Querying with Łukasiewicz logic

Pietro Codara

Dipartimento di Informatica, Università degli Studi di Milano

(Joint work with S. Aguzzoli, T. Flaminio, B. Gerla, and D. Valota)

FUZZ-IEEE 2015, Istanbul — Aug 4th, 2015

Overview

The Idea

Test an intended semantics for Łukasiewicz logic in a real-world situation.

Overview

The Idea

Test an intended semantics for Łukasiewicz logic in a real-world situation.

The Application

An automotive data-set. We query the database via the pure language of the logic, *i.e.*, via Łukasiewicz formulæ.

Overview

The Idea

Test an intended semantics for Łukasiewicz logic in a real-world situation.

The Application

An automotive data-set. We query the database via the pure language of the logic, *i.e.*, via Łukasiewicz formulæ.

The Question

What can we ask to our database?

Łukasiewicz logic in brief (1)

Łukasiewicz logic can be semantically defined as a many-valued logic, as follows.

Łukasiewicz logic in brief (1)

Łukasiewicz logic can be semantically defined as a many-valued logic, as follows.
Let Form be the set of formulæ over propositional variables $X_1, X_2, \ldots$ in the language $\to, \neg, \bot$.

Łukasiewicz logic in brief (1)

Łukasiewicz logic can be semantically defined as a many-valued logic, as follows.

Let $\textsc{Form}$ be the set of formulæ over propositional variables $X_1, X_2, \ldots$ in the language $\rightarrow, \neg, \perp$.

An assignment is a function $\mu\colon \textsc{Form} \rightarrow [0,1] \subseteq \mathbb{R}$ with values in the real unit interval such that, for any two $\alpha,\ \beta \in \textsc{Form}$,

$$\mu(\alpha \rightarrow \beta) = \min\{1, 1 - (\mu(\alpha) - \mu(\beta))\}$$

$$\mu(\neg\alpha) = 1 - \mu(\alpha)$$

$$\mu(\perp) = 0$$

Łukasiewicz logic in brief (1)

Łukasiewicz logic can be semantically defined as a many-valued logic, as follows.

Let $\textsc{Form}$ be the set of formulæ over propositional variables $X_1, X_2, \ldots$ in the language $\to, \neg, \bot$.

An assignment is a function $\mu \colon \textsc{Form} \to [0,1] \subseteq \mathbb{R}$ with values in the real unit interval such that, for any two $\alpha,\ \beta \in \textsc{Form}$,

$$\mu(\alpha \to \beta) = \min\{1, 1 - (\mu(\alpha) - \mu(\beta))\}$$

$$\mu(\neg\alpha) = 1 - \mu(\alpha)$$

$$\mu(\bot) = 0$$

A tautology is a formula α such that $\mu(\alpha) = 1$ for every assignment μ.

Łukasiewicz logic in brief (2)

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 : Derived connectives in Łukasiewicz logic.

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 $\varphi^{k+1} = \varphi \odot \varphi^k$.

The logic of vague proposition (1)

- Recent results show that Łukasiewicz logic is the logic suitable for dealing with a certain kind of vagueness – essentially the logic of "the more", "the less", and "the much more than".

The logic of vague proposition (1)

- Recent results show that Łukasiewicz logic is the logic suitable for dealing with a certain kind of vagueness – essentially the logic of "the more", "the less", and "the much more than".
- Łukasiewicz logic can deal with properties which has a natural opposite. For instance, fast has as its opposite slow, with their obvious general meanings, while *red*, the property of being red, might not have a natural opposite.

The logic of vague proposition (2)

Consider the propositions P : "The car is fast", and Q : "The car is cheap". Then,

The logic of vague proposition (2)

Consider the propositions P : "The car is fast", and Q : "The car is cheap". Then,

1. $\neg P$ means "The car is slow";

The logic of vague proposition (2)

Consider the propositions P : "The car is fast", and Q : "The car is cheap". Then,

1. $\neg P$ means "The car is slow";
2. $P \wedge Q$ means "The car is fast and cheap";

The logic of vague proposition (2)

Consider the propositions P : "The car is fast", and Q : "The car is cheap". Then,

1. $\neg P$ means "The car is slow";
2. $P \wedge Q$ means "The car is fast and cheap";
3. $P \vee Q$ means "The car is fast or cheap";

The logic of vague proposition (2)

Consider the propositions P : "The car is fast", and Q : "The car is cheap". Then,

1. $\neg P$ means "The car is slow";
2. $P \wedge Q$ means "The car is fast and cheap";
3. $P \vee Q$ means "The car is fast or cheap";
4. $P \odot P$ means "The car is very fast", $P \odot P \odot P = P^3$ means "The car is very very fast", $\ldots$;

The logic of vague proposition (2)

Consider the propositions P : "The car is fast", and Q : "The car is cheap". Then,

1. $\neg P$ means "The car is slow";
2. $P \wedge Q$ means "The car is fast and cheap";
3. $P \vee Q$ means "The car is fast or cheap";
4. $P \odot P$ means "The car is very fast", $P \odot P \odot P = P^3$ means "The car is very very fast", $\ldots$;
5. $P \oplus P$ means "The car is somewhat fast", $\ldots$;

The logic of vague proposition (2)

Consider the propositions P : "The car is fast", and Q : "The car is cheap". Then,

1. $\neg P$ means "The car is slow";
2. $P \wedge Q$ means "The car is fast and cheap";
3. $P \vee Q$ means "The car is fast or cheap";
4. $P \odot P$ means "The car is very fast", $P \odot P \odot P = P^3$ means "The car is very very fast", $\ldots$;
5. $P \oplus P$ means "The car is somewhat fast", $\ldots$;
6. $P \ominus Q$ means "The car is much more fast than cheap".

The database: data

Our model is a single database table where we have collected cars data. The table contains 4684 records, representing (a subset of) the cars on the Italian market in the year 2014.

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

The database: interface

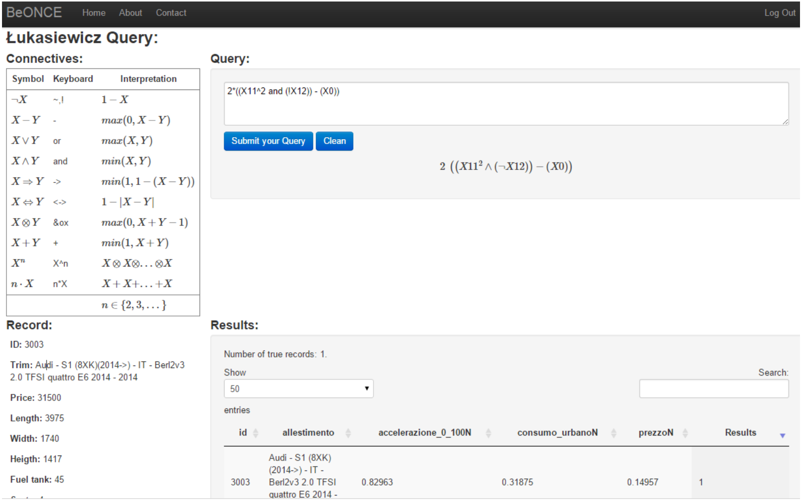

Figure : The web page to submit queries

Using linguistic hedges: an example

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.*

Using linguistic hedges: an example

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

$$X_{11}^2 \wedge \neg X_{12}$$

Using linguistic hedges: an example

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

$$X_{11}^2 \wedge \neg X_{12}$$

The first 10 records of the answer set 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]

Using linguistic hedges: updating queries

Suppose we do not want to spend so much money for our car, but we are looking, instead, for something very, very, (very) cheap.

Using linguistic hedges: updating queries

Suppose we do not want to spend so much money for our car, but we are looking, instead, for something very, very, (very) cheap.
We add a new clause talking about the cost of the car:

$$X_{11}^2 \wedge \neg X_{12} \wedge \neg X_0^3$$

Using linguistic hedges: updating queries

Suppose we do not want to spend so much money for our car, but we are looking, instead, for something very, very, (very) cheap.

We add a new clause talking about the cost of the car:

$$X_{11}^{2} \wedge \neg X_{12} \wedge \neg X_{0}^{3}$$

The answer set seems to satisfy our request:

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]

Using linguistic hedges: comments

- One of the most interesting features of Łukasiewicz querying is the capability of using linguistic hedges such as somewhat or very.

Using linguistic hedges: comments

- One of the most interesting features of Łukasiewicz querying is the capability of using linguistic hedges such as somewhat or very.
- Queries written as formulæ in Ł are easily updatable.

Using linguistic hedges: comments

- One of the most interesting features of Łukasiewicz querying is the capability of using linguistic hedges such as somewhat or very.
- Queries written as formulæ in Ł are easily updatable.
- When we conjunct new conditions, the truth values of the top records of the answer set is decreasing. We shall overcome this obstacle by asking for our final query to be somewhat satisfied. For instance, we can rewrite $X_{11}^2 \wedge \neg X_{12} \wedge \neg X_0^3$ as $2(X_{11}^2 \wedge \neg X_{12} \wedge \neg X_0^3)$.

Using linguistic hedges: comments

- One of the most interesting features of Łukasiewicz querying is the capability of using linguistic hedges such as somewhat or very.
- Queries written as formulæ in Ł are easily updatable.
- When we conjunct new conditions, the truth values of the top records of the answer set is decreasing. We shall overcome this obstacle by asking for our final query to be somewhat satisfied. For instance, we can rewrite $X_{11}^2 \wedge \neg X_{12} \wedge \neg X_0^3$ as $2(X_{11}^2 \wedge \neg X_{12} \wedge \neg X_0^3)$.
- It is 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. Nevertheless, this procedure produces very artificial values for exponents and multipliers. For example, $X_{11} \geq 0.875$, and $X_{12} \leq 0.25$ becomes $20(X_{11}^8 \wedge \neg X_{12}^4)$.

The minus connective $\ominus$

- 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.

The minus connective $\ominus$

- 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.

The minus connective $\ominus$

- 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 $\varphi = \alpha \ominus \beta$. In natural language, φ asks for a car which is much more α than β. 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.

The minus connective $\ominus$

- 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 $\varphi = \alpha \ominus \beta$. In natural language, φ asks for a car which is much more α than β. 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.
- An assertion like φ can be interpreted as a measure of the level of satisfaction of the buyer: satisfaction begin when pros overcome cons, and is maximal when we have all possible pros, and no cons.

The minus connective $\ominus$: an example

- The buyer, asked for a definition of a *good car*: "A good car has a *remarkable* acceleration, but a *reduced* urban fuel consumption". In symbols: $X_{11}^2 \wedge \neg X_{12}$.

The minus connective $\ominus$: an example

- The buyer, asked for a definition of a *good car*: "A good car has a *remarkable* acceleration, but a *reduced* urban fuel consumption". In symbols: $X_{11}^2 \wedge \neg X_{12}$.
- The buyer, asked for a definition of a *bad car*: "A bad car is an expensive car". In symbols: X_0.

The minus connective $\ominus$: an example

- The buyer, asked for a definition of a *good car*: "A good car has a *remarkable* acceleration, but a *reduced* urban fuel consumption". In symbols: $X_{11}^2 \wedge \neg X_{12}$.
- The buyer, asked for a definition of a *bad car*: "A bad car is an expensive car". In symbols: X_0.
- In the preceding example, we have *forced* a car to be cheap, using $\wedge$. 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. We are asking for a car that is much more good than bad.

The minus connective $\ominus$: an example, cont.

To make such request, we use the following query.

$$X^2_{11} \wedge \neg X_{12} \ominus \neg X^3_0$$

The minus connective $\ominus$: an example, cont.

To make such request, we use the following query.

$$X_{11}^{2} \wedge \neg X_{12} \ominus \neg X_{0}^{3}$$

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]

Conclusion, and further work

- The pure Łukasiewicz logic can be effectively, and efficiently, used to treat real problems: in our case, to query an online database of cars.

Conclusion, and further work

- The pure Łukasiewicz logic can be effectively, and efficiently, used to treat real problems: in our case, to query an online database of cars.
- 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.

Conclusion, and further work

- The pure Łukasiewicz logic can be effectively, and efficiently, used to treat real problems: in our case, to query an online database of cars.
- 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.
- To this end, we need to design the mechanisms of interaction between user and system, and

Conclusion, and further work

- The pure Łukasiewicz logic can be effectively, and efficiently, used to treat real problems: in our case, to query an online database of cars.
- 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.
- To this end, we need to design the mechanisms of interaction between user and system, and
- ...the system must be equipped with an algorithm able to translate the user's desiderata in a Łukasiewicz formula.

Thank you for your attention.