Querying with Łukasiewicz logic

Stefano Aguzzoli
and Pietro Codara
Dipartimento di Informatica
Università degli Studi di Milano, Italy
{aguzzoli, codara}@di.unimi.it

Tommaso Flaminio
and Brunella Gerla
Dipartimento di Scienze Teoriche e Applicate
Università dell'Insubria,
Varese, Italy
{tommaso.flaminio, brunella.gerla}@uninsubria.it

Diego Valota
Artificial Intelligence Research Institute (IIIA)
CSIC, Spain
diego@iiia.csic.es

***Abstract*—In this paper we present, by way of case studies, a proof of concept, based on a prototype working on a automotive data set, aimed at showing the potential usefulness of using formulas of Łukasiewicz propositional logic to query databases in a fuzzy way. Our approach distinguishes itself for its stress on the purely linguistic, contraposed with numeric, formulations of queries. Our queries are expressed in the pure language of logic, and when we use (integer) numbers, these stand for shortenings of formulas on the syntactic level, and serve as linguistic hedges on the semantic one. Our case-study queries aim first at showing that each numeric-threshold fuzzy query is simulated by a Łukasiewicz formula. Then they focus on the expressing power of Łukasiewicz logic which easily allows for updating queries by clauses and for modifying them through a potentially infinite variety of linguistic hedges implemented with a uniform syntactic mechanism. Finally we shall hint how, already at propositional level, Łukasiewicz natural semantics enjoys a degree of reflection, allowing to write syntactically simple queries that semantically work as meta-queries weighing the contribution of simpler ones.**

I. Introduction, and motivation

The aim of this paper is to give a rather informal presentation of a natural semantics for Łukasiewicz logic (contraposed with the formal [0,1]-valued semantics) where formulas are interpreted as fuzzy queries to a database.

In this framework, database entries (the rows of a table) are identified with truth-value assignments, or possible worlds, and the evaluation of a query is simply the truth-value of the formula encoding the query in the considered possible worlds.

The negation connective plays the rôle of asking for the opposite quality to the one being negated. Lattice connectives behave much like their Boolean counterparts, creating unions and intersections of answer sets. Monoidal, non-idempotent connectives, which characterise Łukasiewicz logic, are used both to implement a mechanism to formulate an infinite variety of linguistic hedges, and to act as a comparison and weighing operator between simpler queries.

We apply these notions to a prototype system that allows to query a database of cars via formulæ of pure propositional Łukasiewicz logic. We analyse some examples to show how users can benefit from the flexibility and expressive power of this language.

We remark that, at least in principle, our queries are purely linguistic objects, translatable in natural language, where the linguistic hedges are translated as applications (maybe iterated) of *somewhat* and *very*, while the monoidal connective $\ominus$ corresponds to a query asking for objects (cars in our example application) that satisfy some given properties *much more* than other ones. We strike a comparison of our purely linguistic queries with numeric-threshold based queries, and show how the latter are very easily replicated in our chosen language.

The paper is organised as follows. Section II is a brief introduction to infinite-valued propositional Łukasiewicz logic, with statement of the most relevant results which are at the base of our proposed natural semantics for Łukasiewicz logic based queries.

Section III shortly describes our prototype implementation to query a database of cars.

Section IV analyses several queries to illustrate why we think that querying a database with formulas of Łukasiewicz may be useful, and reflect the proposed natural semantics.

II. Łukasiewicz logic in brief

Łukasiewicz (*infinite-valued propositional*) *logic* is a non-classical many-valued system going back to the 1920's, cf. the early survey [1, §3], and its annotated English translation in [2, pp. 38–59]. The standard modern reference for Łukasiewicz logic is [3], while [4] deals with topics at the frontier of current research. Łukasiewicz logic can also be regarded as a member of a larger hierarchy of many-valued logics that was systematised by Petr Hájek in the late Nineties, cf. [5], and later extended by Esteva and Godo in [6]; see also [7], [8]. Let us recall some basic notions.

Let us fix once and for all the countably infinite set of propositional variables:

$$\textsc{Var} = \{X_1, X_2, \ldots, X_n, \ldots\}.$$

Let us write $\perp$ for the logical constant *falsum*, $\neg$ for the unary negation connective, and $\to$ for the binary implication connective. The set FORM of (well-formed) formulæ is defined exactly as in classical logic over the language $\{\bot, \neg, \to\}$. Derived connectives $\top, \vee, \wedge, \leftrightarrow, \oplus, \odot, \ominus$ are defined in the following table, for every formula α and β:

Derived connective	**Definition**
$\top$	$\neg\bot$
$\alpha \vee \beta$	$(\alpha \to \beta) \to \beta$
$\alpha \wedge \beta$	$\neg(\neg\alpha \vee \neg\beta)$
$\alpha \leftrightarrow \beta$	$(\alpha \to \beta) \wedge (\beta \to \alpha)$
$\alpha \oplus \beta$	$\neg\alpha \to \beta$
$\alpha \odot \beta$	$\neg(\neg\alpha \oplus \neg\beta)$
$\alpha \ominus \beta$	$\neg(\alpha \to \beta)$

TABLE I
DERIVED CONNECTIVES IN ŁUKASIEWICZ LOGIC.

Let us present the $[0,1]$-valued semantics of Łukasiewicz logic. An *atomic assignment*, or *atomic evaluation*, is an arbitrary function $\overline{w}\colon \mathrm{VAR} \to [0,1]$. Such an atomic evaluation is uniquely extended to an *evaluation* of all formulæ, or *possible world*, *i.e.* to a function $w\colon \mathrm{FORM} \to [0,1]$ [1], via the compositional rules:

$$\begin{aligned} w(\bot) &= 0\,, \\ w(\alpha \to \beta) &= \min\{1, 1-(w(\alpha)-w(\beta))\}\,, \\ w(\neg\alpha) &= 1-w(\alpha)\,. \end{aligned}$$

It follows by trivial computations that the formal semantics of derived connectives is the one reported in Table II. *Tautologies* are defined as those formulæ that evaluate to 1 under every evaluation.

Notation	**Formal semantics**
$\bot$	$w(\bot) = 0$
$\top$	$w(\top) = 1$
$\neg\alpha$	$w(\neg\alpha) = 1 - w(\alpha)$
$\alpha \to \beta$	$w(\alpha \to \beta) = \min\{1, 1-(w(\alpha)-w(\beta))\}$
$\alpha \vee \beta$	$w(\alpha \vee \beta) = \max\{w(\alpha), w(\beta)\}$
$\alpha \wedge \beta$	$w(\alpha \wedge \beta) = \min\{w(\alpha), w(\beta)\}$
$\alpha \leftrightarrow \beta$	$w(\alpha \leftrightarrow \beta) = 1 - \lvert w(\alpha) - w(\beta)\rvert$
$\alpha \oplus \beta$	$w(\alpha \oplus \beta) = \min\{1, w(\alpha) + w(\beta)\}$
$\alpha \odot \beta$	$w(\alpha \odot \beta) = \max\{0, w(\alpha) + w(\beta) - 1\}$
$\alpha \ominus \beta$	$w(\alpha \ominus \beta) = \max\{0, w(\alpha) - w(\beta)\}$

TABLE II
FORMAL SEMANTICS OF CONNECTIVES IN ŁUKASIEWICZ LOGIC.

Each propositional formula φ whose occurring variables are in $\{X_1, X_2, \ldots, X_n\}$ uniquely determines a term function

$$\bar{\varphi}\colon [0,1]^n \to [0,1]\,,$$

by the following inductive prescription, for every $(t_1, \ldots, t_n) \in [0,1]^n$.

1) If $\varphi = X_i$ then $\bar{\varphi}(t_1, \ldots, t_n) = t_i$.
2) If $\varphi = \bot$ then $\bar{\varphi}(t_1, \ldots, t_n) = 0$.
3) If $\varphi = \neg\psi$ then $\bar{\varphi}(t_1, \ldots, t_n) = 1 - \bar{\psi}(t_1, \ldots, t_n)$.
4) If $\varphi = \psi \to \vartheta$ then $\bar{\varphi}(t_1, \ldots, t_n) = \min\{1, 1 - (\psi(t_1, \ldots, t_n) - \vartheta(t_1, \ldots, t_n))\}$.

This definition implies that $w(\varphi) = \bar{\varphi}(\bar{w}(X_1), \ldots, \bar{w}(X_n))$ for all possible worlds w.

McNaughton's Representation Theorem [9] states that the class of n-variable term functions $\bar{\varphi}\colon [0,1]^n \to [0,1]$ coincides with the class of all functions $f\colon [0,1]^n \to [0,1]$ that are continuous (in the standard Euclidean topology), and piecewise linear with integer coefficients, that is, there exist finitely many linear polynomials $p_1, p_2, \ldots, p_u\colon [0,1]^n \to [0,1]$ such that each p_i has the form $p_i(t_1, \ldots, t_n) = b_i + \sum_{j=1}^{n} a_{i,j} t_i$ for all coefficients $a_{i,j}$ and b_i being integers, and a function $\iota\colon [0,1]^n \to \{1, 2, \ldots, u\}$ such that

$$f(t_1, \ldots, t_n) = p_{\iota(t_1, \ldots, t_n)}(t_1, \ldots, t_n)\,,$$

for all $(t_1, \ldots, t_n) \in [0,1]^n$.

Menu-Pavelka's Theorem [10] states that Łukasiewicz logic is characterised in the Hájek's hierarchy BL of Basic Fuzzy Logics, or in the even larger Esteva and Godo's hierarchy MTL [6] of Monoidal t-norm-based Logics, as the unique logic having continuous term functions. This fact, together with involutiveness of negation, that is

$$w(\neg\neg\varphi) = w(\varphi)$$

for all possible worlds w, and the simultaneous failure of idempotency for the monoidal connectives $\oplus$ and $\odot$, not to mention the deep connection with lattice-ordered abelian groups [3], which allows to model real arithmetic on the unit interval, renders Łukasiewicz logic a very interesting tool to implement fuzzy-based applications.

In this paper we shall focus on some other properties of Łukasiewicz logic that further support the notion that this logic may constitute the ideal theoretical backbone to certain fuzzy-based applications.

Our first concern is actually rather philosophical and constitutes in our opinion a defensible rebuttal against the frequent attack to fuzzyness consisting in the observation that graded, or $[0,1]$ fuzzy truth-values are arbitrary or have no meaning at all. We counter this statement considering maximally consistent theories.

A theory Θ in Łukasiewicz logic is a deductively closed set of formulas, that is, if a formula φ is such that $w(\varphi) = 1$ for all possible worlds w such that $w(\vartheta) = 1$ for all $\vartheta \in \Theta$, then $\varphi \in \Theta$, too.

A theory Θ is maximally consistent if it cannot be enlarged without losing consistency, that is, $w(\varphi) < 1$ for all $\varphi \notin \Theta$ and for all possible worlds w such that $w(\vartheta) = 1$.

A set of postulates for a theory Θ is a set of formulas $\Gamma \subseteq \Theta$ such that Θ is the deductive closure of Γ, that is, Θ contains exactly all formulas ϑ such that $w(\vartheta) = 1$ for all possible worlds w for which $w(\gamma) = 1$ for all $\gamma \in \Gamma$.

In [11] Marra points out that the set of maximally consistent theories written in the variables $\{X_1, X_2, \ldots, X_n\}$ corresponds bijectively to the set of all n-tuples $[0,1]^n$. Moreover, each $\bar{t} = (t_1, \ldots, t_n) \in ([0,1] \cap \mathbb{Q})^n$ corresponds to a theory

[1] Since we are working with a purely truth-functional propositional logic we safely consider *evaluation* and *possible world* as synonymous.

$\Theta_{\bar{t}}$ having a set of postulates $\Gamma_{\bar{t}} = \{\gamma_{\bar{t}}\}$ for a suitable formula $\gamma_{\bar{t}}$.

When we consider formulas in just one variable, the above bijection tells us that each truth-value $\delta \in [0,1]$ corresponds, in the formal semantics of Łukasiewicz logic, exactly to one maximally consistent theory. That is, the choice of a value in $[0,1]$ is canonically and consistently reflected in the choice of a maximally consistent theory. So, any truth value has a canonical and fixed semantics, formed by the formulas in the corresponding theory. *Vice versa*, each truth-value is described linguistically by a set of formulas. We refer the reader to [11] for a thorough treatment of this topic. In this paper we are specially interested in some viable pragmatic consequences of this correspondence.

As a matter of fact, for each rational $\delta \in [0,1] \cap \mathbb{Q}$, the formula $\gamma_{\bar{\delta}}$ can be constructed as $\alpha \vee \beta$, where α and β are built from X_1 and $\neg X_1$ using only the *minus* connective $\ominus$.

As we shall see, the $\ominus$ connective will play a major rôle in the semantics we mean to use to query databases with Łukasiewicz logic.

A. An intended semantics for Łukasiewicz logic

We shall sketch in this section a natural semantics, which corresponds and interprets the formal semantics given in the previous section, and prepare the way to use it to express and interpret fuzzy queries to a database.

1) Possible Worlds, Queries and Answer Sets: Possible worlds $w\colon \text{FORM} \to [0,1]$ corresponds bijectively to atomic assignments $\bar{w}\colon \text{VAR} \to [0,1]$. Clearly the value $w(\varphi)$ depends only on the finitely many variables occurring in φ. We shall identify atomic assignments, and hence, possible worlds, with the fuzzy entries in the rows of a database table. More precisely, if a row r belongs to a table where all the columns with values ranging in $[0,1]$ are $c_1, c_2, \ldots, c_u$, then we consider $r\colon \{c_1, c_2, \ldots, c_u\} \to [0,1]$ as the atomic assignment corresponding to a possible world where to evaluate our queries, which will be formulas over the variables $\{c_1, c_2, \ldots, c_u\}$. It is then straightforward that the answer-set to a query φ over some table is formed by the rows r such that $r(\varphi) = 1$. Notice that in the next section for sake of simplicity we shall speak of answer set also when referring to (the top part of) the set of rows r ranked by the values $r(\varphi)$.

2) Variables, and the Negation Connective: But, what entries in the database table shall we consider as fuzzy? In our approach, we shall consider a column to be fuzzy valued if it corresponds to a graded (and $[0,1]$-normalised) property which has an opposite. For instance, *tall* has as its opposite *short*, with their obvious general meanings, while *red*, the property of being red, might not have a natural opposite (clearly, this depends on context, and we may always setup for an artificial opposite *non-red*). Of each pair of opposites (p,q) we clearly only need to store one value $r(p)$ for each row r, stating how much r is p. Clearly the degree of r being q shall be given by $\neg r(p)$.

3) Conjunction and Disjunction: One can form conjunctions and disjunctions of simpler queries just by using the lattice connectives. Recall that $r(\varphi \vee \psi) = \max\{r(\varphi), r(\psi)\}$ and $r(\varphi \wedge \psi) = \min\{r(\varphi), r(\psi)\}$. Observe that the resulting answer sets correspond, as in the crisp Boolean case, to unions and intersections of the simpler answer sets.

4) Basic Literals and Linguistic Hedges: One of the most interesting features of fuzzy querying is the capability of using linguistic hedges such as *somewhat* or *very*. Łukasiewicz logic offers a uniform and syntactically simple mechanisms to implement a collection of infinitely many such hedges, through the use of basic literals.

For each integer $k > 0$ and each formula φ let $1\varphi = \varphi$ and $(k+1)\varphi = \varphi \oplus k\varphi$. Analogously, let $\varphi^1 = \varphi$ and $\varphi^{k+1} = \varphi \odot \varphi^k$. Basic literals [12] are the class of formulas described inductively by the following.

1) Each variable X_i is a basic literal.
2) If φ is a basic literal then $k\varphi$ is a basic literal.
3) If φ is a basic literal then φ^k is a basic literal.

Notice that $w(2\varphi) = 1$ iff $w(\varphi) \geq 1/2$, whence we use 2φ to model *somewhat* φ. Analogously, $w(\varphi^2) = 0$ iff $w(\varphi) \leq 1/2$, and we use it to model *very* φ. Moreover, we can intensify our notion of being *somewhat* φ, by using $k\varphi$ for some $k > 2$. Analogous intensifications of *very* φ are possible, and moreover we can create more complex hedges by means of general basic literals. For instance $2(\varphi^2)$ may model being somewhat but not extremely φ (or, *somewhat very* φ, to play the game just syntactically).

Clearly, complex basic literals are very hard to translate in comprehensible linguistic expressions, but this is true also of natural language sentences built with many occurrences of words such as *somewhat* and *very*.

Being linguistic hedges, basic literals can be used to express threshold queries, and, as a matter of fact they can reproduce the answer set of any numeric-threshold query. Consider any such query $r(\varphi) \geq \delta$ for some $\delta \in [0,1]$ (clearly, in applications, δ is rational, but the theory of basic literals deals with irrational δ, too). As the database table contains finitely many rows, either the answer set does not discard any row, or there is a row r_1 such that $r_1(\varphi) = \max\{r(\varphi) < \delta\}$. In this case, pick any two rationals q_1, q_2 with $r_1(\varphi) \leq q_1 < q_2 \leq \delta$. As proved in [12] there is a basic literal ψ such that $r(\psi(\varphi)) = 1$ iff $r(\varphi) \geq q_2$, and $r(\psi(\varphi)) = 0$ iff $r(\varphi) \leq q_1$. This shows that any query $r(\varphi) \geq \delta$ can be reproduced. Analogously, any query $r(\varphi) \leq \delta$ can be reproduced, too.

5) The $\ominus$ Connective and Comparisons: Usually, the algebraic treatment of Łukasiewicz logic is conducted electing one of the non-idempotent operations of the formal semantics as primitive. Tipically the chosen one is $\oplus$, as in MV-algebras [3], or $\odot$ as in involutive BL-algebras [5], or $\to$ as in Wajsberg hoops [3]. We base our semantics on the $\ominus$ connective instead, for it affords us to model comparisons between queries.

Recall that for any possible world (or row of a database table) r, we have $r(\varphi \ominus \psi) = \max\{0, r(\varphi) - r(\psi)\}$, that is, the semantics of $\ominus$ is truncated difference.

The truncated difference $r(\varphi \ominus \psi)$ evaluates to 1 iff $r(\varphi) = 1$ and $r(\psi) = 0$, and goes linearly to 0, which is achieved when

$r(\varphi) = r(\psi)$. It is then very tempting to read $r(\varphi \ominus \psi)$ as

$$r \text{ is } \varphi \text{ much more than it is } \psi\,,$$

or

$$\text{in the world } r,\ \varphi \text{ holds much more than } \psi\,.$$

Notice the use of *much* since r will appear in the answer set of $\varphi \ominus \psi$ iff $r(\varphi) = 1$ and $r(\psi) = 0$. This causes no real problem, since we can get a milder version attenuating the impact of the hedge *much* by use of the hedge *somewhat*.

We can then use $\ominus$ to model queries that gauge the difference between the values of other queries. We recall here that the linguistic power of $\ominus$ afford a genuine interpretation of fuzzy truth-values, since, as pointed out in the previous section, each rational in $[0,1]$ encodes the meaning of a formula built from X_1 and $\neg X_1$ by means of a disjunction of two subformulas each one of them built only using, possibly iterated, occurences of $\ominus$. Each irrational is encoded by infinitely many formulas built in a similar fashion.

III. A System to Query a Database with Formulas of Łukasiewicz Logic

To empirically test the intended semantics of Łukasiewicz logic given in Section II-A, we have implemented a simple web interface that translates Łukasiewicz logic formulas into SQL statements. This web application has been developed following the standard programming pattern *Model-View-Controller* (MVC) [13]. To facilitate the development we have used the PHP Phalcon[2] programming framework on the server side, hence all the *controllers* in our programming paradigm are written in PHP language. On the other hand, on the client side of the application HTML and Javascript are the languages of election. To make *views* more user-friendly we have employed different libraries, such as JQuery, Bootstrap, MathJax and MathLex.

Finally, our *model* is a single database table where we have collected cars data. The table fields with associated datatypes are summarised in the first two columns of Table III. The database table contains 4684 records. We use MySQL as DBMS.

The web interface has two key pages: one allows to normalise the numerical fields of the table, and the second allows to write a logical formula and submit it as a query to the database.

The first page shows the maximum M and the minimum m values of each numerical field stored in the table, allowing the user to choose his own maximum $M_u \leq M$ and minimum $m_u \geq m$ to normalise every field according to his personal bias. For instance, the field *max speed* has a maximum value M equal to 350. One can choose to fix $M_u = 250$; in this way all cars with max speed ≥ 250 will be considered as the fastest cars stored in the database (they are fast in degree 1). The normalised values n_i are obtained by the standard linear formula $n_i = (v_i - m_u)/(M_u - m_u)$, where v_i is

[2]Phalcon is a PHP module developed in C: this makes Phalcon one of the fastest PHP web frameworks.

TABLE III

Field	Type	Associated Variable
id	int(10) unsigned	–
manufacturer	varchar(50)	–
model	varchar(50)	–
trim	varchar(200)	–
price	int(11)	X0
length	int(11)	X1
width	int(11)	X2
height	int(11)	X3
fuel tank	int(11)	X4
seating capacity	tinyint(4)	X5
car segment	varchar(50)	–
drive	varchar(50)	–
fuel	varchar(50)	–
cubic capacity - cc	int(11)	X6
horsepower	int(11)	X7
power	int(11)	X8
environmental classification	varchar(10)	–
co2 emission	int(11)	X9
gearbox	varchar(50)	–
max speed	smallint(6)	X10
acceleration 0/100	decimal(5,2)	X11
urban cycle consumption	decimal(5,2)	X12
extra-urban cycle consumption	decimal(5,2)	X13
combined cycle consumption	decimal(5,2)	X14

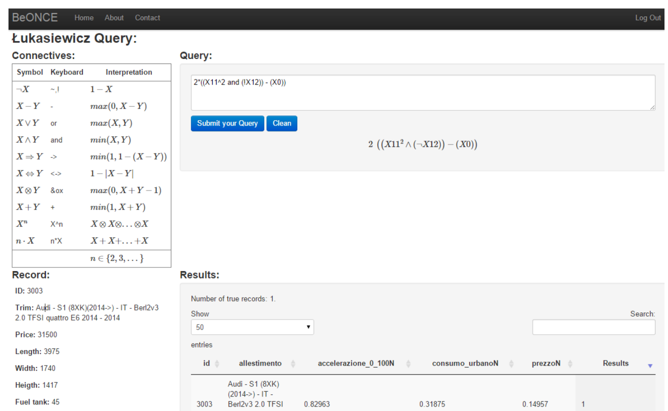


Fig. 1. The web page to submit queries

the value of the considered field at the i-th row. There are some numerical fields where the best result is not given by the maximum value but by the minimum (for istance *acceleration 0/100*). In such cases, by ticking a checkbox the user can reverse the normalisation of the given field. That is, $n_i = 1 - ((v_i - m_u)/(M_u - m_u))$.

The second page is the one where the database can be queried through the use of logical formulas. Therefore, in the page appears a list of variables `XN`, with `N`$\in \{0, \ldots, 14\}$. Every variable `XN` represents a precise field of the table. The third column of Table III gives the association between variables and fields. Then, the user is allowed to write a logical formula using such variables and the logical connectives described in Section II. The correspondence between connectives and *text strings* recognised by the web application is given in the first two columns of Table IV.

The formula has to be written in pure text, but thanks to MathJax the formula will be displayed nicely.

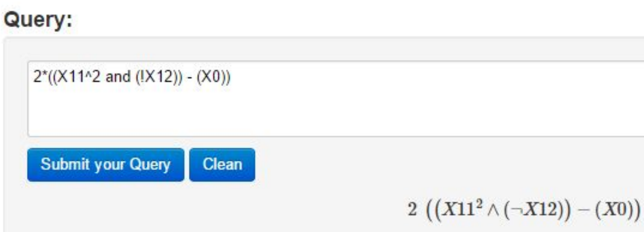


Fig. 2. Form for the insertion of queries

This formula is then parsed and translated in an SQL query according the association fields-variables given in Table III, and to the following table:

TABLE IV

Connective	Text strings	SQL statement
$\neg\alpha$	`!`	`1 - (`α`)`
$\alpha \rightarrow \beta$	`->`	`least(1,1-(`α`- `β`))`
$\alpha \vee \beta$	`or`	`greatest(`α`,`β`)`
$\alpha \wedge \beta$	`and`	`least(`α`,`β`)`
$\alpha \leftrightarrow \beta$	`<->`	`1-ABS(`α`-`β`)`
$\alpha \oplus \beta$	`+`	`least(1,`α`+`β`)`
$\alpha \odot \beta$	`ox`	`greatest(0,`α`+`β`-1)`
$\alpha \ominus \beta$	`-`	`greatest(0,`α`-`β`)`

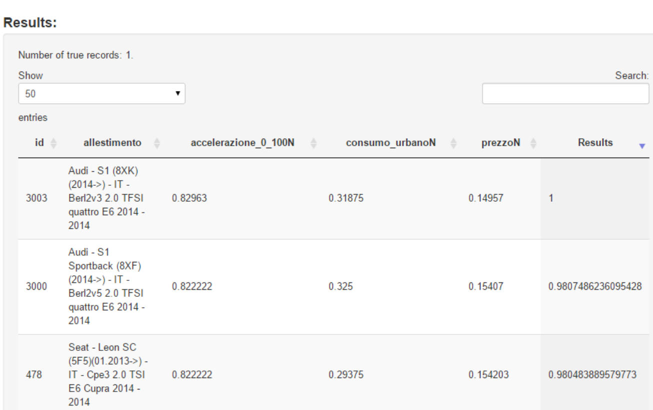
where α and β are formulas built from the variables `X0`,...,`X14`. For instance, the formula:

```
X1 and (X5 or !X7)
```

will be translated into the SQL query:

```
SELECT id, trim, length, seats, horsepower,
least(length,greatest(seats,horsepower))
As Results FROM auto;
```

The record set is then displayed as a table in the web page.


Results:

Number of true records: 1.

Show 50 entries Search:

id	allestimento	accelerazione_0_100N	consumo_urbanoN	prezzoN	Results
3003	Audi - S1 (8XK) (2014->) - IT - Berl2v3 2.0 TFSI quattro E6 2014 - 2014	0.82963	0.31875	0.14957	1
3000	Audi - S1 Sportback (8XF) (2014->) - IT - Berl2v5 2.0 TFSI quattro E6 2014 - 2014	0.822222	0.325	0.15407	0.9807486236095428
478	Seat - Leon SC (5F5)(01.2013->) - IT - Cpe3 2.0 TSI E6 Cupra 2014 - 2014	0.822222	0.29375	0.154203	0.980483889579773

Fig. 3. The record set

We observe that the iterated conjunction α`^n` is translated into the SQL expression `greatest(0,n*`α`-n+1)`. Similarly, the iterated disjunction `n`α is translated into `least(1,n*`α`)`. Whence, the translation of a Łukasiewicz fomula (written using connectives in Table IV and the iterated ones) into an SQL query takes only linear time.

IV. Case studies

A. *Rewriting a traditional query with Łukasiewicz logic*

Traditionally, an online database of cars is queried via the following steps. First, one select a (possibly empty) set of filters in order to restrict the search range. For example, one could choose to look only at diesel powered SUVs. After this preliminary step, one is asked to apply a conjunction of other filters, expressed in the form of constraints. For instance, one could ask that the price is between 30,000€ and 50,000€, *and* that the maximum speed is over 200 km/h. Two major Italian websites designed along these lines are Quattroruote[3] and Repubblica Motori[4].

Our first basic example aims to describe how to translate a classical query in our language, *i.e.* in a formula of Łukasiewicz logic. Suppose that our priority is an excellent acceleration, if possible accompanied with low urban consumption. In our normalised database, we may, for instance, query with the following instruction

```
(0.875<=X11) and (X12<=0.25)        (1)
```

Under a linear normalisation of the data to the unit interval $[0,1]$ this corresponds with cars whose acceleration 0-100 *km/h* is 5.10 *sec*, or better, and whose urban consumption is under 8 *l* per 100 *km*. If we query our database with (1), the answer set is

2969	Audi SQ5 (8R) 3.0 TDI DPF quattro 2012
4272	BMW Serie 3 GT 335d xDrive 2014
4275	BMW Serie 3 GT 335d xDrive 2014

The latter two cars are two different trim level of the same model of car. In this specific case, we can obtain exactly the same result using a query in Łukasiewicz logic. We first find a query capable of discard all non-valid cars:

```
X11^8 and (!X12)^4        (2)
```

In logical terms, the truth value of this formula when evaluated in a car which does not belong to our answer set is 0. Then, we need to ensure that the truth value of the formula is 1 when the formula is evaluated on a car belonging to the answer set. We obtained the desired result by the query:

```
20(X11^8 and (!X12)^4)        (3)
```

In general, is it always possible to rewrite a traditional query in the form `X`$\leq k$ or `X`$\geq k$, for $k \in [0,1]$ a rational value, in a Łukasiewicz query which provide exactly the same result. (For theoretical background on this, please refer to Section II-A4.) Nevertheless, this procedure produces very artificial values for exponents and multipliers, wiping out any possibility of exploiting in a totally natural way the expressive potential of the language of Łukasiewicz logic.

[3] `http://www.quattroruote.it/listino/`
[4] `http://listino-motori.repubblica.it/`

In this regard it is worth to observe that, when stepping from the query (2) to (3), we lose information. Indeed, the query (2) evaluated in the answer set is capable to rank the cars. In our case, we obtain that, though all three car are compatible with our queries, the BMWs better fit our desires. The answer set of (2), with the assigned truth valued in square brackets, is, indeed,

4272	BMW Serie 3 GT 335d xDrive 2014	[0.170]
4275	BMW Serie 3 GT 335d xDrive 2014	[0.170]
2969	Audi SQ5 (8R) 3.0 TDI DPF quattro 2012	[0.052]

A further consideration deserves to be done, concerning the query (2). Such query is obtained with the exact purpose of rewriting the traditional query (1). But, almost always, our starting point is the expression of our *desiderata* in natural language: *I want a car with remarkable acceleration, but with reduced urban fuel consumption*. The more natural way to query our database is thus to ask

$$\texttt{X11\^{}2 and (!X12)} \tag{4}$$

The first 10 records of the answer set of (4) are:

4272	BMW Serie 3 GT 335d xDrive 2014	[0.793]
4275	BMW Serie 3 GT 335d xDrive 2014	[0.793]
2969	Audi SQ5 (8R) 3.0 TDI DPF quattro 2012	[0.763]
4287	BMW Serie 6 Coupe 640d xDrive 2012	[0.748]
4369	BMW Serie 6 GC 640d xDrive 2013	[0.748]
4433	BMW X4 xDrive 3.5d 2014	[0.748]
4462	BMW Serie 4 Cabrio 435d xDrive 2014	[0.748]
4468	BMW Serie 4 Cabrio 435d xDrive 2014	[0.748]
665	Infiniti Q50 S Hybrid 2013	[0.737]
3028	Audi A7 Sportback 3.0 TDI quattro 2012	[0.733]

Not surprising, the first three cars in the (ranked) answer set of this query coincides with the answer set of the traditional query.

Another observation is in order here. We notice that the query (3) uses quite complex linguistic hedges. This may seem highly unnatural. On the other hand the numeric-threshold query (1) uses a three-digit precision. One may wonder whether and how this precision is needed, and if it really is, how it came about. One can imagine a process of trial and errors, where the trials `(0.87<=X11)` and `(0.88<=X11)` fail to produce any usable solution. Or one can imagine some algorithm that comes to produce this threshold via some sophisticated heuristics able to sift through to the third decimal before even submitting the query. In any case, whatever the process, it seems that if a precision of many decimals is really needed then the determination of the query requires a non-negligible additional computational effort, either performed by the human user, or by the machine. We do not want to embark in the task of measuring precisely this effort, but we only offer the remark that the growing complexity we see in the use of many decimals is reflected in the growing complexity of the linguistic hedges used in the Łukasiewicz query. One query should be deemed as natural as the other one. The fact that we have an infinite reservoir of distinct hedges then comes as a useful tool if we ever have to gauge complexity of threshold determination.

B. Exploiting the power of Łukasiewicz logic

Consider the query (4). Suppose we want to add more conditions to our request. Namely, suppose we do not want to spend so much money for our car, but we are looking, instead, for something *very cheap*. We can model our request by adding to (4) a new clause talking about the cost of the car, as follows.

$$\texttt{X11\^{}2 and (!X12) and (!X0)\^{}3} \tag{5}$$

The answer set seems to satisfy our request (the first 10 results are displayed):

4688	Ford Focus ST EcoBoost 250Cv 2012	[0.556]
3003	Audi S1 2.0 TFSI quattro 2014	[0.551]
2745	Volkswagen Golf VII 2.0 TSI GTI 2013	[0.546]
2205	Renault Megane III Coupe 2.0 TCe 265 2014	[0.544]
4781	Ford Focus Wagon ST 2012	[0.541]
3000	Audi S1 Sportback 2.0 TFSI quattro 2014	[0.538]
478	Seat Leon SC TSI Cupra 2014	[0.537]
3292	Opel Astra J GTC Turbo OPC 2012	[0.535]
496	Seat Leon 2.0 TSI Cupra 2014 [	0.530]
2736	Volkswagen Golf VII 2.0 TSI GTI 2013	[0.528]
52	Renault Clio IV 1.6 TCe 200 Monaco GP 2014	[0.526]

If we want, in addition, the cubic capacity to be *very small*, we can use the query

$$\texttt{X11\^{}2 and (!X12) and (!X0)\^{}3 and (!X6)\^{}2} \tag{6}$$

obtaining (the first 6 results are displayed):

52	Renault Clio IV 1.6 TCe 200 Monaco GP 2014	[0.526]
54	Renault Clio IV 1.6 TCe 200 R.S. 2013	[0.526]
2395	Renault Clio IV 1.6 TCe 200 R.S. 2013	[0.526]
3967	MINI CLUBMAN 2010	[0.511]
1196	MINI COUPE JCW 2011	[0.507]
448	Seat Ibiza SC 1.4 TSI SC Cupra 2013	[0.496]

After these simple examples, two main consideration are in order. First, queries written as formulæ in Łukasiewicz logic are easily updatable. In a traditional query like (2), based on the enforcement of crisp thresholds to the designated features, one usually have to reformulate the thresholds before stating some new conditions. This is especially true in our specific case, where the answer set contains as few as 3 cars, and there is no chance to regain new answers where asking for cars that are *also* cheap. The adaptive behavior of our queries is instead very natural. Of course, when we conjunct new conditions, the truth values of the top records of the answer set is decreasing. This is quite intuitive, since the more we are demanding, the less there is the possibility for our query to be fully satisfied by a car. We shall overcome this obstacle by asking for our final query to be *somewhat satisfied*. To do so, if α is the query, we simple need to write $n * \alpha$, with $n > 1$: the bigger is n, the more we relax our query. With this technique we can also get a query that is fully satisfied (*i.e.*, in degree 1) by a non-empty set of cars. For instance, the query

$$\texttt{2*(X11\^{}2 and (!X12) and (!X0)\^{}3 and (!X6)\^{}2)}$$

52	Renault Clio IV 1.6 TCe 200 Monaco GP 2014	[1]
54	Renault Clio IV 1.6 TCe 200 R.S. 2013	[1]
2395	Renault Clio IV 1.6 TCe 200 R.S. 2013	[1]
3967	MINI CLUBMAN 2010	[1]
1196	MINI COUPE JCW 2011	[1]
448	Seat Ibiza SC 1.4 TSI SC Cupra 2013	[0.993]

returns the following

The second consideration: it is easy and natural dealing with linguistic hedges. A linguistic hedge is a mitigating or intensifying word used to lessen or increase the impact of a sentence (see [14] for a fuzzy-set-theoretic interpretation of linguistic hedges). As described in the introduction, Łukasiewicz logic has the expressive power to deal with expressions like *very*, or *somewhat*. Consider the query (6). There, we are asking for a car with a *very* good acceleration, with low urban consumption, which is *very very* cheap, and with a *very* small cubic capacity. Suppose we want to lighten some requests. Suppose, for instance, that we can fit with a car with a good acceleration, which is *very* cheap, with a small cubic capacity and with urban consumption that are *somewhat* low. We query with

```
X11 and 2*(!X12) and (!X0)^2 and (!X6)
```
(7)

and obtain (the first 6 results are displayed):

52	Renault Clio IV 1.6 TCe 200 Monaco GP 2014	[0.763]
54	Renault Clio IV 1.6 TCe 200 R.S. 2013	[0.763]
2395	Renault Clio IV 1.6 TCe 200 R.S. 2013	[0.763]
448	Seat Ibiza SC 1.4 TSI SC Cupra 2013	[0.748]
4720	Ford Fiesta 1.6 EcoBoost ST 2013	[0.748]
1198	MINI COUPE S 2011	[0.744]

As expected, results are similar, though with higher values of truth degrees.

C. Towards a better understanding of the intended semantics of Łukasiewicz logic

We discuss here the rôle of the *minus* connective $\ominus$. Our starting point is the query (4). Let us state that (4) describe our notion of a *good* car. That is, a car is *good* if fully meets our demand, it is *poor* if it does not satisfy our request at all, it is *medium*, otherwise. Like in the preceding paragraph, we want to add a further request: that the car is economic. However, we aim to state something different from what is stated by the query (5). We do not want, in fact, to force the car to be cheap. We want, instead, to assert that we do not make any difference between a good car with a medium price, and a medium car with a low price. To make this request, we use the following query.

```
(X11^2 and (!X12)) - (X0)
```
(8)

According to the semantics presented in the introduction (and remembering that the left part of the expression represent our notion of a good car), we are asking for a car that is much more good than expensive. Thus, the perfect matching is given by a *top* car (as before, according to our notion) with a ridiculously low price. Most probably, such a car does not exist. As a matter of fact, the answer we obtain querying the database is the following.

3003	Audi S1 2.0 TFSI quattro 2014	[0.510]
3000	Audi S1 Sportback 2.0 TFSI quattro 2014	[0.490]
478	Seat Leon SC 2.0 TSI Cupra 2014	[0.490]
496	Seat Leon 2.0 TSI Cupra 2014	[0.488]
2988	Audi S3 2.0 TFSI quattro 2013	[0.481]
487	Seat Leon SC 2.0 TSI Cupra 2014	[0.480]
4275	BMW Serie 3 GT 335d xDrive 2014	[0.480]
497	Seat Leon 2.0 TSI Cupra 2014	[0.478]
2993	Audi S3 Sportback 2.0 TFSI quattro 2013	[0.476]
1595	BMW Serie 2 Coupe 228i 2014	[0.472]

None of the cars in our database fully fit the request. Though, as mention in the previous subsection, we could easily lighten the request to

```
2*((X11^2 and (!X12)) - (X0))
```

obtaining, as the only car satisfying (at 1) our formula:

3003	Audi S1 2.0 TFSI quattro 2014	[1]

We want to stress the difference between (8) and the following query, seemingly similar in meaning.

```
X11^2 and (!X12) and (!X0)
```
(9)

The top ranked cars according to the latter query are

4270	BMW Serie 3 GT 330d xDrive 2013	[0.711]
3685	Mercedes-Benz A 45 AMG 2012	[0.700]
665	Infiniti Q50 S Hybrid 2013	[0.698]

while 3003, the only car satisfying (8), ends up here in 20^{th} position. Comparing those entries we discover that while 4270 is better than 3003, the latter is distinguished by the following property: it maximises the difference between the truth value of the formula stating (in short) that *the car is good*, and the formula asserting that *the car is expensive*.

A more general example may clarify this point. Suppose you can describe, via a formula α, what (in your opinion) makes a car a good car. Further, suppose you can describe, via a formula β, what makes a car a bad car. Consider the formula $\gamma = \alpha \wedge \neg\beta$. Intuitively, γ describe your perfect car, with all pros and no cons. Let A be a car satisfying α in degree 0.5, and β in degree 0.5. Thus, A satisfy γ in degree 0.5.

Let now $\varphi = \alpha - \beta$. Remember that, in natural language, φ asks for a car which is *much more α than β*. Observe that the truth value of φ when evaluated in A is 0. While according to γ, the car A is a medium car, φ discard the same car. While γ simply measures the degree in which the car fits our collection (conjunction) of requests, the formula φ add a higher level to the query measuring the distance between the two simpler queries α and β. An assertion like φ can be, in fact, interpreted as a measure of the level of satisfaction of the buyer. In essence, the buyer get no satisfaction in buying whichever car with equivalent pros and cons. Satisfaction begin when pros overcome cons, and is maximal when we have all possible pros, and no cons. When dealing with queries constructed in the manner of φ, this is essentially how the logical connective $\ominus$ may serve us.

To stress this point, we provide another example. Assume the buyer asks for a car which is both γ and δ, say, "a powerful but economic car" (that is, "a powerful but not expensive car"). On further thinking the buyer makes the additional point that "in any case I prefer the car being economic rather than powerful", that can be modeled by $\gamma \ominus \neg\delta$ (or, equivalently, by $\delta \ominus \neg\gamma$), possibly attenuated by the use of a *somewhat* hedge. With this structure it appears clearly that use of $\ominus$ provides, semantically, a higher level in the language, since the clarification added by the buyer directly refers to the evaluation and weighing of the two simpler queries γ and δ. This cannot be achieved with the use of the other connectives $\neg, \vee, \wedge$ only.

V. Conclusion

The goal of this work is twofold. The case studies we proposed provide a better understanding of the intended semantics presented in the introduction. Indeed, in appreciating the answer given by our database system to our queries, we make clearer the natural linguistic meaning of a formula of Łukasiewicz logic. Moreover, we provide a further, pragmatic, justification of the proposed semantics.

With this work, we also aim to convince the reader that the pure Łukasiewicz logic can be effectively, and efficiently, used to treat real problems: in our case, to query an online database of cars. (Other, distinct, cases where Łukasiewicz logic is used in applications can be found in literature. See, for instance, [15], [16], where the use of Łukasiewicz implication in control systems is first theoretically discussed and then analyzed in an experiment on the technical analysis of the financial markets. Another example is [17], where a Łukasiewicz many-valued logic similarity based fuzzy control algorithm is introduced, and tested in three realistic traffic signal control systems. See also [18], [19].) Much work remains to be done on this front to develop an online system which is meant for a general user, typically not enough expert in logic to query with logical formulæ. First, we need to design the mechanisms of interaction between user and system (the way in which the user will form its query). This will probably be done with repeated questions to the user, in order to identify the most of her/his desires, her/his preferences, and the issues to which it is indifferent. Following this, the system must be equipped with an algorithm able to translate the user's desiderata in a Łukasiewicz formula. Only at this point, the system we developed and presented in this paper will go into action, providing an answer to the query.

Acknowledgment

We thank Vincenzo Marra for many useful discussions on the topics of this paper.

The authors were supported by the MIUR-FIRB research project *PNCE - Probability theory of non-classical events*. Valota acknowledges also partial support from a Marie Curie INdAM-COFUND Outgoing Fellowship. The research reported in this paper was carried out while Valota was a postdoc fellow of the Dipartimento di Scienze Teoriche e Applicate (Università dell'Insubria), supported by the PNCE research project.

References

[1] J. Łukasiewicz and A. Tarski, "Untersuchungen über den Aussagenkalkül," *C. R. Soc. Sc. Varsovie*, no. 23, pp. 30–50, 1930.

[2] A. Tarski, *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*. Oxford at the Clarendon Press, 1956, translated by J.H. Woodger.

[3] R. L. O. Cignoli, I. M. L. D'Ottaviano, and D. Mundici, *Algebraic foundations of many-valued reasoning*, ser. Trends in Logic—Studia Logica Library. Kluwer Academic Publishers, Dordrecht, 2000, vol. 7.

[4] D. Mundici, *Advanced Łukasiewicz calculus and MV-algebras*, ser. Trends in Logic—Studia Logica Library. Springer, Dordrecht, 2011, vol. 35.

[5] P. Hájek, *Metamathematics of fuzzy logic*, ser. Trends in Logic—Studia Logica Library. Kluwer Academic Publishers, Dordrecht, 1998, vol. 4.

[6] F. Esteva and L. Godo, "Monoidal t-norm based logic: towards a logic for left-continuous t-norms," *Fuzzy Sets and Systems*, vol. 124, no. 3, pp. 271–288, 2001.

[7] P. Cintula, P. Hájek, and C. Noguera, Eds., *Handbook of mathematical fuzzy logic. Volume 1*, ser. Studies in Logic (London). College Publications, London, 2011, vol. 37.

[8] ——, *Handbook of mathematical fuzzy logic. Volume 2*, ser. Studies in Logic (London). College Publications, London, 2011, vol. 38.

[9] R. McNaughton, "A theorem about infinite-valued sentential logic," *J. Symbolic Logic*, vol. 16, pp. 1–13, 1951.

[10] J. Menu and J. Pavelka, "On the poset of tensor products on the unit interval," *Comment. Math. Univ. Carolinae*, vol. 18, no. 2, pp. 329–341, 1977.

[11] V. Marra, "The Problem of Artificial Precision in Theories of Vagueness: A Note on the Rôle of Maximal Consistency," *Erkenntnis*, vol. 79, no. 5, pp. 1015–1026, 2014.

[12] S. Aguzzoli, "A note on the representation of McNaughton lines by basic literals," *Soft Computing*, vol. 2, no. 3, pp. 111–115, 1998.

[13] A. Leff and J. T. Rayfield, "Web-application development using the model/view/controller design pattern," in *5th International Enterprise Distributed Object Computing Conference (EDOC 2001), Proceedings*, 2001, pp. 118–127.

[14] L. A. Zadeh, "A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges," *Journal of Cybernetics*, vol. 2, no. 3, pp. 4–34, 1972.

[15] S. Bova, P. Codara, D. Maccari, and V. Marra, "A Logical Analysis of Mamdani-type Fuzzy Inference, I. Theoretical Bases," in *2010 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010*. IEEE, 2010, pp. 254–261.

[16] ——, "A Logical Analysis of Mamdani-type Fuzzy Inference, II. An Experiment on the Technical Analysis of Financial Markets," in *2010 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010*. IEEE, 2010, pp. 262–269.

[17] J. Niittymäki and E. Turunen, "Traffic signal control on similarity logic reasoning," *Fuzzy Sets and Systems*, vol. 133, no. 1, pp. 109–131, 2002.

[18] A. Casali, L. Godo, and C. Sierra, "A graded BDI agent model to represent and reason about preferences," *Artificial Intelligence*, vol. 175, no. 7-8, pp. 1468 – 1478, 2011.

[19] A. D. Nola and C. Russo, "Łukasiewicz transform and its application to compression and reconstruction of digital images," *Information Sciences*, vol. 177, no. 6, pp. 1481 – 1498, 2007.