Delta(F_\chi)$$  \hspace{1cm} (61)$$

and the claim clearly holds (observe that both $\psi$ and $\chi$ have the same free variables), cf. (45) and Definition 5.8.

**Case (vi)** We proceed for the existential and universal quantifiers separately. First, assume that $\psi$ is ($\exists x \in R) \chi$ and $\{x, x_1, \ldots, x_n\}$ is the set of all variables free in the scope $\chi$ (this is without loss of generality – if $x$ were not free in $\chi$, one can consider $\chi \otimes x \approx x$ as the scope of the quantifier instead of $\chi$). Now, we put

$$F_\psi = \pi_{\{x_1, \ldots, x_n\}}(F_\chi \circ E_{s,x}).$$  \hspace{1cm} (62)$$
By induction hypothesis, (a)–(c) are satisfied for \( \chi \) and thus we may let \( rd_\psi(x_i) = rd_\chi(x_i) \) for each \( i = 1, \ldots, n \) (the value \( rd_\psi(x) \) can be arbitrary since \( x \) is not free in \( \psi \)). Furthermore, we can see that \( (F_\chi \sqsupseteq E_{s,x})(r) = a > 0 \) iff \( r \) consists of values from the ranges of the corresponding variables (in particular, the \( x \)-value of \( r \) comes from \( \mathbb{R}^D \), i.e. from the range defined by the quantifier) and \( F_\chi^D(r) = a > 0 \). Using the induction hypothesis and taking \( rd_\psi \) the same as \( rd_\chi \),

\[
F_\psi^D(r) = \bigvee_{s \in \text{Tupl}_{(x_1)}} (F_\chi \sqsupseteq E_{s,x})^D(rs) = \bigvee_{s \in \text{Tupl}_{(x_1)}} (F_\chi^D(rs) \otimes E_{s,x}^D(s)) = \bigvee_{s(x) \in \mathbb{R}^D} F_\chi^D(rs) = \bigvee_{d \in \mathbb{R}^D} \|\chi\|_{\mathcal{D},v_d}^{rd_\chi} = \|((\exists x \in \mathbb{R})\chi)\|_{\mathcal{D},v},
\]

where \( v \) is induced by \( r \) and each \( v_d \) results from \( v \) and \( d \in \mathbb{R}^D \) by putting \( v_d(x) = d \) and \( v_d(y) = v(y) \) for all \( y \neq x \).

Second, under the same assumptions as for the existential quantifier, assume that \( \psi \) is \((\forall s \in \mathbb{R})\chi\) and put

\[
F_\psi = F_\chi \oplus \Gamma E_{x,x}, \text{ where } G = E_{s_1,x_1} \sqsupseteq \cdots \sqsupseteq E_{s_n,x_n}.
\]

Applying Theorem 4.18(i), \( F_\psi^D \subseteq G^D \). Since \( G^D \) is nonranked, we therefore have \( G^D(r) = 1 \) iff tuple \( r \) consists of values from the ranges of the corresponding variables (and \( = 0 \) otherwise). Furthermore, \( G^D(r) = 1 \) together with the induction hypothesis yield

\[
F_\psi^D(r) = \bigwedge_{s \in \text{Tupl}_{(x_1)}} (E_{s,x}^D(s) \rightarrow G^D(r) F_\chi^D(rs)) = \bigwedge_{s \in \text{Tupl}_{(x_1)}} (E_{s,x}^D(s) \rightarrow F_\chi^D(rs)) = \bigwedge_{s(x) \in \mathbb{R}^D} (1 \rightarrow F_\chi^D(rs)) = \bigwedge_{s(x) \in \mathbb{R}^D} F_\chi^D(rs) = \bigwedge_{d \in \mathbb{R}^D} \|\chi\|_{\mathcal{D},v_d}^{rd_\chi} = \|((\forall s \in \mathbb{R})\chi)\|_{\mathcal{D},v},
\]

where, again, \( v \) is induced by \( r \) and each \( v_d \) results from \( v \) and \( d \in \mathbb{R}^D \) by putting \( v_d(x) = d \) and \( v_d(y) = v(y) \) for all \( y \neq x \). This last case concludes the proof. \( \square \)

For the next assertion, we require a stronger assumption on similarities in database instances. Namely, we assume they satisfy (Sep) from Remark 3.2.

**Theorem 6.2:** Let \( E \) be an RA-expression over \( \langle \mathbb{R}, \varrho \rangle \). Then there exist a range declaration \( rd_E : X \rightarrow \text{Rd}(\mathbb{R}, \varrho, \mathcal{C}, \mathcal{A}) \), a formula \( \varphi_E \) safe with respect to \( rd_E \), and a target set \( \mathcal{T}_E \) based on \( rd_E \) such that \( E^D = \mathcal{T}_E \varphi_E^{rd_E,D} \) holds true for any instance \( D \) of \( \langle \mathbb{R}, \varrho \rangle \) which respects \( \mathcal{A} \) provided all similarities in \( D \) satisfy (Sep).

**Proof:** The proof involves induction over the structure of RA-expressions. In the induction step for \( E \), we assume that Theorem 6.2 holds for all subexpressions of \( E \), according to Definition 4.24. In the following arguments, we write the DRC-RD queries represented by \( \varphi_E, rd_E \), and \( \mathcal{T}_E \) as in (53) and we assume that all variables which appear free in \( \varphi_E \) are exactly all the variables contained in \( \mathcal{T}_E \) and that the names of the free variables are corresponding to the names of attributes (i.e. if \( y \in \text{sch}(E) \), then the corresponding target component is \( y:y \)).

In addition, in the proof we utilize the following transformation of formulas: for any \( \varphi_E \) with free variables \( \{y_1, \ldots, y_n\} \) and \( rd_E \) (as mentioned above), we introduce formula \( \varphi_E^{y} \) as
follows:

\[(\exists y'_1 \in rd_E(y_1)) \cdots (\exists y'_n \in rd_E(y_n))(\Delta(y_1 \approx y'_1 \otimes \cdots \otimes y_n \approx y'_n) \otimes \varphi'_E), \]  

where \(\varphi'_E\) results from \(\varphi_E\) by substituting \(y'_i\) for all occurrences of the variables \(y_i\) which are free in \(\varphi_E\). Clearly, \(\varphi'_E\) has the same free variables as \(\varphi_E\). Since each similarity \(\approx\) is separating,

\[||\varphi_E||_{rd_D}^D = \begin{cases} ||\varphi_E||_{rd_D}^D, & \text{if } v(y_i) \in rd_E(y_i) \text{ for all } i = 1, \ldots, n, \\ 0, & \text{otherwise}. \end{cases} \]  

Put in words, the value of \(\varphi'_E\) in \(D\) under \(v\) with respect to \(rd\) coincides with the value of \(\varphi\) in \(D\) under \(rd_E\) provided that \(v\) is also a \(D\)-valuation with respect to \(rd_E\). We use this transformation of \(\varphi_E\) to \(\varphi'_E\) to ensure consistent range declarations of free variables in formulas constructed from simpler ones where particular free variables may have different range declarations.

According to Definition 4.24, we recognize the following base cases.

**Case (i)** If \(E \in \mathbb{R}\), then we may consider the following query:

\[\{y_1 : y_1 \in \{r(y_1)\}, \ldots, y_n : y_n \in \{r(y_n)\}\} \circ (y_1, \ldots, y_n : y_n).\]  

Clearly, its value in \(D\) is an RDT over \(\{y_1, \ldots, y_n\}\) which is the same as \(v^D\). Observe that range declarations allow the free variables in (66) to go only over the values which are present in tuples of \(v^D\) with nonzero ranks.

**Case (ii)** If \(E\) is an expression \([y : d]\), we put

\[\{y : y \in \{d\}\} \approx y,\]  

i.e. \(y\) is allowed to be evaluated only to \(d^D\), and \(y \approx y\) is always true to degree 1. As a consequence, the result is a nonranked RDT which coincides with \([y : d]^D\). Note that instead of \(y \approx y\), one can use an arbitrary formula which is true to degree 1 in any database instance (and has \(y\) as its only free variable).

Assuming that the claim holds for all subexpressions of \(E\), we distinguish the following cases:

**Case (iv)** First, suppose that \(E\) is \(E_1 \cap E_2\). Since \(\text{sch}(E_1) = \text{sch}(E_2)\), both \(\varphi_{E_1}\) and \(\varphi_{E_2}\) have the same free variables. Let us note, however, that \(\mathcal{E}_1\) and \(\mathcal{E}_2\) may define different ranges for the free variables. We can consider query

\[\{y_1 : y_1 \in rd_{E_1}(y_1), \ldots, y_n : y_n \in rd_{E_1}(y_n)\} \varphi_{E_1} \land \varphi_{E_2}^*,\]  

because the result of \(E_1 \cap E_2\) in \(D\) consists at most of tuples containing only the values present in \(E_1^D\) (because \(0 \land a = a \land 0 = 0\)). In order to ensure that \(\varphi_{E_2}\) does not yield more
than it should considering the range declarations from $rd_{E_1}$ (and not $rd_{E_2}$), we have used $\varphi^*_E$ instead of $\varphi_E$ in (69). The rest follows from (65).

Second, if $E$ is $E_1 \cup E_2$, we proceed as before and consider

$$\{y_1: y_1 \in rd_{E_1}(y_1) \cup rd_{E_2}(y_1), \ldots, y_n: y_n \in rd_{E_1}(y_n) \cup rd_{E_2}(y_n)\} \varphi^*_E \sqcup \varphi^*_{E_2}. \quad (70)$$

Notice that here we use range declarations which are unions of the range declarations $rd_{E_1}$ and $rd_{E_2}$ for all free variables. This corresponds to the fact that $(E_1 \cup E_2)^D(r) > 0$ iff $E_1^D(r) > 0$ or $E_2^D(r) > 0$. As in the previous case, in order to ensure that (70) does not yield more than it should, $\varphi^*_E$ and $\varphi^*_E$ are used instead of $\varphi_E$ and $\varphi_E$, respectively.

Case (v) If $E$ is $E_1 \ominus E_3 E_2$, we consider the following query:

$$\{y_1: y_1 \in rd_{E_3}(y_1), \ldots, y_n: y_n \in rd_{E_3}(y_n)\} \varphi_{E_3} \otimes (\varphi^*_E \Rightarrow \varphi^*_E). \quad (71)$$

Again, we use the same argument as in Case (iv).

Case (vi) Suppose that $E$ is a join expression of the form $E_1 \Join E_2$. We can denote the relation scheme of the expressions by $sch(E) = \{y_1, \ldots, y_n\}$, $sch(E_1) = \{y_1, \ldots, y_m\}$, and $sch(E_2) = \{y_{p+1}, \ldots, y_n\}$ such that $1 \leq p \leq m \leq n$. Hence, the set of all attributes common to both $E_1$ and $E_2$ is $S = \{y_{p+1}, \ldots, y_m\}$ including the possibility of $S = \emptyset$ whenever $p = m$. Now, we may consider the following query:

$$\{y_1: y_1 \in rd_{E_1}(y_1), \ldots, y_m: y_m \in rd_{E_1}(y_m), \quad y_{m+1}: y_{m+1} \in rd_{E_2}(y_{m+1}), \ldots, y_n: y_n \in rd_{E_2}(y_n)\} \varphi_E \otimes \varphi^*_{E_2}. \quad (72)$$

As in the previous cases, we have used $\varphi^*_E$ instead of $\varphi_E$ since the values assigned to $y_{p+1}, \ldots, y_m$ in $\varphi_E$ must be taken from $rd_{E_2}$. Therefore, with the obvious exception that $E_1$ and $E_2$ have in general different relation schemes, the argument is the same as in Case (iv), cf. (69).

Case (vii) If $E$ is $\pi_R(F)$ such that $sch(F) = \{y_1, \ldots, y_n\}$ and $R = \{y_1, \ldots, y_m\}$ for $m \leq n$, then the corresponding query is:

$$\{y_1: y_1 \in rd_{F}(y_1), \ldots, y_m: y_m \in rd_{F}(y_m)\}$$

$$(\exists y_{m+1} \in rd_{F}(y_{m+1}) \ldots(\exists y_n \in rd_{F}(y_n)) \varphi_F. \quad (73)$$

Case (viii) Suppose that $E$ is $E_1 \div^{E_3} E_2$ and let $sch(E_1) = \{y_1, \ldots, y_n\}$, $sch(E_2) = \{y_1, \ldots, y_m\}$, and $sch(E_3) = \{y_{m+1}, \ldots, y_n\}$ for $m \leq n$. In this case, we may consider the following query:

$$\{y_{m+1}: y_{m+1} \in rd_{E_3}(y_{m+1}), \ldots, y_n: y_n \in rd_{E_3}(y_n)\}$$

$$(\forall y_1 \in rd_{E_2}(y_1)) \ldots(\forall y_m \in rd_{E_2}(y_m))(\varphi_{E_3} \otimes (\varphi_{E_2} \Rightarrow \varphi^*_E)). \quad (74)$$

The result of the query in $D$ is an RDT on $sch(E_3)$ and it is obviously a subset of $E_3^D$. Notice that $\varphi_{E_3}$ is used directly – all its free variables are bound by the universal quantifiers in (74) which use range declarations from $rd_{E_2}$. On the contrary, $\varphi^*_E$ must again be used instead of $\varphi_{E_1}$ because $rd_{E_2}$ and $rd_{E_1}$ (as well as $rd_{E_3}$ and $rd_{E_1}$) may assign different ranges for $y_1, \ldots, y_m$ (and $y_{m+1}, \ldots, y_n$).
Case (ix) If \( E \) is \( \sigma_{y \approx y}(F) \) and \( \text{sch}(F) = \{y_1, \ldots, y_n\} \), we may consider

\[
\{y_1 : y_1 \in \text{rd}_F(y_1), \ldots, y_n : y_n \in \text{rd}_F(y_n)\} \varphi_F \otimes y_i \approx y_j.
\]

(75)

Analogously, if \( E \) is \( \sigma_{y \approx \varnothing}(F) \) and \( \varnothing \in \mathcal{C} \), we consider

\[
\{y_1 : y_1 \in \text{rd}_F(y_1), \ldots, y_n : y_n \in \text{rd}_F(y_n)\} \varphi_F \otimes y_i \approx \varnothing.
\]

(76)

Case (x) If \( E \) is \( \Delta(F) \), we simply put

\[
\{y_1 : y_1 \in \text{rd}_F(y_1), \ldots, y_n : y_n \in \text{rd}_F(y_n)\} \Delta \varphi_F.
\]

(77)

Suppose that \( E \) is \( \nabla(F) \) and let \( \psi \) denote an arbitrary formula. Then, if \( ||\psi|| = 0 \) (we omit \( \text{rd}, D \), and \( v \) here since they are not essential for the argument), then \( ||\psi \Rightarrow \varnothing|| = 1 \) and thus \( ||\Delta(\psi \Rightarrow \varnothing)|| = 1 \), which yields \( ||\Delta(\psi \Rightarrow \varnothing) \Rightarrow \varnothing|| = 0 \). In addition, if \( ||\psi|| > 0 \), then \( ||\psi \Rightarrow \varnothing|| < 1 \) and so \( ||\Delta(\psi \Rightarrow \varnothing)|| = 0 \), i.e. \( ||\Delta(\psi \Rightarrow \varnothing) \Rightarrow \varnothing|| = 1 \). Therefore, we may use the following query:

\[
\{y_1 : y_1 \in \text{rd}_F(y_1), \ldots, y_n : y_n \in \text{rd}_F(y_n)\} \Delta(\varphi_F \Rightarrow \varnothing) \Rightarrow \varnothing.
\]

(78)

Case (xi) As in the ordinary model, if \( E \) is \( \rho_h(F) \) for \( \text{sch}(F) = \{y_1, \ldots, y_n\} \), we just change the attributes and variables in the target (and their free appearances in \( \varphi_F \)) according to the map \( h \), i.e. we put

\[
\{h(y_1) : h(y_1) \in \text{rd}_E(h(y_1)), \ldots, h(y_n) : h(y_n) \in \text{rd}_E(h(y_n))\} \varphi_E,
\]

(79)

where each \( h(y_i) \) denotes the variable \( y_i \) renamed by \( h \), \( \text{rd}_E(h(y_i)) = \text{rd}_F(y_i) \) for all \( i = 1, \ldots, n \), and \( \varphi_E \) results from \( \varphi_F \) by renaming all free occurrences of variables \( y_i \) to \( h(y_i) \).

All the constructed formulas were evidently safe with respect to the corresponding range declarations for variables, finishing the proof. \( \blacksquare \)

Remark 6.3: In general, the procedure from Theorem 6.2 does not yield the shortest possible formula for a given RA-expression. In some cases, one may omit the transformation based on (64) if free variables in formulas corresponding to subexpressions all have the same range declarations. In general, the transformation cannot be omitted. For instance, if \( E \) is \( [y:c] \cap [y:\varnothing] \), then for any database instance \( D \) such that \( c^D \neq \varnothing^D \), we get \( E^D = 0_{\{y\}} \), i.e. \( E^D \) is the empty RDT on \( \{y\} \), which is the same value as the result of

\[
\{y : y \in \{c\} \} \approx y \land (\exists y' \in \{\varnothing\})(\Delta y \approx y' \otimes y' \approx y')
\]

in \( D \) because \( \Delta y \approx y' \) is always evaluated to zero which is ensured by the assumption of Theorem 6.2 that \( \approx_y \) satisfies (Sep). Clearly, (Sep) cannot be dropped because in case of \( c^D \neq \varnothing^D \) such that \( c^D \approx_y \varnothing^D = 1 \), the result of the DRC-RD query would be nonempty. Notice that the DRC-RD query is constructed exactly according to (68) and (69).

The obvious consequence of Theorems 6.1 and 6.2 is that RA and RDC-RD as we have introduced them in our model are equivalent. We highlight two important aspects here. First, as in the ordinary model, we have shown there is a reasonable logic behind the
relational querying represented by evaluation of relations expressions which is crucial for any further formal investigation of the model. The semantics structures used in the DRC-RD can be seen as standard structures which appear, e.g. in predicate $\Delta$-core fuzzy logics as defined by Cintula and Hájek (2006). Second, declarative queries represented by formulas of DRD-RD (and queries in query languages based on that formalism) can be rewritten to simple algebraic manipulations with RDTs which constitute the core for practical query execution and its optimized implementation. Note that Vaverka and Vychodil (2016) shows an alternative tuple-based calculus for our model.

7. Derived relational operations

In this section, we show examples of derived relational operations, i.e. operations with RDTs which can be expressed by RA-expressions (or equivalently by DRC-RD queries). We focus mainly on various similarity-based operations.

7.1. General similarity-based $\theta$-restrictions

The correspondence of RA and DRC-RD allows us to generalize restrictions to use more complex formulas instead of identities as in (40). Indeed, for any DRC-RD query using a safe formula $\varphi$ with free variables $x_1, \ldots, x_n$ and considering any safe formula $\theta$ with free variables among $x_1, \ldots, x_n$, the safe formula $\varphi \otimes \theta$ (together with $rd$ and $\mathcal{T}$) can be seen as prescribing a $\theta$-based restriction of $\varphi$. Therefore, for any RA-expression $E$ and a safe formula $\theta$ whose free variables are in a correspondence with the attributes from $\text{sch}(E)$, we can introduce a (similarity-based) $\theta$-restriction $\sigma_\theta(E^D)$ of RDT $E^D$ by $\theta$ as follows:

$$\sigma_\theta(E^D) = \mathcal{T}_E(\varphi_E \otimes \theta)^{rd_E, D},$$

where $\varphi_E$, $rd_E$, and $\mathcal{T}$ are corresponding to $E$ as in the proof of Theorem 6.2. Trivially, the restriction defined by (40) is covered by (80) because we may use $x_1 \approx x_2$ (or $x \approx c$) for $\theta$. An example of a derived restriction is a threshold similarity-based restriction which allows to emphasize validity of a similarity-based condition and which results by taking $\bar{a} \Rightarrow y \approx c$ for $\theta$. Therefore, it can be written as

$$(\sigma_{\bar{a} \Rightarrow y \approx c}(D))(r) = D(r) \otimes (a \rightarrow r(y) \approx_y c^D)$$

for all $r \in \text{Tupl}(R)$. If $D$ is the result of query $Q$, then the rank of $r$ in (81) has the following meaning: “$r$ matches $Q$ and its $y$-value equals (the value of) $c$ at least to degree $a \in L$”. Let us note that the result is an RDT with possibly intermediate ranks, not just 0 and 1. Further examples are shown in Section 9.

7.2. Similarity-based semijoins and similarity closures

We now introduce derived operations which are motivated by similarity-based matches. In Section 7.3, we build on these operations and introduce similarity-based joins. The key idea is that our model allows to consider joins which allow to put together tuples based on similar values of common attributes and not just equal attributes as in case of (25).
The important aspect we would like to highlight is that a variety of similarity-based joins, as we call them, are indeed derivable from the basic relational operations introduced in Section 4. We start by considering an auxiliary notion of similarity of tuples. For any tuples \( r_1, r_2 \in \text{Tupl}(R) \), we put

\[
r_1 \approx_R r_2 = \bigwedge_{y \in R} r_1(y) \approx y \wedge r_2(y)
\]

and call \( r_1 \approx_R r_2 \) the degree to which \( r_1 \) and \( r_2 \) are similar. Values (82) depend on similarities \( \approx \) on the underlying domains, i.e. each database instance induces its own similarities on tuples (82). It can be easily seen that \( \approx_R \) is a similarity on \( \text{Tupl}(R) \), cf. Section 3.2. Alternatively, one can use \( \otimes \) instead of \( \bigwedge \) in (82), cf. Remark 4.1.

Now, for any RDTs \( D_1 \) and \( D_2 \) on \( R \cup S \) and \( S \cup T \) such that \( R \cap S = R \cap T = S \cap T = \emptyset \), we may define a (natural) similarity-based semijoin \( D_1 \bowtie \approx D_2 \) of \( D_1 \) and \( D_2 \) (in this order) by putting

\[
(D_1 \bowtie \approx D_2)(rs) = D_1(rs) \otimes \bigvee_{t \in \text{Tupl}(T)} \bigvee_{s' \in \text{Tupl}(S)} (D_2(s') \otimes s' \approx_S s)
\]

for all \( r \in \text{Tupl}(R) \) and \( s \in \text{Tupl}(S) \). If \( D_1 \) and \( D_2 \) are viewed as results of queries \( Q_1 \) and \( Q_2 \), then (83) is a degree to which “\( rs \) matches \( Q_1 \) and \( s \) is similar to \( s' \) for which \( s' \) matches \( Q_2 \)”. Thus, (83) is a degree to which \( rs \) from \( D_1 \) approximately matches a tuple from \( D_2 \). By moment’s reflection, \( D_1 \bowtie D_2 \subseteq D_1 \bowtie \approx D_2 \subseteq D_1 \), i.e. the similarity-based semijoin contains all tuples corresponding to the equality-based matches and in addition, the similarity-based matches. Looking at (83), it should be obvious now that the operation is expressible by a DRC-RD formula using existential quantification and connectives \( \otimes \) and \( \bigwedge \).

The similarity-based semijoin is an important operation. For instance, we can consider its special case for \( R = T = \emptyset \) in which case (83) simplifies to

\[
(D_1 \bowtie \approx D_2)(s) = D_1(s) \otimes \bigvee_{s' \in \text{Tupl}(S)} (D_2(s') \otimes s' \approx_S s).
\]

If in addition \( D_1 \) is nonranked and \( D_2 \subseteq D_1 \), we call (84) a similarity closure of \( D_2 \) (with respect to \( D_1 \)) and denote it by \( C_{D_1}^\approx (D_2) \). Under our assumptions, \( C_{D_1}^\approx \) is a closure operator satisfying a graded monotony condition

\[
S(D_2, D_2') \leq S(C_{D_1}^\approx (D_2), C_{D_1}^\approx (D_2'))
\]

for all \( D_2, D_2' \subseteq D_1 \). The degree \( C_{D_1}^\approx (D_2)(s) \) can be interpreted as the degree to which “\( s \) is similar to a tuple matching \( Q_2 \)” and thus, \( C_{D_1}^\approx (D_2) \) extends \( D_2 \) by all tuples (from \( D_1 \)) similar to the tuples from \( D_2 \). Examples of similarity-based semijoins and similarity closures are in Section 9.4.

7.3. Further join-like operations

Our model admits a family of similarity-based operations that generalize the classic joins which are based on combining information from two (or more) RDTs into a single one based on similarity of tuples or, in a more general setting, on a degree computed by an
additional formula which involves similarity. We outline here some possibilities in this direction.

Using cross joins and \( \theta \)-restrictions, for \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) on disjoint relation schemes, we can introduce a generalization of the classic \( \theta \)-join:

\[
\mathcal{D}_1 \bowtie_\theta \mathcal{D}_2 = \sigma_\theta (\mathcal{D}_1 \bowtie \mathcal{D}_2).
\]

(85)

Using (25) and (80), for any RA-expressions \( E_1 \) and \( E_2 \) such that \( \text{sch} (E_1) \cap \text{sch} (E_2) = \emptyset \), (85) can be rewritten as

\[
(E_1^\mathcal{D} \bowtie_\theta E_2^\mathcal{D}) (rt) = E_1^\mathcal{D} (r) \otimes E_2^\mathcal{D} (t) \otimes ||\theta||^\mathcal{D}_{D, v},
\]

where \( v(y) = r(y) \) if \( y \in \text{sch} (E_1) \); \( v(y) = t(y) \) otherwise (provided that the free variables in \( \theta \) are corresponding to the attributes from \( \text{sch} (E_1) \cup \text{sch} (E_2) \), cf. Section 7.1). A particular case of (85) is a similarity-based equijoin \( \mathcal{D}_1 \bowtie_{y_1 \approx y_2} \mathcal{D}_2 \) for \( \theta \) being the atomic formula \( y_1 \approx y_2 \). Note that unlike the classic equi-join, \( \mathcal{D}_1 \bowtie_{y_1 \approx y_2} \mathcal{D}_2 \) includes both the attributes \( y_1 \) and \( y_2 \) with possibly different values.

Natural equality-based joins can be extended to approximate matches as follows. Consider RDTs \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) on \( R \cup S \) and \( S \cup T \) such that \( R \cap S = R \cap T = S \cap T = \emptyset \). Furthermore, let \( \mathcal{D}_3 \) be an RDT on \( S \). We define a (natural) similarity-based join \( \mathcal{D}_1 \bowtie_\approx_{\mathcal{D}_3} \mathcal{D}_2 \) of \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) (constrained by \( \mathcal{D}_3 \)) by putting

\[
(\mathcal{D}_1 \bowtie_\approx_{\mathcal{D}_3} \mathcal{D}_2) (rst) = \mathcal{D}_3 (s) \otimes \bigvee_{s', s''} (\mathcal{D}_1 (rs') \otimes \approx_S s \otimes \approx_S s' \otimes \mathcal{D}_2 (s''t)),
\]

(87)

where \( s', s'' \) range over \( \text{Tupl} (S) \). Described verbally, (87) is the degree to which “\( s \) matches \( Q_3 \) and for some \( s' \) similar to \( s \), \( rs' \) matches \( Q_1 \) and for some \( s'' \) similar to \( s \), \( s''t \) matches \( Q_2 \)”. Another join-like operations is a similarity-based composition of ranked data tables:

\[
(\mathcal{D}_1 \circ_\approx \mathcal{D}_2) (rt) = \bigvee_{s', s'' \in \text{Tupl} (S)} (\mathcal{D}_1 (rs') \otimes \approx_S s' \otimes \mathcal{D}_2 (s''t)).
\]

(88)

As in the previous cases and unlike the equality-based composition, (88) allows to compose RDTs based on similar values on common attributes and not just the equal values as in the classic case. If each \( \approx_y (y \in S) \) is transitive, \( \mathcal{D}_1 \circ_\approx \mathcal{D}_2 \) prescribed by (88) can be expressed by

\[
\mathcal{D}_1 \circ_\approx \mathcal{D}_2 = \pi_{R \cup S} (\mathcal{D}_1 \bowtie_{\approx_{(\pi_S (\mathcal{D}_1))}} \mathcal{D}_2).
\]

(89)

Further types of similarity-based joins, compositions, and other operations can be introduced as derived operations in our model. A detailed description will be presented in a forthcoming paper devoted to derivable similarity-based operations. The operations showed in Section 7 are illustrated by examples in Section 9.

8. Further extensions

We briefly discuss here possible extensions of the model which result by introducing new relational operations not definable by the original ones.
8.1. General comparators

As one can expect, the model can be extended by comparators other than $\approx$. Naturally, one may consider, e.g. $\precsim$ (and its inverse $\succsim$) representing “approximately less (greater) than or equal” or “graded priority relation”, crisp comparators $\leq$ (and its inverse $\geq$) with the usual meaning, general $n$-ary comparators, etc. The general comparators require us to consider domains with general comparators $\langle D_y, \approx y, \precsim y, \ldots \rangle$ and introduce new restrictions $\sigma_y \precsim z (E)$ in RA and new atomic formulas like $x \precsim z$ in the DRC-RD with the obvious semantics. The extension to handle general comparators is therefore straightforward.

8.2. Operations based on additional connectives

Another kind of extensions results from introducing new componentwise operations with RDTs based on additional logical connectives. Let us briefly mention two important groups of possible extensions. First, one might want to extend the model by linguistic hedges like “very”, “somewhat”, “more or less” which are formalized by monotone unary truth functions on $L$ which preserve 0’s and 1’s (in the ordinary relational model, such operations are trivial because only identity satisfies the conditions). A seminal contribution to hedges in residuated logics is due to Hájek (2001). Hedges allow us to put emphasis on conditions in queries (“very similar”, “more or less matching”, ...). Note that $\Delta$ and $\nabla$ from (45) and (46) are two borderline cases of such hedges. Other hedges may be definable in our model (like in case of “at least 0.5” or “above 0.5”, see Section 9) but not in general. Hedges will be exploited in the follow-up to this paper dealing with data dependencies.

Second, we may want to introduce an independent difference-like operation. In Remark 4.9 (a), we noted that $\boxplus_0$ can be seen as a particular difference but it lacks some natural properties of difference. For instance, $D_1 \boxplus_0 D_2 = 0_R$ does not imply $D_1 \subseteq D_2$ in general. Alternatively, we can introduce difference as a new independent binary operation with RDTs defined componentwise using a truth function for difference $\ominus$ (a nonimplication) on $L$. Indeed, we may postulate the following reasonable properties of $\ominus$: for all $a, b, c, b_i (i \in I)$,

\[
a \ominus 0 = a, \quad (a \ominus b) \ominus c = (a \ominus c) \ominus b, \quad \bigvee_{i \in I} a_i \ominus b = \bigvee_{i \in I} (a_i \ominus b). \quad (90)
\]

It can be shown that such $\ominus$ induces an operation with RDTs such that $D_1 \ominus D_2 = 0_R$ iff $D_1 \subseteq D_2$. Moreover, we can show that

\[
a \oplus b = \bigvee \{c \in L \mid c \ominus a \leq b\} \quad (91)
\]

defines a binary operation $\oplus$ such that $(L, \oplus, 0)$ is a commutative monoid, and $\ominus$ is adjoint to $\ominus$ in the following sense:

\[
a \ominus b \leq c \text{ iff } a \leq b \oplus c \quad (92)
\]

and vice versa, i.e. (92) and the fact that $(L, \oplus, 0)$ is a commutative monoid imply (91). Therefore, it turns out that our selection of properties for $\ominus$ ensures in addition the existence (of a uniquely given) operation $\oplus$ which can be regarded as a nonidempotent disjunction ($\oplus$ is distributive over $\bigwedge$), cf. Remark 4.1. If $L$ is defined on the real unit
interval with its natural ordering, one may take, e.g. the usual algebraic minus and plus (bounded to \([0,1]\) for \(\ominus\) and \(\oplus\)).

Extensions of the model based on introducing new connectives require to add new componentwise operations to RA and corresponding connectives to the DRC-RD.

8.3. Generalized quantifiers

Our model can be extended by considering graded generalized quantifiers. Generalized quantifiers in predicate logics have been introduced by Mostowski (1957).

In data analysis and particular in relational querying, generalized quantifiers are recently gaining interest. The readers can find a useful survey in the monograph by Badia (2009). From our perspective, the major approaches to generalized quantifiers in querying including the query language QLGQ proposed by Badia (2009) use generalized quantifiers in the classic bivalent setting. We argue that in many cases, generalized quantifiers are naturally “graded” rather than bivalent. A typical example in this category is quantifier “many”. Clearly, it does not make much sense to sharply distinct “many” from “not many”, e.g. by postulating that “many means more than 10,000 values (of some kind)” because 9,999 would also be perceived as “many” by rational observers (a small unit does not make a difference).

In order to incorporate graded generalized quantifiers in our model, we can follow the approach of Hájek (1998) and proceed as follows. For each \(rd, D, v\), and safe formula \(\varphi\) with free variable \(x\), we define a map \(Sat_{rd, D, v}(\varphi, x) : rd(x)D \rightarrow L\) by

\[
(Sat_{rd, D, v}(\varphi, x))(d) = ||\varphi||_{rd, D, w}, \text{ where } w =_x v \text{ and } w(x) = d
\]

for all \(d \in rd(x)D\). Clearly, the degree defined by (93) expresses the degree to which \(\varphi\) is true in \(D\) if \(x\) takes the value \(d\) and the interpretation of the other variables is given by \(v\).

The semantics of a (unary) generalized quantifier, written as \((Qx \in R)\), is defined for each \(D\) by a collection of maps \(Q_{\tau} : 2^{D_y} \times L^{D_y} \rightarrow L\) where \(\lambda(y) = \tau\). Then, the truth value of a formula quantified by \((Qx \in R)\) is expressed as follows:

\[
||((Qx \in R)\varphi)||_{D, v}^{rd} = Q_{\tau}^{\lambda}(R^D, Sat_{rd, D, v}(\varphi, x)),
\]

where \(rd'\) is the same as in Definition 5.8 (vi). Now, the semantics of the quantifiers \(\forall\) and \(\exists\) introduced in Definition 5.8 becomes a particular case of (94), by putting \(\forall_{D}^{\tau}(D, M) = \bigwedge_{d \in D} M(d)\) and \(\exists_{D}^{\tau}(D, M) = \bigvee_{d \in D} M(d)\), respectively. The semantics (94) allows to have various graded quantifiers including “graded many”. For instance, if \(L\) is the real unit interval, one can introduce

\[
Many_{D}^{\tau}(D, M) = \frac{1}{|D|} \sum_{d \in D} M(d)
\]

as one of the possible semantics for a quantifier \((Manyx \in R)\). Further examples can be found in the Section 9. Analogously, one can introduce binary \((n\text{-ary})\) graded generalized quantifiers. A survey of results on quantifiers in residuated logics can be found in the monograph by Hájek (1998).
9. Examples of queries

In this section, we present examples of queries. In the following examples, we use a complete residuated lattice $L$ on the real unit interval with $\otimes$ and $\rightarrow$ being the Łukasiewicz operations, see (7). The results of Łukasiewicz operations can be depicted by shades of gray in unit squares as in Figure 1 (white represents degree 0, black represents degree 1, and the gradual transition between the white and black corresponds to the intermediate degrees from the real unit interval). From Figure 1, we can read that $a \otimes b = 0$ if $a + b \leq 0.5$, $a \rightarrow b = 1$ if $a \leq b$, etc.

Furthermore, we assume a database scheme with the following relation variables:

- $\mathbb{h}$ (representing “houses”) with attributes $id$ (unique numeric house identifier), $type$ (house type), $location$ (house location), $built$ (construction year), $bdrm$ (number of bedrooms), and $sqft$ (property area in square feet);
- $\mathbb{f}$ (representing “houses for sale”) with attributes $id$ (house identifier), $agent$ (name of agent selling the property), and $price$ (property price);
- $\mathbb{b}$ (representing “prospective house buyers”) with attributes $name$ (buyer name), $type$ (desired property type), $location$ (desired location), $bdrm$ (desired number of bedrooms), $sqft$ (desired property area), and $price$ (desired price).

We assume that the domain of $id$ is a set of integers together with $\approx_{id}$ which is the identity on the set of integers (distinct IDs are similar to degree 0). It reflects the fact that querying the database based on “similarity of IDs” instead of their equality does not make much sense since the IDs are used only to ensure the referential integrity between relation variables $\mathbb{h}$ and $\mathbb{f}$. In a similar way, the names of prospective buyers (attribute $name$) and agents (attribute $agent$) can be taken as strings of literals with $\approx_{name}$ and $\approx_{agent}$ being identities – again, in querying a database of properties for sale, we do not anticipate users to be interested in “similarity of names”. The intended application of our model database is of course an important factor for defining similarities on domains. For instance, in case of a database of “criminal suspects”, querying the database based on similarity of names (e.g. based on their pronunciation or edit distance) instead of their equality may be of considerable interest.

For the remaining attributes in this example, we consider domains equipped with non-trivial similarities. Similarities on domains of (nonnegative) numeric values of attributes $built$, $bdrm$, $sqft$, and $price$ can be defined based on a distance (a metric) which is transformed to the real unit interval using an antitone scaling function. This corresponds well with our perception of similarity of numeric values because more distant values are
perceived as less similar. In particular, we define the similarities using the following linear scaling functions \( s_y : [0, \infty) \to [0, 1] \):

\[
s_y(x) = \begin{cases} 
\frac{1}{y}(y - x), & \text{if } x \leq y, \\
0, & \text{otherwise}.
\end{cases}
\] (96)

In case of the similarity \( \approx_{\text{built}} \) on the domain of construction years, we may use \( s_{150} \) together with the absolute difference of years and put

\[
d_1 \approx_{\text{built}} d_2 = s_{150}(|d_1 - d_2|)
\] (97)

for all \( d_1, d_2 \in D_{\text{built}} \), meaning that construction years with distance more than 150 years are considered not similar at all (similar to degree 0) and smaller distances between years are gradually assigned values from the unit interval. The similarity is depicted in Figure 2 (left) containing similarity degrees for years from the interval 1800–2000 (degrees as visualized by shades of gray as in Figure 1). Analogously, we may define the similarity \( \approx_{\text{bdrm}} \) on the domain of the number of bedrooms, considering, e.g. the value of 10 bedrooms being the borderline difference value beyond which we do not consider the numbers of bedrooms similar:

\[
d_1 \approx_{\text{bdrm}} d_2 = s_{10}(|d_1 - d_2|).
\] (98)

For domains like house prices, we may require that the similarities incorporate our perception of considering “larger values more similar”. For instance, we usually consider seven-digit prices like $1,010,000 and $1,020,000 more similar than $10,000 and $20,000 even if their absolute distance is the same. Obviously, for someone who is willing to pay $1,010,000 for a house the difference of $10,000 is almost entirely insignificant. On the other hand, when seeking a low-budget solution, $10,000 and $20,000 can be considered quite different. To accommodate this kind of perception, we may define for any nonzero values \( d_1 \) and \( d_2 \) the following value:

\[
s_{\text{price}}(d_1, d_2) = s_{10^4}(\left| \log_{1+10^{-4}}(d_1) - \log_{1+10^{-4}}(d_2) \right|)
\] (99)
and introduce \( \approx_{\text{price}} \) by extending \( s_{\text{price}} \) to all values from \( [0, \infty) \):

\[
d_1 \approx_{\text{price}} d_2 = \begin{cases} 
1, & \text{if } d_1 = d_2 = 0, \\
 s_{\text{price}}(d_1, d_2), & \text{if } \min(d_1, d_2) > 0, \\
0, & \text{otherwise.}
\end{cases}
\] (100)

Now, \( 1,010,000 \approx_{\text{price}} 1,020,000 = 0.99 \) and \( 10,000 \approx_{\text{price}} 20,000 = 0.306 \), cf. the shapes of \( \approx_{\text{built}}, \approx_{\text{bdrm}}, \) and \( \approx_{\text{price}} \) in Figure 2. In a similar way, we may approach the similarity \( \approx_{\text{sqft}} \) on the domain of property areas (larger areas tend to be more similar) and define, e.g.

\[
d_1 \approx_{\text{sqft}} d_2 = s_{30}\left(\sqrt{d_1} - \sqrt{d_2}\right).
\] (101)

Domains of both the attributes type and location can be seen as finite domains of nominal (qualitative) data identified by their names. For instance, the domain of type can contain values Condominium, Log Cabin, Ranch, Single Family, … which identify particular property types; the domain of location can contain locations like Forest Ave, Bay Rd, … including abstract locations more general than simple street-based location like Old Palo Alto, Stanford U, … It may be clear that defining
\(\approx_{\text{type}}\) and \(\approx_{\text{location}}\) based on a distance as in the previous cases may not be adequate or even possible. In case of \(\approx_{\text{type}}\), there does not seem to be intuitively acceptable way to define a distance on property types. In case of \(\approx_{\text{location}}\), we may use a geographical distance of locations but arguably it will not incorporate important socioeconomic aspects like “good schools” or “quiet neighborhood” which are frequently taken into account when comparing locations by prospective house buyers. Typically, similarities on such domains are specified by experts who may utilize their expert knowledge together with purpose-specific metrics on domains. Since there is a wide variety of possible choices of similarities, the formal model must ensure that all of the possible choices are acceptable – which it does because we allow any binary \(L\)-relation to be taken as similarity as long as it is at least reflexive and symmetric, see Section 3.2. For our illustrative purposes, we assume the similarities are defined by the matrices in Figure 3; as in the previous examples, the black and white stand for 1 and 0, respectively.

Houses (value of relation variable \(\mathfrak{h}\)):

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>56</td>
<td>Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
<td>3400</td>
</tr>
<tr>
<td>58</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
<tr>
<td>63</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
<td>1840</td>
</tr>
<tr>
<td>71</td>
<td>Ranch</td>
<td>Channing Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
</tr>
<tr>
<td>82</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
</tr>
<tr>
<td>85</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
<td>4580</td>
</tr>
<tr>
<td>87</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
</tr>
<tr>
<td>93</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
<td>1130</td>
</tr>
<tr>
<td>95</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
<td>1250</td>
</tr>
</tbody>
</table>

Houses for sale (value of relation variable \(\mathfrak{f}\)):

<table>
<thead>
<tr>
<th>id</th>
<th>agent</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Adams</td>
<td>$654,000</td>
</tr>
<tr>
<td>45</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>58</td>
<td>Black</td>
<td>$829,000</td>
</tr>
<tr>
<td>63</td>
<td>Adams</td>
<td>$748,000</td>
</tr>
<tr>
<td>71</td>
<td>Adams</td>
<td>$849,000</td>
</tr>
<tr>
<td>71</td>
<td>Black</td>
<td>$798,000</td>
</tr>
<tr>
<td>82</td>
<td>Adams</td>
<td>$648,000</td>
</tr>
<tr>
<td>85</td>
<td>Black</td>
<td>$998,000</td>
</tr>
<tr>
<td>87</td>
<td>Adams</td>
<td>$430,000</td>
</tr>
<tr>
<td>87</td>
<td>Black</td>
<td>$448,000</td>
</tr>
<tr>
<td>93</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>95</td>
<td>Black</td>
<td>$675,000</td>
</tr>
</tbody>
</table>

House buyers (value of relation variable \(\mathfrak{b}\)):

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>location</th>
<th>bdrm</th>
<th>sqft</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chang</td>
<td>Condominium</td>
<td>Old Palo Alto</td>
<td>2</td>
<td>1000</td>
<td>$500,000</td>
</tr>
<tr>
<td>Chang</td>
<td>Penthouse</td>
<td>Old Palo Alto</td>
<td>2</td>
<td>1000</td>
<td>$500,000</td>
</tr>
<tr>
<td>Davis</td>
<td>Single Family</td>
<td>The Willows</td>
<td>4</td>
<td>2000</td>
<td>$600,000</td>
</tr>
<tr>
<td>Evans</td>
<td>Ranch</td>
<td>Stanford U</td>
<td>4</td>
<td>3000</td>
<td>$800,000</td>
</tr>
<tr>
<td>Evans</td>
<td>Residential</td>
<td>Stanford U</td>
<td>4</td>
<td>3000</td>
<td>$800,000</td>
</tr>
<tr>
<td>Evans</td>
<td>Single Family</td>
<td>Stanford U</td>
<td>4</td>
<td>3000</td>
<td>$600,000</td>
</tr>
<tr>
<td>Frank</td>
<td>Log Cabin</td>
<td>Heritage Park</td>
<td>3</td>
<td>2000</td>
<td>$500,000</td>
</tr>
<tr>
<td>Frank</td>
<td>Ranch</td>
<td>Heritage Park</td>
<td>3</td>
<td>3000</td>
<td>$700,000</td>
</tr>
</tbody>
</table>

Figure 4. Ranked data tables appearing in the examples.
We have now specified the database scheme and the domains with similarities. Further in the section, we specify queries by RA-expressions and formulas of DRC-RD. In both cases, we evaluate the queries in a database instance $\mathcal{D}$ which interprets the relation variables $h, f, b$ by (nonranked) RDTs $h^\mathcal{D}, f^\mathcal{D}, b^\mathcal{D}$ listed in Figure 4 in the usual table-like form. Note that the ranks are not shown because they all equal to 1 for tuples (represented by rows) which are displayed in Figure 4 and are 0 otherwise. We keep this notation further in this section – nonranked RDTs will be displayed without the ranks as the usual database tables.

**Comments for Reviewers:** The examples in the following subsections do not exactly follow the structure of Section 4 which groups the relational operations based on their theoretical role in the model. In contrast, the following examples follow a tutorial-like approach, going from elementary examples to more complex ones. For brevity of the presentation, we not distinguish between object constants and their values and we tacitly identify the constants with the objects themselves (this does not introduce any ambiguity since we work with a fixed database instance), see Section 3.1. Analogously, we do not distinguish between the truth values $a, b, \ldots$ from $\mathbf{L}$ and the corresponding constants for the truth values $\bar{a}, \bar{b}, \ldots$.

### 9.1. Similarity-based restriction

We start by showing examples of similarity-based restrictions to obtain RDTs from the nonranked tables from Figure 4. The following RA-expression selects houses from $h^\mathcal{D}$ with locations similar to "Bay Rd":

$$\sigma_{\text{location} \approx \text{"Bay Rd"}}(h)$$

(102)

The query result contains one full (exact) match (the house with $id$ equal to 56) and other imperfect matches. The result can be displayed as a two-dimensional table with the left-most column containing ranks:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>56</td>
<td>Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
</tr>
<tr>
<td>0.972</td>
<td>45</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
</tr>
<tr>
<td>0.931</td>
<td>58</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
</tr>
<tr>
<td>0.676</td>
<td>71</td>
<td>Ranch</td>
<td>Channing Ave</td>
<td>1923</td>
<td>3</td>
</tr>
<tr>
<td>0.646</td>
<td>82</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
</tr>
<tr>
<td>0.646</td>
<td>87</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
</tr>
<tr>
<td>0.355</td>
<td>85</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
</tr>
<tr>
<td>0.309</td>
<td>93</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
</tr>
<tr>
<td>0.221</td>
<td>95</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
</tr>
<tr>
<td>0.140</td>
<td>63</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
</tr>
</tbody>
</table>

Since the input table in the previous example was nonranked, the ranks in the query result above are in fact the similarity degrees of the particular locations and the value "Bay Rd". Analogously, one may query the database for houses with construction years similar to 1995 as in:
\[ \sigma_{\text{built} \approx 1995}(\mathcal{h}) \]  

which gives the following result with two full matches:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>56</td>
<td>Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
</tr>
<tr>
<td>1.000</td>
<td>63</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
</tr>
<tr>
<td>0.946</td>
<td>58</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
</tr>
<tr>
<td>0.926</td>
<td>45</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
</tr>
<tr>
<td>0.893</td>
<td>95</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
</tr>
<tr>
<td>0.793</td>
<td>85</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
</tr>
<tr>
<td>0.780</td>
<td>82</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
</tr>
<tr>
<td>0.740</td>
<td>87</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
</tr>
<tr>
<td>0.600</td>
<td>93</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
</tr>
<tr>
<td>0.520</td>
<td>71</td>
<td>Ranch</td>
<td>Channing Ave</td>
<td>1923</td>
<td>3</td>
</tr>
</tbody>
</table>

The previous two queries can be chained to get houses near "Bay Rd" and built around 1995. Since \( \otimes \), which plays an important role in the definition of restriction (43), is associative and commutative, we may use either of the following equivalent queries:

\[ \sigma_{\text{built} \approx 1995}(\sigma_{\text{location} \approx "Bay Rd"}(\mathcal{h})) \]
\[ \sigma_{\text{location} \approx "Bay Rd"}(\sigma_{\text{built} \approx 1995}(\mathcal{h})) \]  

(104)

Note that the same query, can be written as a DRC-RD query using the following target set

\[ \{\text{id}:x_1 \in \{\mathcal{h}(\text{id})\}, \text{type}:x_2 \in \{\mathcal{h}(\text{type})\}, \ldots, \text{sqft}:x_6 \in \{\mathcal{h}(\text{sqft})\}\} \]  

(105)

and the following safe formula

\[ \mathcal{h}(\ldots, \text{type}:x_2, \text{location}:x_3, \ldots) \otimes (x_3 \approx "Bay Rd" \otimes x_4 \approx 1995). \]  

(106)

The latter query is in fact a similarity-based \( \theta \)-restriction and using the notation from Section 7.1, we may express it by

\[ \sigma(\text{location} \approx "Bay Rd") \otimes (\text{built} \approx 1995)(\mathcal{h}) \]  

(107)

The result of any of (104)–(107) is
Let us comment on the result if one uses the idempotent conjunction $\land$ instead of the nonidempotent $\otimes$ in (107). As we argued in Remark 4.1(c), these two choices have a different interpretation and in our case

$$\sigma(\text{location}\approx\text{"Bay Rd"})\land(\text{built}\approx1995)(h)$$

(108)

which stands for a DRC-RD query with target set (105) and safe formula

$$h(...,\text{type}:x_2,\text{location}:x_3,...)\otimes(x_3\approx\text{"Bay Rd"} \land x_4\approx1995)$$

(109)

gives a different result:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
<td>3400</td>
</tr>
<tr>
<td>0.898</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>0.877</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
<tr>
<td>0.426</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
</tr>
<tr>
<td>0.386</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
</tr>
<tr>
<td>0.361</td>
<td>Ranch</td>
<td>Channing Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
</tr>
<tr>
<td>0.148</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
<td>4580</td>
</tr>
<tr>
<td>0.120</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
<td>1840</td>
</tr>
<tr>
<td>0.114</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
<td>1250</td>
</tr>
</tbody>
</table>

Notice that compared to the previous result, the second and the third best matches are interchanged and their numerical values are different. Indeed, considering the similarity degrees which are involved in computing the ranks for these tuples:

"Bay Rd" $\approx_{\text{location}}$ "Michigan Ave" = 0.972, $1995 \approx_{\text{built}} 1984 = 0.926,
"Bay Rd" $\approx_{\text{location}}$ "Oakdale Ave" = 0.931, $1995 \approx_{\text{built}} 2003 = 0.946,

it is clear that $0.972 \otimes 0.926 = 0.898 > 0.877 = 0.931 \otimes 0.946$ whereas $0.972 \land 0.926 = 0.926 < 0.931 = 0.931 \land 0.946$.

The similarity-based $\theta$-restriction (108) involving $\land$ can be expressed by means of the fundamental relational operations as follows:

$$h \otimes (\sigma_{\text{location}\approx\text{"Bay Rd"}}(\nabla(h)) \land \sigma_{\text{built}\approx1995}(\nabla(h)))$$

(110)
i.e. using the $\land$-intersection, $\otimes$-intersection, and support. The expression (110) is in fact general and would yield the same result as (108) even if $h^D$ were a general RDT (with other ranks than 0 and 1). Since in our case $h$ is interpreted by a nonranked RDT, we may simplify (110) as follows:

$$\sigma_{\text{location}\approx\text{"Bay Rd"}}(h) \cap \sigma_{\text{built}\approx1995}(h).$$  \hspace{1cm} (111)

In the next example, we add an additional condition for the property area to be similar to 2500:

$$\sigma(\text{location}\approx\text{"Bay Rd"})@(\text{built}\approx1995)@(\text{sqft}\approx2500)(h)$$  \hspace{1cm} (112)

It is important to note that in this case, the result

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.847</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>0.723</td>
<td>Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
<td>3400</td>
</tr>
<tr>
<td>0.608</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
<tr>
<td>0.375</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
</tr>
</tbody>
</table>

has no exact match, meaning that the ordinary counterpart of the query would have an empty result. Moreover, we can see that unlike the previous cases, some tuples from $h$ do not match the query at all, i.e. have zero ranks and as such are not shown in the table.

Other important cases of similarity-based $\theta$-joins include the threshold similarity-based restriction like (81). Interestingly, the threshold degrees and implications in similarity-based $\theta$-joins can be used to put different emphasis on different similarity-based conditions in the queries. For instance, the following query

$$\sigma(0.5 \Rightarrow \text{location}\approx\text{"Bay Rd"})@(\text{built}\approx1995)@(0.5 \Rightarrow \text{sqft}\approx2500)(h)$$  \hspace{1cm} (113)

can be seen as a query in which we put full emphasis on the construction year being similar to 1995 but only partial emphasis on the other conditions. Therefore, using the thresholds we indicate that for the query result the similarity on construction years is more important (i.e. it contributes more) than the similarities of locations and property areas. The result is

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
<td>3400</td>
</tr>
<tr>
<td>0.946</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
<tr>
<td>0.926</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>0.780</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
</tr>
<tr>
<td>0.640</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
<td>1840</td>
</tr>
<tr>
<td>0.614</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
<td>1250</td>
</tr>
<tr>
<td>0.558</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
<td>4580</td>
</tr>
<tr>
<td>0.520</td>
<td>Ranch</td>
<td>Channing Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
</tr>
<tr>
<td>0.423</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
</tr>
<tr>
<td>0.362</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
<td>1130</td>
</tr>
</tbody>
</table>
In contrast to (112), we again have a full match in the result. This is due to the facts that 3400 \approx_{sqft} 2500 = 0.723 and 0.5 \rightarrow 0.723 = 1, i.e. the house with \textit{id} equal to 56 is a full match. The query

$$\Delta(\sigma(0.5 \rightarrow location \approx \text{"Bay Rd"}) \circ (0.5 \rightarrow built \approx 1995) \circ (0.5 \rightarrow sqft \approx 2500)(h))$$

shows a modification of the previous one in which we have used a threshold for all the similarity-based conditions and in addition we have used the kernel to extract only the full matches, resulting in the following nonranked RDT:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>56</td>
<td>Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
<td>3400</td>
</tr>
<tr>
<td>58</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
<tr>
<td>71</td>
<td>Ranch</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
</tr>
<tr>
<td>82</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
</tr>
</tbody>
</table>

As demonstrated, the shifts can help put less emphasis on selected conditions. If one wishes to add extra emphasis on particular conditions, one way to proceed is by utilizing a $\theta$-restriction based on an expression using the $\Delta$-connective which allows to use crisp (bivalent) conditions in queries. Another way to proceed is to multiply the subcondition in a query. Since $\otimes$ is not idempotent in general, queries $\sigma_{built \approx 1995}(\cdots)$ and $\sigma_{(built \approx 1995) \circ (built \approx 1995)}(\cdots)$ which in fact is $\sigma_{built \approx 1995}(\sigma_{built \approx 1995}(\cdots))$ yield in general different results, the latter being more strict than the first one. For instance, compared to (112), the query

$$\sigma(location \approx \text{"Bay Rd"}) \circ (built \approx 1995) \circ (built \approx 1995) \circ (sqft \approx 2500)(h)$$

yields (observe the lower values of ranks):

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.773</td>
<td>45 Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>0.723</td>
<td>56 Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
<td>3400</td>
</tr>
<tr>
<td>0.554</td>
<td>58 Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
<tr>
<td>0.155</td>
<td>82 Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
</tr>
</tbody>
</table>

In addition, if we use $built \approx 1995$ three times in the query (which is indicated by the standard power notation in the following query):

$$\sigma(location \approx \text{"Bay Rd"}) \circ (built \approx 1995)^3 \circ (sqft \approx 2500)(h),$$

The result is
and hence, by putting more emphasis on the construction year, the house with \(id\) equal to 56 has become the best match and the house with \(id\) equal to 82 has disappeared from the result because its construction year was too distant.

### 9.2. Projection

We now show projections in conjunction with the previous query (107). By performing the projection of (107) onto the attributes \(type\) and \(bdrm\), i.e.

\[
\pi\{type, bdrm\}(\sigma(location\approx"Bay\ Rd") \land (built\approx1995)(h)),
\]

we no longer have a tuple with rank 0.426 in the query result

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.723</td>
<td>56 Residential</td>
<td>Bay Rd</td>
<td>1995</td>
<td>3</td>
<td>3400</td>
</tr>
<tr>
<td>0.699</td>
<td>45 Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>0.500</td>
<td>58 Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
</tbody>
</table>

because the result of (107) contains a tuple (with \(id\) equal to 58) which has the same values of attributes \(type\) and \(bdrm\) as the tuple with \(id\) equal to 82 but with a strictly higher rank. By projecting just onto \(type\), i.e.

\[
\pi\{type\}(\sigma(location\approx"Bay\ Rd") \land (built\approx1995)(h)),
\]

we get a further reduction:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>Residential</td>
</tr>
<tr>
<td>0.898</td>
<td>Single Family</td>
</tr>
<tr>
<td>0.877</td>
<td>Single Family</td>
</tr>
<tr>
<td>0.386</td>
<td>Condominium</td>
</tr>
<tr>
<td>0.196</td>
<td>Ranch</td>
</tr>
<tr>
<td>0.148</td>
<td>Residential</td>
</tr>
<tr>
<td>0.140</td>
<td>Ranch</td>
</tr>
<tr>
<td>0.114</td>
<td>Penthouse</td>
</tr>
</tbody>
</table>

In the borderline case of projecting onto the empty relation scheme as in

\[
\pi\emptyset(\sigma(location\approx"Bay\ Rd") \land (built\approx1995)(h)),
\]
we get the result which is $1_\emptyset$, representing the highest rank in the result of (107). Note, however, that the result of a projection onto the empty relation scheme may be different from $1_\emptyset$ (and $0_\emptyset$). For instance, in case of (112) and

$$\pi_\emptyset(\sigma(location \approx "Bay Rd") @ (built \approx 1995) @ (sqft \approx 2500)(h))$$

the result is $0.847_\emptyset$.

### 9.3. Equality-based join

For nonranked RDTs, the results of applying $\Join$ are the same as in the ordinary model which follows from the fact that $\otimes$ restricted to {0, 1} behaves as the ordinary conjunction. Thus,

$$\pi_\emptyset(\sigma(location \approx "Bay Rd") @ (built \approx 1995) @ (sqft \approx 2500)(h))$$

yields

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
<th>agent</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
<td>Adams</td>
<td>$654,000</td>
</tr>
<tr>
<td>45</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>58</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
<td>Black</td>
<td>$829,000</td>
</tr>
<tr>
<td>63</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
<td>1840</td>
<td>Adams</td>
<td>$748,000</td>
</tr>
<tr>
<td>71</td>
<td>Ranch</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
<td>Adams</td>
<td>$849,000</td>
</tr>
<tr>
<td>71</td>
<td>Ranch</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
<td>Black</td>
<td>$798,000</td>
</tr>
<tr>
<td>82</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
<td>Adams</td>
<td>$648,000</td>
</tr>
<tr>
<td>85</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
<td>4580</td>
<td>Black</td>
<td>$998,000</td>
</tr>
<tr>
<td>87</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
<td>Adams</td>
<td>$430,000</td>
</tr>
<tr>
<td>87</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
<td>Black</td>
<td>$448,000</td>
</tr>
<tr>
<td>93</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
<td>1130</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>95</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
<td>1250</td>
<td>Black</td>
<td>$675,000</td>
</tr>
</tbody>
</table>

For the next examples, we consider the following relation scheme $R$ and project the following query results onto $R$:

$$R = \{id, location, built, sqft, agent, price\}$$

Using such a projection onto (122), we express a join of the houses for sale with the result of (113):

$$\pi_R(\sigma(0.5 \Rightarrow location \approx "Bay Rd") @ (built \approx 1995) @ (0.5 \Rightarrow sqft \approx 2500)(h) \Join f)$$

which can be verbally described as “houses for sale with construction year similar to 1995, location more or less similar to "Bay Rd" and area size more or less similar to 2500”. Notice how we have used the linguistic hedge “more or less” here to verbally express that we do not put full emphasis on the latter two conditions. The result of (123) is
In the previous example, we performed a join of an RDT (with ranks other than 0 and 1) with a nonranked one. In contrast, the query

$$\pi_R(\sigma(0.5 \rightarrow \text{location} \approx \text{"Bay Rd"}) \oslash (\text{built} \approx 1995) \oslash (0.5 \rightarrow \text{sqft} \approx 2500) (\text{f}))$$

(124)

can be seen as a join of the RDTs, both with general ranks and yields:

```
<table>
<thead>
<tr>
<th>id</th>
<th>location</th>
<th>built</th>
<th>sqft</th>
<th>agent</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.845</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>1760</td>
<td>Black</td>
<td>$829,000</td>
</tr>
<tr>
<td>0.789</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>2350</td>
<td>Adams</td>
<td>$654,000</td>
</tr>
<tr>
<td>0.699</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>2350</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>0.637</td>
<td>Forest Ave</td>
<td>1995</td>
<td>1840</td>
<td>Adams</td>
<td>$748,000</td>
</tr>
<tr>
<td>0.633</td>
<td>45 Michigan Ave</td>
<td>1984</td>
<td>2350</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>0.508</td>
<td>Amherst St</td>
<td>1979</td>
<td>1250</td>
<td>Black</td>
<td>$675,000</td>
</tr>
<tr>
<td>0.457</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3280</td>
<td>Black</td>
<td>$798,000</td>
</tr>
<tr>
<td>0.396</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3280</td>
<td>Black</td>
<td>$798,000</td>
</tr>
<tr>
<td>0.272</td>
<td>Ash St</td>
<td>1964</td>
<td>4580</td>
<td>Black</td>
<td>$998,000</td>
</tr>
<tr>
<td>0.135</td>
<td>Cornell St</td>
<td>1935</td>
<td>1130</td>
<td>Black</td>
<td>$598,000</td>
</tr>
</tbody>
</table>
```

In this particular case, the query can be equivalently stated by projecting the following similarity-based $\theta$-restriction of nonranked RDTs onto $R$:

$$\sigma(0.5 \rightarrow \text{location} \approx \text{"Bay Rd"}) \oslash (\text{built} \approx 1995) \oslash (0.5 \rightarrow \text{sqft} \approx 2500) \oslash (\text{price} \approx $750,000) (\text{f})$$

(125)

For both (124) and (125), the verbal description of the query result may be that the ranks in the result represent degrees to which "the house is located more or less near "Bay Rd", has the area more or less about 2500 square feet, is built ca. 1995, and is sold for approximately $750,000". So the full emphasis in the restriction conditions contained in the query is put on the price and the construction year. If one wishes to emphasize just the price, one may modify the query, e.g. by adding the threshold 0.5 to the condition involving the construction year, i.e.

$$\sigma(0.5 \rightarrow \text{location} \approx \text{"Bay Rd"}) \oslash (0.5 \rightarrow \text{built} \approx 1995) \oslash (0.5 \rightarrow \text{sqft} \approx 2500) \oslash (\text{price} \approx $750,000) (\text{f})$$

(126)

which gives the following result:
Our counterpart of the classic semijoin, the so-called equality-based semijoin, see (30), of the results of similarity-based restrictions of \( h \) and \( f \) which appear in the previous example can be expressed as an equality-based join of RDTs projected onto the set

\[
R_1 = \{id, type, location, built, bdrm, sqft\}
\]

of attributes from the relation scheme of \( h \). Namely,

\[
\begin{align*}
\pi_{R_1} \left( \sigma(0.5 \rightarrow \text{location} \approx \text{"Bay Rd"}) @ (0.5 \rightarrow \text{built} \approx 1995) @ (0.5 \rightarrow \text{sqft} \approx 2500) (h) \right) & \bowtie \\
\sigma_{\text{price} \approx $750,000} (f).
\end{align*}
\]

Considering the fact that we assume that \( \otimes \) is distributive over general \( \vee \) (equivalent to adjointness of \( \otimes \) and \( \rightarrow \), see Section 2 and (3) for details), the same can be expressed by

\[
\begin{align*}
\sigma(0.5 \rightarrow \text{location} \approx \text{"Bay Rd"}) @ (0.5 \rightarrow \text{built} \approx 1995) @ (0.5 \rightarrow \text{sqft} \approx 2500) (h) & \bowtie \\
\pi \{id\} \left( \sigma_{\text{price} \approx $750,000} (f) \right)
\end{align*}
\]

and it gives the following result:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.937</td>
<td>Ranch</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
</tr>
<tr>
<td>0.899</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
</tr>
<tr>
<td>0.863</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>0.853</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
</tr>
<tr>
<td>0.773</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
</tr>
<tr>
<td>0.653</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
<td>1840</td>
</tr>
<tr>
<td>0.615</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
<td>1250</td>
</tr>
<tr>
<td>0.535</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
<td>1130</td>
</tr>
<tr>
<td>0.479</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
<td>4580</td>
</tr>
<tr>
<td>0.167</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
</tr>
</tbody>
</table>

9.4. Similarity-based join and derived operations

The equality-based join

\[
h \bowtie f \bowtie b
\]

(130)
is empty since there are no tuples which are joinable in the ordinary sense. On the other hand, it is natural to consider similarity-based joins of the tables in which we utilize conditions involving similarity instead of equality (e.g. houses sold for prices similar to prices specified by buyers). For instance, we may use the following similarity-based $\theta$-join:

$$\pi_{\{id, agent, name, price\}}(\sigma(0.5 \rightarrow location \approx blocation)(0.5 \rightarrow sqft \approx bsqft)(price \approx bprice)(h \bowtie f \bowtie \rho btype, blocation, bsqft, bbdrm, bprice \leftarrow type, location, sqft, bdrm, price(b))).$$  \hspace{1cm} (131)$$

The join in (131) which is projected onto $\{id, agent, name, price\}$ may be regarded as a query which matches prospective buyers and houses for sale according to similar values of attributes $type$, $location$, $sqft$, $bdrm$, and $price$, i.e. the attributes which are common to both $h \bowtie f$ (houses for sale) and $b$ (prospective buyers). The result of the query is the following:

<table>
<thead>
<tr>
<th>id</th>
<th>agent</th>
<th>name</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.677</td>
<td>71</td>
<td>Black</td>
<td>Frank</td>
</tr>
<tr>
<td>0.658</td>
<td>82</td>
<td>Adams</td>
<td>Davis</td>
</tr>
<tr>
<td>0.627</td>
<td>45</td>
<td>Black</td>
<td>Davis</td>
</tr>
<tr>
<td>0.616</td>
<td>71</td>
<td>Adams</td>
<td>Frank</td>
</tr>
<tr>
<td>0.555</td>
<td>71</td>
<td>Black</td>
<td>Evans</td>
</tr>
<tr>
<td>0.548</td>
<td>87</td>
<td>Black</td>
<td>Chang</td>
</tr>
<tr>
<td>0.544</td>
<td>45</td>
<td>Adams</td>
<td>Davis</td>
</tr>
<tr>
<td>0.507</td>
<td>87</td>
<td>Adams</td>
<td>Chang</td>
</tr>
<tr>
<td>0.498</td>
<td>71</td>
<td>Adams</td>
<td>Evans</td>
</tr>
<tr>
<td>0.436</td>
<td>58</td>
<td>Black</td>
<td>Davis</td>
</tr>
<tr>
<td>0.386</td>
<td>82</td>
<td>Adams</td>
<td>Evans</td>
</tr>
<tr>
<td>0.357</td>
<td>63</td>
<td>Adams</td>
<td>Evans</td>
</tr>
<tr>
<td>0.310</td>
<td>85</td>
<td>Black</td>
<td>Evans</td>
</tr>
<tr>
<td>0.285</td>
<td>95</td>
<td>Black</td>
<td>Chang</td>
</tr>
<tr>
<td>0.217</td>
<td>82</td>
<td>Adams</td>
<td>Frank</td>
</tr>
<tr>
<td>0.189</td>
<td>93</td>
<td>Black</td>
<td>Frank</td>
</tr>
<tr>
<td>0.107</td>
<td>63</td>
<td>Adams</td>
<td>Frank</td>
</tr>
<tr>
<td>0.078</td>
<td>45</td>
<td>Black</td>
<td>Evans</td>
</tr>
</tbody>
</table>

The best match gives information that the house no. 71 which is sold by Black for $798,000 matches the requirements of customer Frank to degree 0.677. Analogously for the remaining tuples. In the next example, we show a modification of the query in which we require only full matches on the property type and the number of bedrooms and we consider the similarity of locations and property areas not as important as the price. The corresponding query can be written as

$$\pi_{\{id, agent, name, price\}}(\sigma(0.5 \rightarrow location \approx blocation)(0.5 \rightarrow sqft \approx bsqft)(price \approx bprice)(h \bowtie f \bowtie \rho blocation, bsqft, bprice \leftarrow location, sqft, price(b))).$$  \hspace{1cm} (132)$$

and it gives the following result:
As in the case of the equality-based natural join, both the (equality-based) semijoins \((h \bowtie f) \bowtie b\) and \(b \bowtie (h \bowtie f)\) yield empty results because there are no joinable tuples in the ordinary sense, see (130). On the other hand, the similarity-based semijoins (83) are nontrivial. Indeed, for

\[
(h \bowtie f) \bowtie b
\]

we get the following result

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
<th>agent</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.868</td>
<td>Ranch</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
<td>Black</td>
<td>$798,000</td>
</tr>
<tr>
<td>0.861</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
<td>Adams</td>
<td>$648,000</td>
</tr>
<tr>
<td>0.807</td>
<td>Ranch</td>
<td>Chaning Ave</td>
<td>1923</td>
<td>3</td>
<td>3280</td>
<td>Adams</td>
<td>$849,000</td>
</tr>
<tr>
<td>0.795</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
<td>Adams</td>
<td>$430,000</td>
</tr>
<tr>
<td>0.795</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
<td>Black</td>
<td>$448,000</td>
</tr>
<tr>
<td>0.757</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
<td>Adams</td>
<td>$654,000</td>
</tr>
<tr>
<td>0.757</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>0.699</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
<td>1250</td>
<td>Black</td>
<td>$675,000</td>
</tr>
<tr>
<td>0.676</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
<td>Black</td>
<td>$829,000</td>
</tr>
<tr>
<td>0.629</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
<td>1130</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>0.604</td>
<td>Ranch</td>
<td>Cedro Way</td>
<td>1995</td>
<td>4</td>
<td>1840</td>
<td>Adams</td>
<td>$748,000</td>
</tr>
<tr>
<td>0.569</td>
<td>Residential</td>
<td>Ash St</td>
<td>1964</td>
<td>5</td>
<td>4580</td>
<td>Black</td>
<td>$998,000</td>
</tr>
</tbody>
</table>

The ranks in the previous result can be interpreted as degrees to which the particular houses can be matched to prospective buyers. Analogously, the information to which prospective buyers can be matched with houses for sale is given by

\[
b \bowtie (h \bowtie f)
\]

which yields

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>location</th>
<th>bdrm</th>
<th>sqft</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.868</td>
<td>Frank</td>
<td>Ranch</td>
<td>Heritage Park</td>
<td>3</td>
<td>3000</td>
</tr>
<tr>
<td>0.861</td>
<td>Davis</td>
<td>Single Family</td>
<td>The Willows</td>
<td>4</td>
<td>2000</td>
</tr>
<tr>
<td>0.795</td>
<td>Chang</td>
<td>Condominium</td>
<td>Old Palo Alto</td>
<td>2</td>
<td>1000</td>
</tr>
<tr>
<td>0.699</td>
<td>Chang</td>
<td>Penthouse</td>
<td>Old Palo Alto</td>
<td>2</td>
<td>1000</td>
</tr>
<tr>
<td>0.680</td>
<td>Frank</td>
<td>Log Cabin</td>
<td>Heritage Park</td>
<td>3</td>
<td>2000</td>
</tr>
<tr>
<td>0.673</td>
<td>Evans</td>
<td>Single Family</td>
<td>Stanford U</td>
<td>4</td>
<td>3000</td>
</tr>
<tr>
<td>0.642</td>
<td>Evans</td>
<td>Ranch</td>
<td>Stanford U</td>
<td>4</td>
<td>3000</td>
</tr>
<tr>
<td>0.642</td>
<td>Evans</td>
<td>Residential</td>
<td>Stanford U</td>
<td>4</td>
<td>3000</td>
</tr>
</tbody>
</table>

If one wants to see the result of a similarity-based semijoin for just one particular tuple from \(b\), say for Chang and house type Condominium, the query can be modified so that
one uses an appropriate equality-based restriction of $b$ such as

$$\left( h \bowtie f \right) \bowtie \sigma_{\text{name}=\text{Chang} \land \text{type}=\text{Condominium}}(b) \tag{135}$$

which in fact can be seen as an abbreviation for

$$\left( h \bowtie f \right) \bowtie \left( b \bowtie \left[ \text{name}\bowtie \text{Chang} \bowtie \left[ \text{type}\bowtie \text{Condominium} \right] \right] \right) \tag{136}$$

and both (135) and (136) yield the following result:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
<th>agent</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.795</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
<td>Adams</td>
<td>$430,000</td>
</tr>
<tr>
<td>0.795</td>
<td>Condominium</td>
<td>Forest Ave</td>
<td>1956</td>
<td>2</td>
<td>650</td>
<td>Black</td>
<td>$448,000</td>
</tr>
<tr>
<td>0.680</td>
<td>Penthouse</td>
<td>Amherst St</td>
<td>1979</td>
<td>1</td>
<td>1250</td>
<td>Black</td>
<td>$675,000</td>
</tr>
<tr>
<td>0.240</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
<td>Adams</td>
<td>$654,000</td>
</tr>
<tr>
<td>0.240</td>
<td>Single Family</td>
<td>Michigan Ave</td>
<td>1984</td>
<td>3</td>
<td>2350</td>
<td>Black</td>
<td>$598,000</td>
</tr>
<tr>
<td>0.240</td>
<td>Single Family</td>
<td>Oakdale Ave</td>
<td>2003</td>
<td>4</td>
<td>1760</td>
<td>Black</td>
<td>$829,000</td>
</tr>
<tr>
<td>0.240</td>
<td>Single Family</td>
<td>Forest Ave</td>
<td>1962</td>
<td>4</td>
<td>2350</td>
<td>Adams</td>
<td>$648,000</td>
</tr>
<tr>
<td>0.240</td>
<td>Log Cabin</td>
<td>Cornell St</td>
<td>1935</td>
<td>2</td>
<td>1130</td>
<td>Black</td>
<td>$598,000</td>
</tr>
</tbody>
</table>

Another derived operation which is described in Section 7.3 is the similarity-based composition (88). By the following query, we express which houses can possibly be offered to prospective buyers:

$$\left( h \bowtie f \right) \circ \bowtie b \tag{137}$$

Since all common attributes are omitted in a composition, we obtain an RDT containing 47 tuples with nonzero ranks (the top ten and the bottom two are shown in the following table):

<table>
<thead>
<tr>
<th>name</th>
<th>agent</th>
<th>built</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.868</td>
<td>Frank</td>
<td>1923</td>
<td>71</td>
</tr>
<tr>
<td>0.861</td>
<td>Davis</td>
<td>1962</td>
<td>82</td>
</tr>
<tr>
<td>0.807</td>
<td>Frank</td>
<td>1923</td>
<td>71</td>
</tr>
<tr>
<td>0.795</td>
<td>Chang</td>
<td>1956</td>
<td>87</td>
</tr>
<tr>
<td>0.795</td>
<td>Chang</td>
<td>1956</td>
<td>87</td>
</tr>
<tr>
<td>0.757</td>
<td>Davis</td>
<td>1984</td>
<td>45</td>
</tr>
<tr>
<td>0.757</td>
<td>Davis</td>
<td>1984</td>
<td>45</td>
</tr>
<tr>
<td>0.699</td>
<td>Chang</td>
<td>1979</td>
<td>95</td>
</tr>
<tr>
<td>0.680</td>
<td>Frank</td>
<td>1962</td>
<td>82</td>
</tr>
<tr>
<td>0.676</td>
<td>Davis</td>
<td>2003</td>
<td>58</td>
</tr>
<tr>
<td>0.024</td>
<td>Evans</td>
<td>1956</td>
<td>87</td>
</tr>
<tr>
<td>0.024</td>
<td>Evans</td>
<td>1956</td>
<td>87</td>
</tr>
</tbody>
</table>

Analogously as in the case of natural join, the classic (equality-based) counterpart $\pi_{\text{name}, \text{agent}, \text{built}, \text{id}}\left( h \bowtie f \bowtie \bowtie b \right)$ of (137) would be empty.

Similarity-based closures can be seen as particular cases of similarity-based semijoins, see (84). For illustration, we can consider the following query which represents a similarity-based closure of house buyers projected onto $\text{type}$, $\text{bdrm}$, and $\text{location}$ with respect to the nonranked RDT which consists of all combinations of values of $\text{type}$, $\text{bdrm}$, and
location which can be found among the houses for sale:

\[
\left( \pi_{\{type\}}(h \bowtie f) \bowtie \pi_{\{bdrm\}}(h \bowtie f) \bowtie \pi_{\{location\}}(h \bowtie f) \right) \bowtie \\
\pi_{\{type,bdrm,location\}}(b)
\]

The query result is

<table>
<thead>
<tr>
<th>type</th>
<th>bdrm</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranch</td>
<td>4</td>
<td>Cornell St</td>
</tr>
<tr>
<td>Residential</td>
<td>4</td>
<td>Cornell St</td>
</tr>
<tr>
<td>Single Family</td>
<td>4</td>
<td>Cornell St</td>
</tr>
<tr>
<td>Ranch</td>
<td>4</td>
<td>Ash St</td>
</tr>
<tr>
<td>Residential</td>
<td>4</td>
<td>Ash St</td>
</tr>
<tr>
<td>Single Family</td>
<td>4</td>
<td>Ash St</td>
</tr>
<tr>
<td>Log Cabin</td>
<td>3</td>
<td>Forest Ave</td>
</tr>
<tr>
<td>Ranch</td>
<td>3</td>
<td>Forest Ave</td>
</tr>
<tr>
<td>Ranch</td>
<td>4</td>
<td>Amherst St</td>
</tr>
<tr>
<td>Residential</td>
<td>4</td>
<td>Amherst St</td>
</tr>
<tr>
<td>Penthouse</td>
<td>4</td>
<td>Michigan Ave</td>
</tr>
<tr>
<td>Penthouse</td>
<td>5</td>
<td>Michigan Ave</td>
</tr>
</tbody>
</table>

The following query shows a similarity-based natural join of prospective buyers and houses for sale which is constrained by the nonranked table of prospective buyers:

\[
(h \bowtie f) \bowtie \pi_{\{type,location,bdrm,sqft,price\}}(b) \\
\]

The query result is:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>location</th>
<th>built</th>
<th>bdrm</th>
<th>sqft</th>
<th>agent</th>
<th>price</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.868</td>
<td>71 Ranch</td>
<td>Heritage Park</td>
<td>1923</td>
<td>3</td>
<td>3000</td>
<td>Black</td>
<td>$700,000</td>
<td>Frank</td>
</tr>
<tr>
<td>0.861</td>
<td>82 Single Family</td>
<td>The Willows</td>
<td>1962</td>
<td>4</td>
<td>2000</td>
<td>Adams</td>
<td>$600,000</td>
<td>Davis</td>
</tr>
<tr>
<td>0.807</td>
<td>71 Ranch</td>
<td>Heritage Park</td>
<td>1923</td>
<td>3</td>
<td>3000</td>
<td>Adams</td>
<td>$700,000</td>
<td>Frank</td>
</tr>
<tr>
<td>0.795</td>
<td>87 Condominium</td>
<td>Old Palo Alto</td>
<td>1956</td>
<td>2</td>
<td>1000</td>
<td>Adams</td>
<td>$500,000</td>
<td>Chang</td>
</tr>
<tr>
<td>0.795</td>
<td>87 Condominium</td>
<td>Old Palo Alto</td>
<td>1956</td>
<td>2</td>
<td>1000</td>
<td>Black</td>
<td>$500,000</td>
<td>Chang</td>
</tr>
<tr>
<td>0.757</td>
<td>45 Single Family</td>
<td>The Willows</td>
<td>1984</td>
<td>4</td>
<td>2000</td>
<td>Adams</td>
<td>$600,000</td>
<td>Davis</td>
</tr>
<tr>
<td>0.757</td>
<td>45 Single Family</td>
<td>The Willows</td>
<td>1984</td>
<td>4</td>
<td>2000</td>
<td>Black</td>
<td>$600,000</td>
<td>Davis</td>
</tr>
<tr>
<td>0.699</td>
<td>95 Penthouse</td>
<td>Old Palo Alto</td>
<td>1979</td>
<td>2</td>
<td>1000</td>
<td>Black</td>
<td>$500,000</td>
<td>Chang</td>
</tr>
<tr>
<td>0.680</td>
<td>82 Log Cabin</td>
<td>Heritage Park</td>
<td>1962</td>
<td>3</td>
<td>2000</td>
<td>Adams</td>
<td>$500,000</td>
<td>Frank</td>
</tr>
<tr>
<td>0.680</td>
<td>87 Penthouse</td>
<td>Old Palo Alto</td>
<td>1956</td>
<td>2</td>
<td>1000</td>
<td>Adams</td>
<td>$500,000</td>
<td>Chang</td>
</tr>
<tr>
<td>0.099</td>
<td>85 Ranch</td>
<td>Heritage Park</td>
<td>1964</td>
<td>3</td>
<td>3000</td>
<td>Black</td>
<td>$700,000</td>
<td>Davis</td>
</tr>
<tr>
<td>0.009</td>
<td>85 Ranch</td>
<td>Stanford U</td>
<td>1964</td>
<td>4</td>
<td>3000</td>
<td>Black</td>
<td>$800,000</td>
<td>Davis</td>
</tr>
</tbody>
</table>

The following example is a modification of the query in (139) where we impose a restriction on the houses for sale by property type similar to Condominium and price about $450,000:

\[
\sigma_{(price \approx $450,000)}(type \approx \text{Condominium})(h \bowtie f) \bowtie \pi_{\{type,location,bdrm,sqft,price\}}(b) \\
\]

In this case, the result has 15 tuples with nonzero ranks:
We conclude this section with comments on the computation of similarities of two tuples. The similarity-based operations introduced in Section 7.2 and Section 7.3 rely on computing the degrees (82) to which two tuples are similar. We show here that such degrees can be expressed by RA-expressions. Therefore, we show there are RA-expressions which evaluate to RDTs whose ranks can be interpreted as such similarity degrees. For illustration, let us assume that we want to compare tuples from $h \Join f$ and $b$ based on the prices and area sizes. First, we consider the query

$$\rho_{\text{price1, sqft1} \leftarrow \text{price, sqft}}(\pi_{\{\text{price, sqft}\}}(h \Join f)) \Join \\
\rho_{\text{price2, sqft2} \leftarrow \text{price, sqft}}(\pi_{\{\text{price, sqft}\}}(b)) \tag{141}$$

which yields a nonranked RDT with attributes price1, sqft1, price2, and sqft2 whose values are taken from $h \Join f$ and $b$, respectively. Thus, the result of (141) is in fact a cross join of the (properly renamed) projections of $h \Join f$ and $b$ in the ordinary sense. Now, denoting the query (141) by relation variable $s$, we may use the following query:

$$\sigma_{\text{price1} \approx \text{price2}}(s) \cap \sigma_{\text{sqft1} \approx \text{sqft2}}(s) \tag{142}$$

which yields the desired result:

<table>
<thead>
<tr>
<th>price1</th>
<th>sqft1</th>
<th>price2</th>
<th>sqft2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$798,000</td>
<td>3280</td>
<td>$800,000</td>
<td>3000</td>
</tr>
<tr>
<td>$849,000</td>
<td>3280</td>
<td>$800,000</td>
<td>3000</td>
</tr>
<tr>
<td>$598,000</td>
<td>2350</td>
<td>$600,000</td>
<td>2000</td>
</tr>
<tr>
<td>$648,000</td>
<td>2350</td>
<td>$600,000</td>
<td>2000</td>
</tr>
<tr>
<td>$654,000</td>
<td>2350</td>
<td>$600,000</td>
<td>2000</td>
</tr>
<tr>
<td>$798,000</td>
<td>3280</td>
<td>$700,000</td>
<td>3000</td>
</tr>
<tr>
<td>$598,000</td>
<td>1130</td>
<td>$500,000</td>
<td>1000</td>
</tr>
<tr>
<td>$598,000</td>
<td>2350</td>
<td>$500,000</td>
<td>2000</td>
</tr>
<tr>
<td>$849,000</td>
<td>3280</td>
<td>$700,000</td>
<td>3000</td>
</tr>
<tr>
<td>$430,000</td>
<td>650</td>
<td>$500,000</td>
<td>1000</td>
</tr>
</tbody>
</table>
9.5. Residuum and shift

Using residuated shifts, we can modify the previous example so that we compare similarity of tuples but put different emphasis on the similarity of prices and the similarity of property areas. Using (141) which is again denoted by $\triangleright$, we can use the following query:

$$\left(0.5 \rightarrow^5 \sigma_{\text{price1} \approx \text{price2}}(s)\right) \cap \left(0.75 \rightarrow^5 \sigma_{\text{sqft1} \approx \text{sqft2}}(s)\right).$$  

(143)

Described verbally, the ranks in the result of (143) represent degrees to which the tuples in $\triangleright$ have similar prices at least to degree 0.5 and similar areas at least to degree 0.75, i.e. we put more emphasis on the similarity of area than the similarity of prices. The result is

<table>
<thead>
<tr>
<th>price1</th>
<th>sqft1</th>
<th>price2</th>
<th>sqft2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>$430,000</td>
<td>650</td>
<td>$500,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$448,000</td>
<td>650</td>
<td>$500,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$598,000</td>
<td>1130</td>
<td>$500,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$598,000</td>
<td>2350</td>
<td>$500,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$598,000</td>
<td>2350</td>
<td>$600,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$598,000</td>
<td>2350</td>
<td>$700,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$598,000</td>
<td>2350</td>
<td>$800,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$648,000</td>
<td>2350</td>
<td>$500,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$648,000</td>
<td>2350</td>
<td>$600,000</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>1.000</td>
<td>$849,000</td>
<td>3280</td>
<td>$700,000</td>
</tr>
<tr>
<td>1.000</td>
<td>$849,000</td>
<td>3280</td>
<td>$800,000</td>
</tr>
<tr>
<td>0.994</td>
<td>$829,000</td>
<td>1760</td>
<td>$500,000</td>
</tr>
<tr>
<td>0.937</td>
<td>$675,000</td>
<td>1250</td>
<td>$500,000</td>
</tr>
<tr>
<td>0.937</td>
<td>$675,000</td>
<td>1250</td>
<td>$600,000</td>
</tr>
<tr>
<td>0.905</td>
<td>$829,000</td>
<td>1760</td>
<td>$500,000</td>
</tr>
<tr>
<td>0.879</td>
<td>$598,000</td>
<td>1130</td>
<td>$500,000</td>
</tr>
<tr>
<td>0.879</td>
<td>$598,000</td>
<td>1130</td>
<td>$600,000</td>
</tr>
<tr>
<td>0.874</td>
<td>$748,000</td>
<td>1840</td>
<td>$500,000</td>
</tr>
<tr>
<td>0.854</td>
<td>$748,000</td>
<td>1840</td>
<td>$600,000</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>0.274</td>
<td>$448,000</td>
<td>650</td>
<td>$800,000</td>
</tr>
<tr>
<td>0.250</td>
<td>$998,000</td>
<td>4580</td>
<td>$500,000</td>
</tr>
</tbody>
</table>

In order to illustrate the residuum of RDTs, consider the following query:

$$\pi_{\text{id, price}}\left(\sigma_{\text{price} \approx \$600,000} (\triangleright) \rightarrow_{\triangleright}^{\triangleright} \sigma_{\text{price} \approx \$750,000} (\triangleright)\right).$$

(144)

According to the interpretation of $\rightarrow_{\triangleright}$ as an implication, the ranks in the result of (144) can be read as expressing degrees to which, strictly speaking, it is true that if “house price is similar to $\$600,000$, then the house price is similar to $\$750,000$”. Taking into account the fact that $a \rightarrow b = 1$ iff $a \leq b$, each rank in the result of (144) has a natural interpretation as a degree to which it is true that “house price is similar more to the value of $\$750,000$ than to the value of $\$600,000$”. The result of the query is
By switching the arguments of $\rightarrow$ in (144), we obtain a query
\[
\pi_{\{\text{id, price}\}}\left(\sigma_{\text{price} \approx 750,000}(h \triangleright \bowtie f) \rightarrow h \triangleleft \bowtie f \sigma_{\text{price} \approx 600,000}(h \triangleright \bowtie f)\right)
\]
expressing that “house price is similar more to the value of $600,000 than to the value of $750,000” and yielding

<table>
<thead>
<tr>
<th>price</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>$430,000</td>
<td>87</td>
</tr>
<tr>
<td>$448,000</td>
<td>87</td>
</tr>
<tr>
<td>$598,000</td>
<td>45</td>
</tr>
<tr>
<td>$598,000</td>
<td>93</td>
</tr>
<tr>
<td>$648,000</td>
<td>82</td>
</tr>
<tr>
<td>$648,000</td>
<td>93</td>
</tr>
<tr>
<td>$654,000</td>
<td>45</td>
</tr>
<tr>
<td>$675,000</td>
<td>95</td>
</tr>
<tr>
<td>$748,000</td>
<td>63</td>
</tr>
<tr>
<td>$748,000</td>
<td>63</td>
</tr>
<tr>
<td>$798,000</td>
<td>71</td>
</tr>
<tr>
<td>$798,000</td>
<td>71</td>
</tr>
<tr>
<td>$798,000</td>
<td>85</td>
</tr>
<tr>
<td>$798,000</td>
<td>71</td>
</tr>
<tr>
<td>$829,000</td>
<td>85</td>
</tr>
<tr>
<td>$829,000</td>
<td>58</td>
</tr>
<tr>
<td>$998,000</td>
<td>85</td>
</tr>
<tr>
<td>$998,000</td>
<td>85</td>
</tr>
</tbody>
</table>

By taking the $\wedge$-intersection of (144) and (145), we may express a query showing “houses with price which is as similar to $600,000 as to $750,000”: 

<table>
<thead>
<tr>
<th>price</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>$675,000</td>
<td>95</td>
</tr>
<tr>
<td>$654,000</td>
<td>45</td>
</tr>
<tr>
<td>$648,000</td>
<td>82</td>
</tr>
<tr>
<td>$648,000</td>
<td>82</td>
</tr>
<tr>
<td>$598,000</td>
<td>45</td>
</tr>
<tr>
<td>$598,000</td>
<td>45</td>
</tr>
<tr>
<td>$598,000</td>
<td>93</td>
</tr>
<tr>
<td>$598,000</td>
<td>93</td>
</tr>
<tr>
<td>$798,000</td>
<td>71</td>
</tr>
<tr>
<td>$798,000</td>
<td>71</td>
</tr>
<tr>
<td>$798,000</td>
<td>85</td>
</tr>
<tr>
<td>$829,000</td>
<td>85</td>
</tr>
<tr>
<td>$829,000</td>
<td>85</td>
</tr>
<tr>
<td>$849,000</td>
<td>71</td>
</tr>
</tbody>
</table>
9.6. Division, subsethood, and similarity-based constraints

Relational division can be used to express various queries including queries involving graded subsethood (33). For instance, the following query yields a degree to which houses for sale offered for about $400,000 have the property area similar to 1000 square feet:

\[ S(\sigma_{\text{price} \approx 400,000}(h \bowtie f), \sigma_{\text{sqft} \approx 1000}(h \bowtie f)) \] (146)

By definition (33), the graded subsethood is in fact a particular case of residuated division:

\[ \sigma_{\text{sqft} \approx 1000}(h \bowtie f) \div 1 \sigma_{\text{price} \approx 400,000}(h \bowtie f) \] (147)

The formal difference is that the result of (146) is the degree 0.836 while the result of (147) is the RDT 0.836Φ on the empty relation scheme which corresponds to the degree 0.836. The converse inclusion

\[ S(\sigma_{\text{sqft} \approx 1000}(h \bowtie f), \sigma_{\text{price} \approx 400,000}(h \bowtie f)) \] (148)

yields the degree 0.601 as the result. Therefore, the similarity degree (34) given by

\[ E(\sigma_{\text{sqft} \approx 1000}(h \bowtie f), \sigma_{\text{price} \approx 400,000}(h \bowtie f)) \] (149)

is 0.601 because it is the infimum (in this case the minimum) of 0.836 and 0.601. The following query which involves division expresses the numbers of bedrooms of all houses built around the year 2000:

\[ \pi_{\{\text{type, bdrm}\}}(h) \div \pi_{\{\text{bdrm}\}}(h) \pi_{\{\text{type}\}}(\sigma_{\text{built} \approx 2000}(h)) \] (150)

The query result is

<table>
<thead>
<tr>
<th>bdrm</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.140</td>
<td>3</td>
</tr>
<tr>
<td>0.034</td>
<td>4</td>
</tr>
<tr>
<td>0.020</td>
<td>1</td>
</tr>
<tr>
<td>0.020</td>
<td>2</td>
</tr>
<tr>
<td>0.020</td>
<td>5</td>
</tr>
</tbody>
</table>

Note that the ordinary counterpart of the query would be trivial since there are no houses built exactly in the year 2000. Therefore, the result of the ordinary counterpart which in our case can be written (again, \( \sigma_{\text{built} = 2000}(h) \) is an abbreviation for \( h \bowtie [\text{built:2000}] \)) as

\[ \pi_{\{\text{type, bdrm}\}}(h) \div \pi_{\{\text{bdrm}\}}(h) \pi_{\{\text{type}\}}(\sigma_{\text{built} = 2000}(h)) \] (151)

would contain all numbers of bedrooms with rank 1, i.e. it would coincide with \( \pi_{\{\text{bdrm}\}}(h) \) (an implication with false antecedent is always true). The next example shows that using the residuated division and the support, we can express the least nonzero rank in an RDT. Recall that in (119) we have shown that by projections onto the empty set of attributes, we
can express the highest rank. So, getting the least nonzero rank can be seen as type of a “dual operation” on RDTs. Consider the query from (107). Now,

$$\sigma_{(\text{location} \approx \text{"Bay Rd"}) \otimes (\text{built} \approx 1995)} (h) \div^1 \nabla (\sigma_{(\text{location} \approx \text{"Bay Rd"}) \otimes (\text{built} \approx 1995)} (h))$$

(152)

yields 0.114, i.e. the least nonzero rank in the result of (107). It is easily seen that (152) is in fact a graded subsethood degree (33) of an RDT into its support.

**Remark 9.1:** Let us mention that the relational operations $\div$, $\triangledown \triangleleft$, $\cap$, and $\cup$ act like the corresponding operations of $L$ if applied to RDTs on the empty relation scheme. Indeed, it is easily seen that

$$(a \land b)_\emptyset = a_\emptyset \cap b_\emptyset,$$

$$(a \lor b)_\emptyset = a_\emptyset \cup b_\emptyset,$$

$$(a \otimes b)_\emptyset = a_\emptyset \rhd\land b_\emptyset = 1_\emptyset \rightarrow a_\emptyset \cdot b_\emptyset,$$

$$(a \rightarrow b)_\emptyset = b_\emptyset \rightarrow^1 a_\emptyset = a_\emptyset \rightarrow^1 b_\emptyset.$$

Hence, as in the classic relational model, $a_\emptyset (a \in L)$ play the role of truth degrees (now with general $a_\emptyset$ for intermediate degrees $0 < a < 1$) and the operations with the truth degrees (truth functions of the fundamental logical connectives) are derivable from the relational operations.

We conclude this section by demonstrating the difference between the ordinary equality-based integrity constraints and their similarity-based counterparts. Consider the ordinary referential integrity constraint asserting that all $id$s which appear in (the value of) $f$ also appear in (the value of) $h$ (i.e. all houses for sale have their corresponding entries in the table of all houses). In our model, we can express this type of constraint by

$$S(\nabla (\pi_{\{id\}}(f)), \nabla (\pi_{\{id\}}(h))) = 1$$

(153)

or alternative just by $\pi_{\{id\}}(f) \subseteq \pi_{\{id\}}(h)$ since both $f$ and $h$ are interpreted by nonranked RDTs. In any database instance of the database scheme in question, (153) is either matched or not. In contrast, we may postulate useful similarity-based constraints which may be matched to a general degree, not necessarily 0 or 1. For instance, a constraint saying that “houses with property area about 5000 have prices similar to $900,000$” can be seen as an example of such a (graded) constraint. In our model, we may formalize it as (48) by

$$S(\pi_{\{id\}}(\sigma_{\text{sqft} \approx 5000}(h \triangledown\approx f)), \pi_{\{id\}}(\sigma_{\text{price} \approx 900,000}(f)))$$

(154)

The degree to which (154) is satisfied in our database instance is 0.998, i.e. it is not fully satisfied but it is satisfied to a very high degree. Analogously, the following modification of the constraint

$$S(\pi_{\{id\}}(\sigma_{\text{sqft} \approx 2000}(h \triangledown\approx f)), \pi_{\{id\}}(\sigma_{\text{price} \approx 750,000}(f)))$$

(155)

is also satisfied to a high degree, namely 0.979. On the other hand, the satisfaction degree
for the following constraint saying that “houses with area about 3000 square feet are sold for approximately $500,000”:

\[
S(\pi_{\{id\}}(\sigma_{sqft \approx 3000}(h \ni f)), \pi_{\{id\}}(\sigma_{price \approx 500,000}(f)))
\] (156)

is not satisfied to a very high degree (in this case 0.616), at least compared to (154) and (155). Intuitively, this is quite expected since by looking at the data table in Figure 4, we may see that houses with property area about 3000 are sold for considerably higher prices than $500,000.

9.7. Extensions: residuated difference and generalized quantifiers

We illustrate here operations based on additional connectives discussed in Section 8.2. From the point of view of data modification, it may be desirable to have an independent residuated difference \( \ominus \) which satisfies (90). One particular choice in our case can be a binary operation \( \ominus : [0, 1]^2 \rightarrow [0, 1] \) given by

\[
a \ominus b = \begin{cases} 
0, & \text{if } a \leq b, \\
0, & \text{otherwise.}
\end{cases}
\]

which can be called the Gödel difference. The binary operation \( \oplus \) which is adjoint (91) to the Gödel \( \ominus \) coincides with \( \lor \). Now, for

\[
\sigma_{price \approx 900,000}(f)
\]

which yields

\[
\begin{array}{ccc}
\text{id} & \text{agent} & \text{price} \\
0.941 & 71 & Adams & $849,000 \\
0.917 & 58 & Black & $829,000 \\
0.896 & 85 & Black & $998,000 \\
0.879 & 71 & Black & $798,000 \\
0.814 & 63 & Adams & $748,000 \\
0.712 & 95 & Black & $675,000 \\
0.680 & 45 & Adams & $654,000 \\
0.671 & 82 & Adams & $648,000 \\
0.591 & 45 & Black & $598,000 \\
0.591 & 93 & Black & $598,000 \\
0.302 & 87 & Black & $448,000 \\
0.261 & 87 & Adams & $430,000 \\
\end{array}
\]

and for its modification

\[
\sigma_{price \approx 400,000}(f)
\]

which yields
we can consider the difference specifying “houses which are sold for about $900,000 minus houses which are sold for about $400,000”. In other words, the difference expresses “houses which are sold for about $900,000 whose prices are not close to $400,000”. Formally:

\[ \sigma_{\text{price} \approx 900,000} \ominus \sigma_{\text{price} \approx 400,000} \]  

which gives

Further extensions we mention in the paper involve graded quantifiers. For illustration, assume we want to query for types of houses for sale about $600,000. Normally, one would use a projection and a restriction like

\[ \pi_{\{\text{type}\}}(\sigma_{\text{price} \approx 600,000}(h \bowtie f)) \]  

to get a result of the following form:

\[ \begin{array}{c|c|c}
\text{id} & \text{agent} & \text{price} \\
\hline
0.941 & 71 & Adams \$849,000 \\
0.917 & 58 & Black \$829,000 \\
0.896 & 85 & Black \$998,000 \\
0.879 & 71 & Black \$798,000 \\
0.814 & 63 & Adams \$748,000 \\
0.712 & 95 & Black \$675,000 \\
0.680 & 45 & Adams \$654,000 \\
0.671 & 82 & Adams \$648,000 \\
\end{array} \]
\[{\text{type}}: x \in \{h(\text{type})\}\]

\((\exists x_1 \in \{h(\text{id})\})(\exists x_2 \in \{h(\text{location})\}) \cdots (\exists x_6 \in \{f(\text{agent})\})(\exists x_7 \in \{f(\text{price})\})\]

\(\{h(id: x_1, type: x, location: x_2, built: x_3, bdrm: x_4, sqft: x_5) \otimes f(id: x_1, agent: x_6, price: x_7) \otimes x_7 \approx \$600,000\}.\) (161)

According to the existential meaning of projection, the ranks in the result can be seen as degrees to which “there is house sold for about $600,000 of type t” where t ranges from Log Cabin to Residential. Suppose now, that someone is interested in many houses sold for about $600,000, i.e. the intended meaning of the ranks in the modified query result it the degree to which “there are many houses sold for about $600,000 of type t”. Clearly, such a result cannot be obtained by applying the ordinary projection. We may introduce a generalized quantifier “many” with the semantics defined as in (95) and then consider the DRC-RD query (161) with the existential quantifiers \(\exists\) all replaced by the quantifier “many”. In this particular case, we get the following result:

<table>
<thead>
<tr>
<th>type</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.373</td>
<td>Single Family</td>
</tr>
<tr>
<td>0.228</td>
<td>Ranch</td>
</tr>
<tr>
<td>0.146</td>
<td>Condominium</td>
</tr>
<tr>
<td>0.106</td>
<td>Log Cabin</td>
</tr>
<tr>
<td>0.094</td>
<td>Penthouse</td>
</tr>
<tr>
<td>0.052</td>
<td>Residential</td>
</tr>
</tbody>
</table>

We do not argue here that (95) is a faithful interpretation of “many” suitable for all purposes but at least in this example, it yields a result which is acceptable – looking at the data we might say that there are many Single Family houses for sale at $600,000, at least compared to houses of the other types. Notice that the semantics for the utilized quantifier “graded many” is domain independent, i.e. we have used a “relative many” with respect to the number of tuples with nonzero ranks in an RDT (other definitions like domain dependent “absolutely many” are also possible and may be of some interest).

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

This work was supported by the ECOP (Education for Competitiveness Operational Programme) [project number CZ.1.07/2.3.00/20.0059], which is co-financed by the European Social Fund and the state budget of the Czech Republic.
Notes on contributors

**Radim Belohlavek** received PhD degree in computer science from the Technical University of Ostrava, Czech Republic, in 1998, PhD degree in mathematics from Palacky University, Olomouc, Czech Republic, in 2001, and DSc degree in informatics and cybernetics from the Academy of Sciences of the Czech Republic in 2008. He is a professor of computer science at Palacky University. His academic interests are in discrete mathematics, logic, uncertainty and information, and data analysis. He published two books, Fuzzy Relational Systems: Foundations and Principles (Kluwer, 2002) and Fuzzy Equational Logic (Springer, 2005, with Vilem Vychodil) and over 150 papers in conference proceedings and journals. He is a senior member of IEEE (Institute of Electrical and Electronics Engineers), and a member of ACM (Association for Computing Machinery) and AMS (American Mathematical Society), and is a member of editorial boards of several international journals.

**Vilem Vychodil** received MSc degree in computer science (2002) and PhD (2004) degree in mathematics from Palacky University, Olomouc (Czech Republic). He is interested in logic in computer science.

References


