Math. Struct. in Comp. Science (2012), *vol.* 22, *pp.* 533–547.
doi:10.1017/S0960129511000673 First published online 14 December 2011

Formalising PFSQL queries using ŁΠ$\frac{1}{2}$ fuzzy logic[†]

ALEKSANDAR PEROVIĆ[‡], ALEKSANDAR TAKAČI[§] and SRDJAN ŠKRBIĆ[¶]

[‡]*Faculty of Transportation and Traffic Engineering, University of Belgrade, Serbia*
Email: pera@sf.bg.ac.rs
[§]*Faculty of Technology, University of Novi Sad, Serbia*
Email: stakaci@tf.uns.ac.rs
[¶]*Faculty of Science, University of Novi Sad,*
Serbia Trg Dositeja Obradovića 3, 21000 Novi Sad, Serbia
Email: shkrba@uns.ac.rs

Received 23 February 2011; revised 15 September 2011

Using the concept of a generalised priority constraint satisfaction problem, we previously found a way to introduce priority queries into fuzzy relational databases. The results were PFSQL (Priority Fuzzy Structured Query Language) together with a database independent interpreter for it. In an effort to improve the performance of the resolution of PFSQL queries, the aim of the current paper is to formalise PFSQL queries by obtaining their interpretation in an existing fuzzy logic. We have found that the ŁΠ$\frac{1}{2}$ logic provides sufficient elements. The SELECT line of PFSQL queries is semantically a formula of some fuzzy logic, and we show that such formulas can be naturally expressed in a conservative extension of the ŁΠ$\frac{1}{2}$ logic. Furthermore, we prove a theorem that gives the PSPACE containment for the complexity of finding a model for a given ŁΠ$\frac{1}{2}$ logic formula.

1. Introduction

The relational model's inability to model uncertain and incomplete data can be viewed as one of its disadvantages. The idea of using fuzzy sets and fuzzy logic to extend existing database models to include these possibilities has been around since the 1980s. However, although this area has been researched for a long time, working implementations are rare.

The literature contains references to several models of fuzzy knowledge representation in relational databases. For example, the Buckles–Petry model (Buckles and Petry 1982) was the first model to introduce similarity relations in the relational model. The GEFRED (Generalised Model of Fuzzy Relational Databases) model (Medina *et al.* 1994) is a possibilistic model that refers to generalised fuzzy domains and allows the possibility distribution in domains. It includes the case where the underlying domain is not purely numeric, but contains scalars of any type. It also contains the notion of unknown, undefined and null values. Subsequently, it has been expanded in various ways – see, for example, Galindo *et al.* (1999; 2001). A first approach to incorporating fuzzy logic in ER

[†] The authors are grateful for the support of the Serbian Ministry of Education and Science through grants III041013, III47003, TR36001 and OI174009.

(Entity–Relationship) models was made in Zvieli and Chen (1986), which allowed fuzzy attributes in entities and relationships. Chen and Kerre (1998) introduced fuzzy extensions of several major EER (Extended Entity–Relationship) concepts. Chaudry *et al.* (1994) suggested a method for designing fuzzy relational databases following Zvieli and Chen's extension of the ER model by proposing a way to convert a crisp database into a fuzzy one. Galindo *et al.* (2006) described a way to use fuzzy EER to model the database together with a detailed method for representing the fuzzy knowledge modelled in this way using relational databases. They also described the specification and implementation of FSQL, which is an advanced SQL language with fuzzy capabilities. Another recent approach to fuzzy SQL language design that includes a full implementation is given in Hudec (2009).

In Takači and Škrbić (2005), we studied the possibilities for extending the relational model with fuzzy logic capabilities. We elaborated on this in Takači and Škrbić (2008) and gave a detailed model of a fuzzy relational database. Moreover, using the concept of a Generalised Priority Constraint Satisfaction Problem (GPFCSP) from Takači (2005) and Luo *et al.* (2003), the authors have found a way to introduce priority queries into an FRDB (Fuzzy Relational Database). The result was the PFSQL (Priority Fuzzy Structured Query Language) query language. Takači and Škrbić (2007; 2008), Škrbić and Racković (2009) and Škrbić *et al.* (2011) introduced similarity relations on the fuzzy domain for the evaluation of FRDB conditions. They also described a complete solution for a fuzzy relational database application development and gave the architecture of the PFSQL interpreter together with the data model that this implementation is based on – we give a brief overview of this system in Section 2.

Unfortunately, this implementation of the PFSQL interpreter has limited performance. The main goal set earlier in the process of modelling and implementating the PFSQL interpreter was the creation of a database-independent interpreter prototype that was close to the final user and suitable for wider use. In an effort to find a better method of implementation, and to improve the performance of the resolution of PFSQL queries, we have focused on the possibilities of formalising the query structure. The formal development of fuzzy logic is a well-developed area – various Hilbert style axiomatisations can be found in Hájek (1998), Esteva *et al.* (2000) and Cintula *et al.* (2007). In order to obtain a complete axiomatisation of PFSQL queries, we have used the interpretation method. The aim of the current paper is to obtain an interpretation of PFSQL queries in an existing fuzzy logic. We found that the $\text{Ł}\Pi\frac{1}{2}$ logic provides enough elements to interpret these queries.

The SELECT line of PFSQL queries is semantically a formula of some fuzzy logic. As we will show, such formulas can be expressed naturally in the $\text{Ł}\Pi\frac{1}{2}$ logic, which is an amalgam of Łukasiewicz and Product logics. We will illustrate the formal interpretation through an example. We will also prove an original theorem about the PSPACE containment for the complexity of finding a model for a given $\text{Ł}\Pi\frac{1}{2}$ logic formula.

1.1. *Structure of the paper*

The next section describes aspects of the existing solution for a fuzzy relational database application development and gives the architecture of the PFSQL interpreter. In Section 3,

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

we introduce the ŁΠ$\frac{1}{2}$ logic and prove the theorem that gives PSPACE containment for the complexity of finding a model for a given ŁΠ$\frac{1}{2}$ logic formula. In Section 4, we describe the conservative extension of the ŁΠ$\frac{1}{2}$ logic that allows its association with PFSQL queries. In Section 5, we illustrate the PFSQL query transformation to ŁΠ$\frac{1}{2}$ logic construct and the evaluation of a formula acquired in this way. Finally, we present our conclusions in Section 6.

2. The existing solution

In this section we describe the existing system with a view to motivating the introduction of the ŁΠ$\frac{1}{2}$ formalisation. We will omit numerous details and subtleties, but give references that contain complete descriptions of the existing solution and its components.

In order to allow the use of fuzzy values in SQL queries, we extended the classical SQL with several new elements. In addition to fuzzy capabilities, we have added the ability to specify priorities for fuzzy statements. We have called the query language constructed in this way Priority Fuzzy SQL – PFSQL.

The basic difference between SQL and PFSQL interpreters is in the way the database processes records. In a classical relational database, queries are executed so that a tuple is either accepted in the result set, if it fulfills the conditions given in a query, or removed from the result set if it does not fulfill the conditions. In other words, every tuple is given a value true (1) or false (0). By contrast, PFSQL returns a fuzzy relation on the database as the result set. Every tuple considered in the query is given a value from the unit interval, and this value is calculated using fuzzy logic operators.

2.1. *PFSQL*

Details of the PFSQL syntax can be found in Takači and Škrbić (2008; 2009). Here we will just give some pointers to those elements of classical SQL that are extended. The variables in a query should be allowed to have both crisp and fuzzy values, so we need to allow relational operators between different types of fuzzy values as well as between fuzzy and crisp values. Next, we introduced and used the possibility based ordering on the set of fuzzy numbers in the queries. This suggests the introduction of set functions like MIN, MAX and COUNT in PFSQL.

Classical SQL includes the ability to combine conditions using logical operators. This feature, of course, also forms part of our fuzzy extensions, so the ability to combine fuzzy conditions is also required in our implementation. Values are calculated using t-norms, t-conorms and 'strict' negation. Queries are handled using priority fuzzy logic, which is based on GPFCSP (Generalised Priority Fuzzy Constraint Satisfaction Problem) systems (Takači *et al.* 2008; Luo *et al.* 2003; Takači 2005; Takači and Škrbić 2008).

The complete EBNF (Extended Backus-Naur Form) syntax of the PFSQL language can be found in Škrbić and Racković (2009). For brevity, we will not present every possible construction of the language here, but just present some examples of queries that give an overview of the possibilities and extensions built into PFSQL. The queries are executed against a hypothetical student database.

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

The first query returns the names and surnames of students whose GPA (Grade Point Average) is equal to the given triangular fuzzy number:

```
SELECT msd.name, msd.surname FROM MainStudentData msd WHERE
(msd.GPA=#triangle(9,1,0.4)#)
```

The # symbol is chosen to mark fuzzy constants. The construction #triangle(9,1,0.4)# denotes a triangular fuzzy number with peak at 9 and with 1 and 0.4 as left and right offset. In other words, the membership function $f : [6, 10] \longrightarrow [0, 1]$ of the fuzzy set $triangle(9, 1, 0.4)$ is given by

$$f(x) = \begin{cases} 0 & 6 \leqslant x < 8 \text{ or } 9.4 < x \leqslant 10 \\ x - 8 & 8 \leqslant x \leqslant 9 \\ -2.5x + 23.5 & 9 < x \leqslant 9.4\,. \end{cases}$$

If we define a linguistic label 'average GPA' with the value triangle(9,1,0.4), the previous query could have been simplified to

```
SELECT msd.name, msd.surname FROM MainStudentData msd WHERE
(msd.GPA=#ling(averageGPA)#)}
```

Queries like these can be enriched with additional constraints. The next query returns names and surnames of students that have average GPA with priority 0.7, and GPA during the fourth year equal to 8.5 with priority 0.4. The query also contains a threshold clause that limits the results and removes tuples with fuzzy satisfaction degree smaller than 0.2:

```
SELECT msd.name, msd.surname FROM MainStudentData msd WHERE
(msd.GPA=ling(averageGPA) PRIORITY 0.7) AND
      (msd.GPA4=8.5 PRIORITY 0.4)
THRESHOLD 0.2
```

As we mentioned earlier, the aggregate functions MAX, MIN and COUNT can take fuzzy value as an argument. The next query illustrates the use of the aggregate function MIN to return the minimal GPA:

```
SELECT MIN(msd.GPA)
FROM MainStudentData msd
```

If we assume that the variable msd.GPA is fuzzy, the execution of this query becomes complex because it includes the ordering of fuzzy values. For example, we could get the value $triangle(6.9, 0.4, 0.7)$ as a result.

In the case of classical SQL queries, it is clear how to assign a truth value to every elementary condition. With fuzzy attributes, the situation becomes more complex. We first assign a truth value from the unit interval to every elementary condition. The only way to do this is to implement a set of algorithms that calculate truth values for every possible combination of values in a query and values in the database. For instance, if a query contains a condition that compares a trapezoidal fuzzy number value with a triangular fuzzy number in a database, we must have an algorithm that calculates the compatibility degree of the two fuzzy sets. After the truth values from the unit interval are assigned, they are aggregated using fuzzy logic operators. We use a t-norm for the operator AND, and its dual t-conorm for the operator OR. For negation, we use the strict negation

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

$N(x) = 1 - x$. In the case of priority statements, we use the GPFCSP system rules to calculate the result.

We will now describe processes that allow PFSQL queries to be executed. The basic idea is to first transform the PFSQL query into something that a classical SQL interpreter understands. Specifically, conditions with fuzzy attributes are removed from the WHERE clause. Attributes from these conditions are moved up into the SELECT clause. In this way, conditions containing fuzzy constructs are eliminated, so that a database will return all the tuples – both those that fulfill fuzzy conditions and those that do not. As a result of this transformation, we get a classical SQL query. When this query is executed against a database, the results are interpreted using fuzzy mechanisms. These mechanisms assign a value (membership degree) from the unit interval to every tuple in a result set. In this way, the result set contains all tuples – those that partially fulfill query conditions and those that fully fulfill them. In addition, every tuple in the result set has an associated membership degree. This membership degree is a measure of how much a tuple belongs to a result set. If a threshold is given, all the tuples in a result set that have a membership degree below the threshold are removed.

2.2. *Fuzzy-relational database*

It is now clear that the PFSQL implementation has to rely on meta-data related to fuzzy attributes that reside inside a database. To this end, we have defined a FRDB data model, which we will describe briefly in this section.

Our FRDB data model allows data values to be any fuzzy subset of the attribute domain. A user only needs to specify a membership function of a fuzzy set. Hypothetically, we should have an algorithm for each fuzzy set that calculates the values of its membership function. This would lead to a large spatial complexity of the database. This is why we only allow well-known standard types of fuzzy sets (triangular, trapezoidal, and so on) as attribute values. If a type of a fuzzy set is introduced, we only need to store the parameters necessary to calculate the values of its membership function.

We have introduced one further extension to the set of attribute types in the form of linguistic labels, which are just named fuzzy values from the domain. Bearing these extensions in mind, we can define a domain of a fuzzy attribute as follows:

$$D = C_D \cup F_D \cup L_D$$

where C_D is a classical attribute domain, F_D is the set of all fuzzy subsets of the domain and L_D is the set of linguistic labels.

In order to represent these fuzzy values in a database, we extend this model with additional tables that make up the fuzzy meta-data model. Several tables are introduced to cover all described requirements. Details of the structure of this model can be found in Takači and Škrbić (2008) and Škrbić *et al.* (2011).

3. ŁΠ$\frac{1}{2}$ logic

ŁΠ$\frac{1}{2}$ logic is a fuzzy logic combining Łukasiewicz logic and Product logic. The primitive connectives of ŁΠ$\frac{1}{2}$ are:

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

— $\odot$ (product conjunction)
— $\to_L$ (Łukasiewicz implication)
— $\to_\Pi$ (product implication)
— truth constants $\overline{0}$ and $\overline{\frac{1}{2}}$.

In this context, $\overline{0}$ and $\overline{\frac{1}{2}}$ are purely syntactic objects, and should not be identified with 0 and 1. The intended meaning of $\overline{a}$ is to emphasis the canonical interpretation: $\overline{a}$ is interpreted as a in the canonical (standard) model. For instance, $\overline{\frac{1}{2}}$ is interpreted as $\frac{1}{2}$ in the standard model $([0,1], *_\Pi, \Rightarrow_L, \Rightarrow_\Pi, 0, \frac{1}{2})$ of the $\text{Ł}\Pi\frac{1}{2}$ logic. Here:

— $*_\Pi$ is the product t-norm on $[0,1]$;
— $\Rightarrow_L$ is the Łukasiewicz implication on $[0,1]$; and
— $\Rightarrow_\Pi$ is the product implication on $[0,1]$.

The reasons for this choice of primitive connectives are purely technical. For instance, negations (Łukasiewicz, product) can be formally defined using the appropriate implication and $\overline{0}$ by $\phi \to_* \overline{0}$, $* \in \{L, \Pi\}$. Furthermore, all rational numbers from the real unit interval can be formally defined by means of connectives and truth constants $\overline{0}$ and $\overline{\frac{1}{2}}$ – see Example 3.1 below.

The axioms and inference rules of $\text{Ł}\Pi\frac{1}{2}$ can be found in Esteva *et al.* (2001). Semantically, the above connectives are evaluated as follows:

$$e(\overline{0}) = 0$$
$$e\left(\overline{\frac{1}{2}}\right) = \frac{1}{2}$$
$$e(\phi \odot \psi) = e(\phi) \cdot e(\psi)$$
$$e(\phi \to_L \psi) = \min(1, 1 - e(\phi) + e(\psi))$$
$$e(\phi \to_\Pi \psi) = \begin{cases} 1 & e(\phi) \leqslant e(\psi) \\ \dfrac{e(\psi)}{e(\phi)} & e(\phi) > e(\psi). \end{cases}$$

Note that both Łukasiewicz implication and product implication behave like orderings. The following connectives (we will give them semantically) can be defined in $\text{Ł}\Pi\frac{1}{2}$:

$$e(\neg_L \phi) = 1 - e(\phi) \qquad \text{(Łukasiewicz negation)}$$
$$e(\phi \oplus \psi) = \min(1, e(\phi) + e(\psi)) \qquad \text{(Łukasiewicz disjunction)}$$
$$e(\phi \& \psi) = \max(0, e(\phi) + e(\psi) - 1) \qquad \text{(Łukasiewicz conjunction)}$$
$$e(\neg_\Pi \phi) = \begin{cases} 1 & e(\phi) = 0 \\ 0 & e(\phi) > 0 \end{cases} \qquad \text{(product negation)}$$
$$e(\phi \vee_\Pi \psi) = e(\phi) + e(\psi) - e(\phi) \cdot e(\psi) \qquad \text{(product disjunction)}$$
$$e(\triangle \phi) = \begin{cases} 1 & e(\phi) = 1 \\ 0 & e(\phi) < 1 \end{cases}$$
$$e(\nabla \phi) = \begin{cases} 1 & e(\phi) > 0 \\ 0 & e(\phi) = 0 \end{cases}$$
$$e(\phi \ominus \psi) = \max(0, e(\phi) - e(\psi))$$

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

$$e(\phi \wedge \psi) = \min(e(\phi), e(\psi)) \qquad \text{(Gödel conjunction)}$$
$$e(\phi \vee \psi) = \max(e(\phi), e(\psi)) \qquad \text{(Gödel disjunction)}$$
$$e(\phi \equiv \psi) = 1 - |e(\alpha) - e(\beta)|$$
$$e(\phi \ \vee_{pr} \ \psi) = e(\phi \vee_{\Pi} \neg_L \psi). \qquad \text{(priority operator)}$$

Example 3.1. All rational numbers in the real unit interval are definable in $\text{Ł}\Pi\frac{1}{2}$. Indeed, if m, n and k are positive integers such that $m < n < k$, then m/n can be represented by

$$\underbrace{(\phi \oplus \cdots \oplus \phi)}_{n} \rightarrow_{\Pi} \underbrace{(\phi \oplus \cdots \oplus \phi)}_{m},$$

where ϕ is the formula $\underbrace{\overline{\frac{1}{2}} \odot \cdots \odot \overline{\frac{1}{2}}}_{k}$. For practical purposes, in this paper we will denote rational numbers $\overline{\frac{m}{n}}$. Finally, we can represent 1 by $\overline{0} \rightarrow_L \overline{0}$.

In many ways, $\text{Ł}\Pi\frac{1}{2}$ logic resembles the first-order theory of the reals (RCF), at least in terms of expressivity. For instance, Marchioni and Montagna have recently shown that a continuous t-norm is RCF-definable if and only if it is $\text{Ł}\Pi\frac{1}{2}$-definable (Marchioni and Montagna 2008). Simple and coherent conditional probabilities have been formally represented using $\text{Ł}\Pi\frac{1}{2}$ logic by Hájek, Godo, Esteva, Marchioni and others (Godo *et al.* 2000; Godo and Marchioni 2006; Hájek *et al.* 1995). Complete axiomatisations of $\text{Ł}\Pi\frac{1}{2}$ logic can be found in Esteva *et al.* (2001) and Hájek *et al.* (1996).

The next theorem will illustrate the expressive power of $\text{Ł}\Pi\frac{1}{2}$ logic: specifically, we will show that systems of polynomial inequalities can be formally represented in $\text{Ł}\Pi\frac{1}{2}$ logic. As far as we know, the form of the theorem and the proof we present are original.

Theorem 3.1. Let $\varphi(x_1, \ldots, x_n)$ be any quantifier free formula of the language of ordered fields $L = \{+, \cdot, \leqslant, 0, 1\}$ and let $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{Q}$, $a_i < b_i$. Then, there is an $\text{Ł}\Pi\frac{1}{2}$-formula $\phi(p_1, \ldots, p_n)$ such that

$$\mathbb{R} \models \exists x_1 \ldots \exists x_n \left(\bigwedge_{i=1}^{n} a_i \leqslant x_i \leqslant b_i \ \wedge \ \varphi(x_1, \ldots, x_n) \right)$$

if and only if $\phi(p_1, \ldots, p_n)$ is satisfiable.

Proof. Up to equivalence, $\varphi(x_1, \ldots, x_n)$ has the form

$$\bigvee_{i=1}^{r} \left(\bigwedge_{j=1}^{s} f_{i,j}(\bar{x}) \geqslant 0 \ \wedge \ \bigwedge_{j'=1}^{s'} g_{i,j'}(\bar{x}) > 0 \right),$$

where $\bar{x} = x_1, \ldots, x_n$ and $f_{i,j}(\bar{x}), g_{i,j'}(\bar{x})$ are polynomials with rational coefficients. Without loss of generality, we may assume that all coefficients appearing in polynomials are rational numbers from the real unit interval. Indeed, if $f(\bar{x})$ is a polynomial with rational coefficients and $M > 0$ is a rational number such that $M > |c|$ for each coefficient c of $f(\bar{x})$, then

$$\mathbb{R} \models \exists \bar{x} (f(\bar{x}) \geqslant 0) \ \text{ iff } \ \mathbb{R} \models \exists \bar{x} \left(\frac{1}{M} f(\bar{x}) \geqslant 0 \right),$$

so we can use $h(\bar{x}) = \frac{1}{M} f(\bar{x})$ instead of $f(\bar{x})$.

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

Furthermore, we may assume that $a_1 = \cdots = a_n = 0$ and $b_1 = \cdots = b_n = 1$, since

$$\mathbb{R} \models \exists \bar{x} \left(\bigwedge_{i=1}^{n} a_i \leqslant x \leqslant b_i \wedge f(\bar{x}) \geqslant 0 \right)$$

if and only if

$$\mathbb{R} \models \exists \bar{x} \left(\bigwedge_{i=1}^{n} 0 \leqslant x \leqslant 1 \wedge f\big((b_1 - a_1)x + a_1, \ldots, (b_n - a_n)x + a_n\big) \geqslant 0 \right).$$

Finally, we may assume that the sums of absolute values of all monomials do not exceed 1.

We are now ready to construct the required $\text{Ł}\Pi\frac{1}{2}$-formula $\phi(p_1, \ldots, p_n)$. Each p_i will represent the variable x_i. The monomial $a \cdot x_{i1}^{m_1} \cdots x_{i_s}^{m_s}$ that appears in some of the given polynomials will be represented by an $\text{Ł}\Pi\frac{1}{2}$-formula

$$\overline{a} \odot (\underbrace{p_{i_1} \odot \cdots \odot p_{i_1}}_{m_1}) \odot \cdots \odot (\underbrace{p_{i_s} \odot \cdots \odot p_{i_s}}_{m_s}).$$

Recall that $\overline{a}$ is the formal name (syntactical representation) for the rational number a: for example, $\overline{0}$ is the name for 0. This representation is adequate since

$$e\big(\overline{a} \odot (\underbrace{p_{i_1} \odot \cdots \odot p_{i_1}}_{m_1}) \odot \cdots \odot (\underbrace{p_{i_s} \odot \cdots \odot p_{i_s}}_{m_s})\big) = a \cdot e(p_{i1})^{m_1} \cdots e(p_{i_s})^{m_s}.$$

Let

$$f_{i,j}(\bar{x}) = \mu_{i,j,1}(\bar{x}) + \cdots + \mu_{i,j,k_{i,j}}(\bar{x}) - \nu_{i,j,1}(\bar{x}) - \cdots - \nu_{i,j,l_{i,j}}(\bar{x}),$$

where $\mu_{i,j,k}(\bar{x})$ and $\nu_{i,j,l}(\bar{x})$ are monomials. If ϕ_k represents $\mu_{i,j,k}(\bar{x})$ and ψ_k represents $\nu_{i,j,l}(\bar{x})$, then the $\text{Ł}\Pi\frac{1}{2}$ formula

$$(\phi_1 \oplus \cdots \oplus \phi_{k_{i,j}}) \ominus (\psi_1 \oplus \cdots \oplus \psi_{l_{i,j}})$$

represents polynomial $f_{i,j}(\bar{x})$. We can represent polynomials $g_{i,j'}(\bar{x})$ similarly.

Let $\phi_{i,j}, \psi_{i,j'}$ be $\text{Ł}\Pi\frac{1}{2}$-formulas that represent polynomials $f_{i,j}(\bar{x})$ and $g_{i,j'}(\bar{x})$, respectively. We define ϕ as follows:

$$\bigvee_{i=1}^{r} \left(\bigwedge_{j=1}^{s} \phi_{i,j} \wedge \bigwedge_{j'=1}^{s'} \neg_{\Pi} \neg_{\Pi} \psi_{i,j'} \right).$$

Notice that for any truth evaluation e, we have

$$f_{i,j}\big(e(p_1), \ldots, e(p_n)\big) \geqslant 0$$

if and only if $e(\phi_{i,j}) = 1$, and

$$g_{i,j'}\big(e(p_1), \ldots, e(p_n)\big) > 0$$

if and only if $e\big(\neg_{\Pi} \neg_{\Pi} \psi_{i,j'}\big) = 1$, that is, $e(\psi_{i,j'}) > 0$, which gives ourclaim. □

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

4. FRDB formalisation

As we have seen, rational numbers can be represented in $\text{Ł}\Pi\frac{1}{2}$, so hard constraints are expressible in $\text{Ł}\Pi\frac{1}{2}$. In order to capture soft constraints (trapezoidal, fuzzy quantities), we will define the following conservative extension (extension by definitions) of $\text{Ł}\Pi\frac{1}{2}$:

(1) For each $0 \leqslant a < b < c < d \leqslant 1$, $a, b, c, d \in \mathbb{Q}$, we will introduce a new unary connective $[a, b, c, d]$ and add to $\text{Ł}\Pi\frac{1}{2}$ the following axioms:

$$\left(\triangle\left(\phi \to_L \overline{a}\right) \vee \triangle\left(\overline{d} \to_L \phi\right)\right) \to_L \left([a, b, c, d]\phi \equiv \overline{0}\right) \tag{a}$$

$$\left(\triangle\left(\overline{b} \to_L \phi\right) \wedge \triangle\left(\phi \to_L \overline{c}\right)\right) \to_L \left([a, b, c, d]\phi \equiv \overline{1}\right) \tag{b}$$

$$\begin{aligned}&\left(\triangle\left(\overline{a} \to_L \phi\right) \wedge \triangle\left(\phi \to_L \overline{b}\right) \wedge \right.\\ &\left.\neg_\Pi\, \triangle(\phi \equiv a)) \;\wedge \neg_\Pi(\triangle(\phi \equiv b))\right)\end{aligned} \to_L \left([a, b, c, d]\phi \equiv \left(\left(\phi \odot \overline{\frac{1}{b-a}}\right) \ominus \overline{\frac{a}{b-a}}\right)\right) \tag{c}$$

$$\begin{aligned}&\left(\triangle\left(\overline{c} \to_L \phi\right) \wedge \triangle\left(\phi \to_L \overline{d}\right) \wedge \right.\\ &\left.\neg_\Pi(\triangle(\phi \equiv c)) \wedge \neg_\Pi(\triangle(\phi \equiv d))\right)\end{aligned} \to_L \left([a, b, c, d]\phi \equiv \left(\overline{\frac{d}{d-c}} \ominus \left(\phi \odot \overline{\frac{1}{d-c}}\right)\right)\right). \tag{d}$$

Notice that (a)–(d) actually formalise the trapezoidal fuzzy number

$$[a, b, c, d] : [0, 1] \longrightarrow [0, 1]$$

defined by

$$[a, b, c, d](x) = \begin{cases} 0 & x \leqslant a \text{ or } x \geqslant d \\ 1 & b \leqslant x \leqslant c \\ \frac{x}{b-a} - \frac{a}{b-a} & a < x < b \\ \frac{d}{d-c} - \frac{x}{d-c} & c < x < d. \end{cases}$$

(2) For each $0 \leqslant a < b < c \leqslant 1$, $a, b, c \in \mathbb{Q}$, we will introduce a new unary connective $[a, b, c]$ and add to $\text{Ł}\Pi\frac{1}{2}$ the following axioms:

$$\left(\triangle\left(\phi \to_L \overline{a}\right) \vee \triangle\left(\overline{c} \to_L \phi\right)\right) \to_L \left([a, b, c]\phi \equiv \overline{0}\right) \tag{a}$$

$$\triangle(\phi \equiv \overline{b}) \to_L \left([a, b, c]\phi \equiv \overline{1}\right) \tag{b}$$

$$\begin{aligned}&\left(\triangle\left(\overline{a} \to_L \phi\right) \wedge \triangle\left(\phi \to_L \overline{b}\right) \wedge \right.\\ &\left.\neg_\Pi\, \triangle(\phi \equiv a)) \wedge \neg_\Pi(\triangle(\phi \equiv b))\right)\end{aligned} \to_L \left([a, b, c]\phi \equiv \left(\left(\phi \odot \overline{\frac{1}{b-a}}\right) \ominus \overline{\frac{a}{b-a}}\right)\right) \tag{c}$$

$$\begin{aligned}&\left(\triangle\left(\overline{b} \to_L \phi\right) \wedge \triangle\left(\phi \to_L \overline{c}\right) \wedge \right.\\ &\left.\neg_\Pi(\triangle(\phi \equiv b)) \wedge \neg_\Pi(\triangle(\phi \equiv c))\right)\end{aligned} \to_L \left([a, b, c]\phi \equiv \left(\overline{\frac{c}{c-b}} \ominus \left(\phi \odot \overline{\frac{1}{c-b}}\right)\right)\right), \tag{d}$$

which formalises the triangular fuzzy number $[a, b, c] : [0, 1] \longrightarrow [0, 1]$ defined by

$$[a, b, c](x) = \begin{cases} 0 & x \leqslant a \text{ or } c \leqslant x \\ 1 & x = b \\ \frac{x}{b-a} - \frac{a}{b-a} & a < x < b \\ \frac{c}{c-b} - \frac{x}{c-b} & b < x < c. \end{cases}$$

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

(3) For each $0 \leqslant a < b \leqslant 1$, $a, b \in \mathbb{Q}$, we will introduce new unary connectives (a,b), $(a,b]$, $[a,b)$ and $[a,b]$, and add the following axioms to $\mathbf{L}\Pi\frac{1}{2}$:

$$\left(\triangle(\overline{a} \to_L \phi) \wedge \triangle(\phi \to_L \overline{b}) \wedge \neg_\Pi \triangle(\phi \equiv \overline{a}) \wedge \neg_\Pi \triangle(\phi \equiv \overline{b})\right) \to_L ((a,b)\phi \equiv \overline{1}) \quad \text{(a)}$$

$$\left(\triangle(\phi \to_L \overline{a}) \vee (\triangle(\overline{b} \to_L \phi)\right) \to_L ((a,b)\phi \equiv \overline{0}) \quad \text{(b)}$$

$$\left(\triangle(\overline{a} \to_L \phi) \wedge \triangle(\phi \to_L \overline{b}) \wedge \neg_\Pi \triangle(\phi \equiv \overline{a})\right) \to_L ((a,b]\phi \equiv \overline{1}) \quad \text{(c)}$$

$$\left(\neg_\Pi \triangle(\phi \equiv \overline{a}) \wedge (\triangle(\phi \to_L \overline{b}) \vee (\triangle(\overline{b} \to_L \phi))\right) \to_L ((a,b]\phi \equiv \overline{0}) \quad \text{(d)}$$

$$\left(\triangle(\overline{a} \to_L \phi) \wedge \triangle(\phi \to_L \overline{b}) \wedge \neg_\Pi \triangle(\phi \equiv \overline{b})\right) \to_L ([a,b)\phi \equiv \overline{1}) \quad \text{(e)}$$

$$\left(\neg_\Pi \triangle(\phi \equiv \overline{a}) \wedge (\triangle(\phi \to_L \overline{a}) \vee (\triangle(\overline{b} \to_L \phi))\right) \to_L ([a,b)\phi \equiv \overline{0}) \quad \text{(f)}$$

$$\left(\triangle(\overline{a} \to_L \phi) \wedge \triangle(\phi \to_L \overline{b})\right) \to_L ([a,b]\phi \equiv \overline{1}) \quad \text{(g)}$$

$$\left(\neg_\Pi \triangle(\phi \equiv \overline{a}) \wedge \neg_\Pi \triangle(\phi \equiv \overline{b}) \wedge (\triangle(\phi \to_L \overline{a}) \vee (\triangle(\overline{b} \to_L \phi))\right) \to_L ([a,b]\phi \equiv \overline{0}), \quad \text{(h)}$$

which formalises intervals with rational endpoints.

(4) For each $0 \leqslant a_1 < \cdots < a_n \leqslant 1$, $a_1, \ldots, a_n \in \mathbb{Q}$, we will define a new unary connective $\{a_1, \ldots, a_n\}$ and add to $\mathbf{L}\Pi\frac{1}{2}$ the following axioms:

$$\bigwedge_{i=1}^{n} \neg_\Pi \triangle(\phi \equiv \overline{a_i}) \to_L (\{a_1, \ldots, a_n\}\phi \equiv \overline{0}) \quad \text{(a)}$$

$$\bigvee_{i=1}^{n} \triangle(\phi \equiv \overline{a_i}) \to_L (\{a_1, \ldots, a_n\}\phi \equiv \overline{1}). \quad \text{(b)}$$

which formalises crisp (finite) sets.

(5) For each $0 \leqslant a < b \leqslant 1$, $a, b \in \mathbb{Q}$, we will introduce new unary connectives $[a, b, \uparrow]$ and $[a, b, \downarrow]$ and add to $\mathbf{L}\Pi\frac{1}{2}$ the following axioms:

$$\triangle(\phi \to_L \overline{a}) \to_L ([a,b,\uparrow]\phi \equiv \overline{0}) \quad \text{(a)}$$

$$\triangle(\overline{b} \to_L \phi) \to_L ([a,b,\uparrow]\phi \equiv \overline{1}) \quad \text{(b)}$$

$$\left(\triangle(\overline{a} \to_L \phi) \wedge \triangle(\phi \to_L \overline{b}) \wedge \neg_\Pi \triangle(\phi \equiv a) \wedge \neg_\Pi \triangle(\phi \equiv b)\right) \to_L \left([a,b,\uparrow]\phi \equiv \left(\left(\phi \odot \overline{\frac{1}{b-a}}\right) \ominus \overline{\frac{a}{b-a}}\right)\right) \quad \text{(c)}$$

$$\triangle(\phi \to_L \overline{a}) \to_L ([a,b,\downarrow]\phi \equiv \overline{1}) \quad \text{(d)}$$

$$\triangle(\overline{b} \to_L \phi) \to_L ([a,b,\downarrow]\phi \equiv \overline{0}) \quad \text{(e)}$$

$$\left((\overline{a} \to_L \phi) \wedge (\phi \to_L \overline{b}) \wedge \neg_\Pi \triangle(\phi \equiv a) \wedge \neg_\Pi \triangle(\phi \equiv b)\right) \to_L \left([a,b,\downarrow]\phi \equiv \left(\overline{\frac{b}{b-a}} \ominus \left(\phi \odot \overline{\frac{1}{b-a}}\right)\right)\right). \quad \text{(f)}$$

Notice that (a)–(f) actually formalise the increasing ('left shoulder') and decreasing ('right shoulder') fuzzy quantities.

The previously defined operators formalise fuzzy attribute values in FRDB. The only problem is that the domain of the attributes must be the unit interval. Thus, before interpreting a concrete fuzzy value it must be normalised. For example, consider the trapezoidal fuzzy number representing height $trap(180, 190, 10, 10)$. If we agree the maximum height is 250, then the connective interpreting this value is

$$\left[\frac{180-10}{250}, \frac{180}{250}, \frac{190}{250}, \frac{190+10}{250}\right],$$

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

that is, $[0.72, 0.76, 0.8, 0.84]$. Similarly, the increasing fuzzy quantity $fq[180, 190, inc]$ is interpreted as $[0.76, 0.8, \uparrow]$.

Recall that the SELECT operator has three main parts:

— the 'SELECT' part, which defines which attribute values are returned;
— the 'FROM' part, which defines which tables will be searched;
— the 'WHERE' part, which defines constraints.

The interpretation described above allows us to represent the 'where' part of the SELECT operator as an $\text{Ł}\Pi\frac{1}{2}$ formula.

5. FEQ operator

In order to formalise DB queries, we need to formalise the fuzzy counterpart of the relation $=$, that is, the fuzzy relation FEQ (fuzzy equals). First, we recall the definition of FEQ.

Definition 5.1. If A, B are fuzzy sets, then the fuzzy relation $FEQ(A, B)$ is defined by

$$FEQ(A, B) = \sup_{x \in X}\{\min(\mu_A(x), \mu_B(x))\}.$$

If, for example, set $A = \{m\}$ is a crisp set, we have

$$FEQ(A, B) = \mu_B(m).$$

On the other hand, let X be an infinite set, $m_0 \in X, \mu_A(m) = 1$ for all $m \in X, \mu_B(m_0) = 1$ and $\mu_B(m) = 0$ for all $m \in X$ different from m_0. Then

$$FEQ(A, B) = 1.$$

Arguably, if A is a crisp fuzzy number and B is a continuous one such that

$$\int_0^1 B(x)dx > 0,$$

then it would be quite natural to assume that $FEQ(A, B) = 0$. However, the present form of the FEQ operator is much more natural for pairs of continuous fuzzy numbers, and was constructed precisely for that purpose.

Generally, if $A = [a, b, c, d]$ and $B = [e, f, g, h]$ are trapezoidal fuzzy numbers, we have

$$FEQ([a, b, c, d], [e, f, g, h]) = \begin{cases} 0 & d \leqslant e \text{ or } h \geqslant a \\ 1 & \begin{cases} (b \leqslant f \leqslant c) \text{ or} \\ (b \leqslant g \leqslant c) \text{ or} \\ (f \leqslant b \wedge g \geqslant c) \end{cases} \\ \frac{df - ec}{(f - e - c + d)(c - d)} + \frac{d}{d - c} & f > c \text{ and } e < d \\ \frac{hb - ag}{(b - a - g + h)(g - h)} + \frac{h}{h - g} & b > g \text{ and } a < h \end{cases}$$

Similarly, we can define $FEQ([a, b, c, d], [e, f, \uparrow])$, $FEQ([a, b, \downarrow], [c, d, e, f])$, and so on. As before, in order to formalise the FEQ operator, we extend the language of $\text{Ł}\Pi\frac{1}{2}$ with countably many new constant symbols of the form $[a, b, c, d, e, f, g, h]$,

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

Table 1. *Example data.*

Name	*Salary*	*Age*	*Height*
John	2000	trap[25,27,29,32]	fq[160,170,dec]
Dave	trap[1000,1400,1800,2000]	trap[32,33,34,35]	tri[182,185,197]
Ann	fq[2000,2200,inc]	tri[18,21,24]	trap[156,158,161,162]
Michele	tri[1900,2200,2500]	trap[21,25,27,28]	fq[167,169,inc]

$[a,b,c,d,e,f,\uparrow]$, and so on, that correspond to all possible alterations of arguments in the FEQ operator, and add new axioms that formally capture the definition of FEQ.

(6) Here we will give the axioms for $FEQ([a,b,c,d],[e,f,g,h])$ only:

$$\left(\triangle(\overline{e}\to_L\overline{d})\vee\triangle(\overline{h}\to_L\overline{a})\right)\to_L\left([a,b,c,d,e,f,g,h]\equiv\overline{0}\right) \tag{a}$$

$$\begin{array}{l}((\triangle(\overline{f}\to_L\overline{b})\wedge\triangle(\overline{c}\to_L\overline{f}))\vee\\ (\triangle(\overline{g}\to_L\overline{b})\wedge\triangle(\overline{c}\to_L\overline{g}))\vee\\ (\triangle(\overline{b}\to_L\overline{f})\wedge\triangle(\overline{g}\to_L\overline{c}))\end{array}\to_L\left([a,b,c,d,e,f,g,h]\equiv\overline{1}\right) \tag{b}$$

$$(\triangle(\overline{c}\to_L\overline{f})\wedge\neg_\Pi\triangle(\overline{c}\equiv\overline{f})\wedge\triangle(\overline{e}\to_L\overline{d})\wedge\neg_\Pi\triangle(\overline{e}\equiv\overline{d}))\to_L\left([a,b,c,d,e,f,g,h]\equiv\overline{\frac{df-ec}{(f-e-c+d)(c-d)}+\frac{d}{d-c}}\right) \tag{c}$$

$$(\triangle(\overline{g}\to_L\overline{b})\wedge\neg_\Pi\triangle(\overline{g}\equiv\overline{b})\wedge\triangle(\overline{a}\to_L\overline{h})\wedge\neg_\Pi\triangle(\overline{a}\equiv\overline{h}))\to_L\left([a,b,c,d,e,f,g,h]\equiv\overline{\frac{hb-ag}{(b-a-g+h)(g-h)}+\frac{h}{h-g}}\right). \tag{d}$$

If one of the fuzzy sets is actually a crisp set, we define FEQ by $FEQ(c,B)=\mu_B(c)$.

Example 5.1. Suppose we have the data shown in Table 1 in the database. We will now show how a query is executed using PFSQL.

Say we want to find a person who has a salary of approximately 2100 euros and is either age 25–27 or above average height. Also, the salary is very important (priority 1), Age is the least important(priority 0.5) and height has priority 0.7. This translates to the following query.

```
SELECT  Salary, Age, Height FROM DB

WHERE (Salary=tri[1900,2100,2200] Priority 1) AND
(Age=trap[22,25,27,29] Priority 0.5 OR Height=fq[165,175,inc]
Priority 0.7)
```

The calculation will be done using the GPFCSP systems defined in Takači *et al.* (2008).

If we assume that the maximum salary is 10000, the maximum height is 230, and the maximum age is 100, the formalisation of attribute values for John is given in Table 2.

We will now explain how the WHERE line of a query is formalised and evaluated for John. Using ϕ to denote the formula formalising the WHERE line, ϕ is evaluated as

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

Table 2. *Formalised attribute values from table 1.*

Name	*Salary*	*Age*	*Height*
John	$\overline{0.2}$	[0.25, 0.27, 0.29, 0.32]	$[\frac{160}{230}, \frac{170}{230}, \uparrow]$
Dave	[0.10, 0.14, 0.18, 0.20]	[0.32, 0.33, 0.34, 0.35]	$[\frac{182}{230}, \frac{185}{230}, \frac{185}{230}, \frac{197}{230}]$
Ann	[0.20, 0.22, ↑]	[0.18, 0.21, 0.21, 0.24]	$[\frac{156}{230}, \frac{158}{230}, \frac{161}{230}, \frac{162}{230}]$
Michele	[0.19, 0.22, 0.22, 0.25]	[0.21, 0.25, 0.27, 0.28]	$[\frac{167}{230}, \frac{169}{230}, \uparrow]$

Table 3. *Query satisfaction degrees.*

Name	*Salary*	*Age*	*Height*	*QueSat*
John	0.5	1	0.25	0.5
Dave	0.25	0	1	0.25
Ann	1	0.333	0	0.9665
Michele	1	1	1	1

follows:

$$e(\phi) = e(\phi_1) \wedge_L (e(\phi_2) \vee_L e(\phi_3)).$$

The value $e(\phi_1)$ is then calculated as follows:

$$e(\phi_1) = e(\psi_1) \vee_{pr} e(\overline{1}).$$

We have that $\psi_1 \equiv [a, b, c, d]\varepsilon$ where $[a, b, c, d] = [0.19, 0.21, 0.21, 0.22]$ are the quotients of the triangular fuzzy number $tri[1900, 2100, 2200]$ and $\varepsilon = 0.2$. Thus, using axiom 1(c), we have

$$\psi_1 \equiv [0.19, 0.21, 0.21, 022]0.2 \equiv \left(\overline{0.2} \odot \overline{\left(\frac{1}{0.21 - 0.19}\right)}\right) \ominus \overline{\left(\frac{0.19}{0.21 - 0.19}\right)} \equiv \overline{0.5}$$

Trivially, we have $e(\psi_1) = e(\overline{0.5}) = 0.5$. Moreover,

$$e(\phi_1) = e(\psi_1) \vee_{pr} e(\overline{1}) = (0.5 \vee_L (1 - 1)) = 0.5.$$

Similarly, $e(\phi_2) = e(\psi_2) \vee_{pr} e(\overline{0.5})$.

We have that $\psi_2 \equiv [a, b, c, d, e, f, g, h]$ where $[a, b, c, d] = [0.22, 0.25, 0.27, 29]$ are the quotients of the trapezoidal fuzzy number $trap[2200, 2500, 2700, 2900]$ and $[e, f, g, h] = [0.25, 0.27, 0.29, 0.32]$ are the quotients of the trapezoidal fuzzy number that represents John's age. It is easy to see that $\psi_2 \equiv \overline{1}$ (axiom 3(b)), yielding

$$e(\phi_2) = e(\psi_2) \vee_{pr} e(\overline{0.5}) = e(\overline{1}) \vee_{pr} e(\overline{0.5}) = 1.$$

Similarly, $e(\phi_3) = e(\psi_3) \vee_{pr} e(\overline{0.8}) = e(\overline{0.25} \vee_{pr} e(\overline{0.8}) = 0.25$.

Finally, the satisfaction degree for John is obtained by

$$e(\phi) = e(\phi_1) \wedge_L (e(\phi_2) \vee_L e(\phi_3)) = 0.5 \wedge_L (1 \vee_L 0.25) = 0.5\,.$$

Table 3 gives the query satisfaction degrees for each data tuple.

https://doi.org/10.1017/S0960129511000673 Published online by Cambridge University Press

6. Conclusion

The main goal of this paper was the formalisation of PFSQL queries by obtaining their interpretation in some known fuzzy logic, and this has been achieved using $\text{Ł}\Pi\frac{1}{2}$ fuzzy logic. First we defined a conservative extension of the $\text{Ł}\Pi\frac{1}{2}$ logic that is powerful enough to interpret PFSQL queries. We then proved a theorem that illustrates the expressive power of the $\text{Ł}\Pi\frac{1}{2}$ logic and gives the PSPACE containment for the complexity of finding a model for a given $\text{Ł}\Pi\frac{1}{2}$ logic formula. Bearing in mind that we described a way to extend this logic to allow the PFSQL queries to be coded in it, the fact that it is PSPACE contained has an important impact on the performance of PFSQL query resolution.

Example 5.1 illustrates how the actual calculations are done. Future work includes the integration of this formalisation into the existing PFSQL interpreter. The idea is to process PFSQL queries by formalising them and putting the formalised query through an automated theorem prover to obtain results. As with any formalisation, there are potential benefits to be expected. First, it would be possible to prove the correctness of a query's resolution. Next, formalisation guarantees that all possible queries defined by it can be processed by the system, and without errors. Finally, potential benefits in the performance of query execution can be expected for some types of queries. Obviously, a performance analysis and a comparison of the results with the previous version of the PFSQL interpreter should also be carried out.

Acknowledgements

We would like to thank the editor and the anonymous referees for useful remarks and comments that have significantly enhanced the quality of this paper.

References

Buckles, B. and Petry, F. (1982) A Fuzzy representation of data for relational databases. *Fuzzy Sets and Systems* **7** (3) 213–226.

Chaudhry, N., Moyne, J. and Rundensteiner, E. (1994) A design methodology for databases with uncertain data. *Proceedings 7th International Conference on Scientific and Statistical Database Management* 32–41.

Chen, G. Q. and Kerre, E. E. (1998) Extending ER/EER concepts towards fuzzy conceptual data modelling. *Proceedings IEEE International Conference on Fuzzy Systems* 1320–1325.

Galindo, J., Medina, J. and Aranda, M. (1999) Querying fuzzy relational databases through fuzzy domain calculus. *International Journal of Intelligent Systems* **14** (4) 375–411.

Galindo, J., Medina, J., Cubero, J. and Garcia, M. (2001) Relaxing the universal quantifier of the division in fuzzy relational databases. *International Journal of Intelligent Systems* **16** (6) 713–742.

Galindo, J., Urrutia, A. and Piattini, M. (2006) Fuzzy databases: modelling design and implementation. IDEA Group.

Cintula, P., Hájek, P. and Horčik, R. (2007) Formal systems of fuzzy logic and their fragments. *Annals of Pure and Applied Logic* **150** 40–65.

Esteva, F., Godo, L., Hájek, P. and Navara, M. (2000) Residuated fuzzy logics with an involutive negation. *Archive for Mathematical Logic* **39** (2) 103–124.

Esteva, F., Godo, L. and Montagna, F. (2001) The ŁΠ and ŁΠ$\frac{1}{2}$ logics: two complete fuzzy logics joining Łukasiewicz and Product Logics. *Archive for Mathematical Logic* **40** (1) 39–67.

Godo, L., Esteva, F. and Hajek, P. (2000) Reasoning about probability using fuzzy logic. *Neural Network World* **10** (5) 811–824.

Godo, L. and Marchioni, E. (2006) Coherent conditional probability in a fuzzy logic setting. *Logic Journal of the IGPL* **14** (3) 457–481.

Hájek, P. (1998) *Metamathematics of fuzzy logic*, Kluwer academic publishers.

Hájek, P., Godo, L. and Esteva, F. (1995) Fuzzy logic and probability. *Proceedings 11th UAI* 237–244.

Hájek, P., Godo, L. and Esteva, F. (1996) A complete many-valued fuzzy logic with product conjunction. *Archive for Mathematical Logic* **35** (3) 191–208.

Hudec, M. (2009) An approach to fuzzy database querying, analysis and realization. *Computer Science and Information Systems* **6** (2) 127–140.

Luo, X., Lee, J., Leung, H. and Jennings, N. (2003) Prioritized fuzzy constraint satisfaction problems: axioms, instantiation and validation. *Fuzzy Sets and Systems* **136** (2) 151–188.

Marchioni, E. and Montagna, F. (2008) On triangular norms and uninorms definable in ŁΠ$\frac{1}{2}$. *International Journal of Approximate Reasoning* **47** (2) 179–201.

Medina, J. M., Pons, O. and Vila, M. A. (1994) GEFRED: a generalized model of fuzzy relational databases. *Information Sciences* **76** (1-2) 87–109.

Škrbić, S. and Takači, A. (2008) On development of fuzzy relational database applications. *Proceedings 12th IPMU* 268–273.

Škrbić, S. and Racković, M. (2009) PFSQL: a fuzzy SQL language with priorities. *Proceedings of PSU-UNS International Conference on Engineering Technologies, Novi Sad, Serbia* 58–63.

Škrbić, S., Racković, M. and Takači, A. (2011) Towards the methodology for development of fuzzy relational database applications. *Computer Science and Information Systems* **8** (1) 27–40.

Takači, A., Perović, A. and Jovanović, A. (2008) Measuring uncertainty with priority based logic. *Proceedings 12th IPMU* 1490–1496.

Takači, A. (2005) Schur-concave triangular norms: characterization and application in PFCSP. *Fuzzy Sets and Systems* **155** (1) 50–64.

Takači, A. and Škrbić, S. (2005) How to implement FSQL and priority queries. *Proceedings 3rd Serbian–Hungarian Joint Symposium on Intelligent Systems, Subotica, Serbia* 261–267.

Takači, A. and Škrbić, S. (2007) Measuring the similarity of different types of fuzzy sets in FRDB. *Proceedings EUSFLAT-LFA, Ostrava, Czech Republic* 247–252.

Takači, A. and Škrbić, S. (2008) Data model of FRDB with different data types and PFSQL. In: Galindo, J. (ed.) *Handbook of Research on Fuzzy Information Processing in Databases*, Information Science Reference 407–434.

Zvieli, A. and Chen, P. (1986) ER modelling and fuzzy databases. *Proceedings of the 2nd International Conference Data Engineering* 320–327.