A General Fuzzy Logic Using Intervals

Benedek Nagy

Institute of Informatics, University of Debrecen, H-4010 PO Box 12, Hungary,
Research Group on Mathematical Linguistics, Rovira i Virgili University,
Tarragona, Spain
nbenedek@inf.unideb.hu

Abstract: In this paper we introduce an interval-valued logic system in which the truth values are intervals or sets of disjoint intervals over [0,1]. We extend the Boolean operators to these values in a natural way. Non-logical operators are also defined. The simulation of many-valued and fuzzy logic systems (such as Heyting's logic, Belnap's logic, Post's logic, Gödelian logic, Łukasiewicz-type logics and product logic) is presented. Some properties of the interval-valued general fuzzy logic are also presented.

Keywords: interval-valued logic, many-valued logic, fuzzy logic, general fuzzy logic

1 Introduction

The fuzzy systems are applied in several places, not only in Computer Science, but in industry and economy as well. One can choose among several fuzzy systems. In this paper we recall the most important many-valued and fuzzy logics. The three mostly used ones are the Gödelian, the product and the Łukasiewicz-type logics. They represent optimist (maximal profit, cooperation), realistic (moderate outcome, tolerance) and pessimist (minimal lost, competition) point of view, respectively [10]. There are classical logical tautologies which are not laws in some fuzzy systems. We present a general logical system with interval-values which can simulate the other systems. Moreover in the Interval-valued logic exactly the classical tautologies are the logical laws. The non-logical operators of the system give a flexibility to interpret logical systems having numbers as truth-values.

We will use the term truth-value, but in many cases when one uses a fuzzy system it is something else; for example, in many economic systems it is the notion of satisfiability etc., but it works as truth-values theoretically. We will use the letters A, B to refer both for logical formulae and their truth-values.

2 Basic Many-Valued and Fuzzy Logic Systems

In this section we recall several many-valued and fuzzy logic systems. The domain and the defined operations give the semantic of the system. The logics presented in Sections 2.1, 2.2 and 2.3 have only finite domains. The Gödelian and the Łukasiewicz-type systems work on finite and infinite domains as well, while the product logic has only infinite-valued variations.

2.1 Heyting's Three-Valued Logic

The first many valued system which tried to follow the intuitionistic way was Heyting's three valued system [4]. In Table 1 we can see the truth-tables of his negation and implication. In Section 2.4 more valued generalisations of this logic will be recalled (Gödel showed that to have a real intuitionistic system infinitely many values are required).

A	¬A
1	0
½	0
0	1

A→B	1	½	0
1	1	½	0
½	1	1	0
0	1	1	1

Table 1
The truth-table of Heyting's three-valued negation and implication

2.2 Belnap's Four-Valued System

Belnap has the following four epistemic values in his system [1]: true (t), unknown (u), contradictory (±) and false (f) as we can see in Fig. 1.

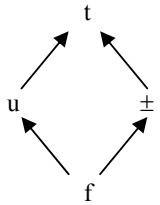

Figure 1
Truth degrees in Belnap's four-valued system

Belnap's logic has recently been used for a representation of possibly contradictory knowledge and the problem of knowledge revision and update. This logic also has some applications in the study of logic programming.

The set of truth values has the structure of a bilattice with two partial orders $\leq_{knowledge}$ and $\leq_{truth}$. The truth functions of connectives $\neg$, $\wedge$, $\vee$ are in Table 2.

A	¬A
t	f
u	u
±	±
f	t

A∧B	t	u	±	f
t	t	u	±	f
u	u	u	f	f
±	±	f	±	f
f	f	f	f	f

A∨B	t	u	±	f
t	t	t	t	t
u	t	u	t	u
±	t	t	±	±
f	t	u	±	f

Table 2
The truth functions of connectives of Belnap's logic

Furthermore we have a unary operator for each truth-value, which decides if the operand is the same as the given truth value giving answer t or f. We will use the notation Et(A), Eu(A), E±(A) and Ef(A).

2.3 Post's Systems

Post introduced the basic logical truth-tables in [9]. He suggests an alternative formal system in which the variables can have not only the traditional two truth-values but any element from the set $\{0,1,...m\}$. He started with the examination of Principia Mathematica, hence he describes his system using the connectives of negation and disjunction. Later, the conjunction derived from these basic connectives. We will use a normalised system in which the truth values are between 0 and 1: $0,1/m,...1$. The interpretations of logical operators in the $(m+1)$-valued Post system:

$$\neg A = \begin{cases} 1, & \text{if } A = 0 \\ A - \frac{1}{m}, & \text{otherwise} \end{cases} \tag{1}$$

$$A \wedge B = \min(A,B) \tag{2}$$

$$A \vee B = \max(A,B) \tag{3}$$

Negation is a circular function; it is sometimes called Post- or cyclic-negation. Post showed that his system is functionally complete. There are several interesting properties of this system according to the definition of negation; for example, De Morgan's rules do not work, etc.

2.4 The Gödelian Fuzzy-Logic

Gödel has introduced a system with infinitely many truth degrees in 1932 [2]. He defined the four basic connectives in the following way:

$$A \to B = \begin{cases} 1, \text{ if } A \le B \\ B \text{ otherwise} \end{cases} \tag{4}$$

$$\neg A = \begin{cases} 1, \text{ if } A = 0 \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

The *conjunction* and the *disjunction* are defined in the same way as in the Post's systems (see equations (2) and (3)).

As we can see, Heyting's system is a special case of Gödel's system in which we allow only three truth-values. There are finite-valued variations of the Gödel-system in which the truth-values are only *k/m* types ($m > 1$, $0 \le k \le m$) and the connectives work in the same way as in the original system. In intuitionistic logic the classical law of double negation $\neg\neg A \to A$ does not work for all possible values of A. Actually, Gödel's system is an intuitionistic system with the law of chain $(A \to B) \vee (B \to A)$.

2.5 Łukasiewicz-Type Logics

Historically Łukasiewicz made three-valued and four-valued systems [5]. Later he extended the system to arbitrary-many ($n \ge 2$) truth-values up to continuum-many. Łukasiewicz originally defined the implication and the negation. The Łukasiewicz-style logic also has & and + connectives. They are defined in the following way:

$$\neg A = 1 - A \tag{6}$$

$$A \,\&\, B = \max(A + B - 1, 0) \tag{7}$$

$$A + B = \min(A + B, 1) \tag{8}$$

$$A \to B = \begin{cases} 1, & \text{if } B \ge A \\ 1 - A + B, & \text{if } A \ge B \end{cases} \tag{9}$$

These kind of connectives are also used in finite valued logical systems, in which the values are (for each integer $k>2$): $0 = 0/(k\text{-}1)$, $1/(k\text{-}1)$, ... $(k\text{-}1)/(k\text{-}1) = 1$. (in a k valued Łukasiewicz-system, for $k=2$ we get back the classical connectives). The conjunction and the disjunction have the name "Łukasiewicz-type" connectives; they are also called "bounded product" and "bounded sum", respectively.

2.6 The Product Logic

The next widely used fuzzy system is the product logic. The connectives of this logic are the following. The *negation* is defined as at the previous case by (6).

$$A \to B = \begin{cases} 1, \text{ if } A \le B \\ \dfrac{B}{A} \text{ otherwise} \end{cases} \tag{10}$$

$$A \wedge B = A \cdot B \tag{11}$$

$$A \vee B = A + B - A \cdot B \tag{12}$$

In this system the *logical and* is the product of the values of the arguments, and the name of *or* is "algebraic sum". This type of conjunction and disjunction look more like operations of a probabilistic system. Assuming that the values A and B are independent we get the result of their common occurrence.

P(A and B) = P(A) P(B)

P(A or B) =1– P(¬A and ¬B) =1– P(¬A)P(¬B) =1– (1 – A) (1 – B) =A + B – AB
The value of A→B is the maximal probability of B if A is true.

3 The Interval-Valued Logic

In this section we define a logical system in which the truth-values are subsets of the interval [0,1]. So let the truth values be sets of points and/or disjoint intervals on [0,1]. Hence an interval-value looks like a characteristic function. Each point in [0,1] is either in the interval-value or not. We denote the empty set by the sign ⊥, and the whole set [0,1] by sign T. We allow every kind of interval, such as open (i.e. (x,y)), closed (i.e. $[x,y]$) or mixed intervals (i.e. $(x,y]$ or $[x,y)$) in an interval-value. Generally we will use type ⟨ ⟩ brackets when we do not care about the type of the interval.

3.1 Logical Operators

We use the basic connectives as in classical logic for each of the points of [0,1]. In details: let A and B be the values of two statements with interval-values. Then Table 3 shows the results of the connectives of them. The result can be defined by point-wise on [0,1] using the appropriate Boolean operator. All Boolean operators, such as *xor*, *nand*, etc. can be extended in this way to interval-values.

Name	negation	conjunction	disjunction	implication
Sign	¬A	$A \wedge B$	$A \vee B$	$A \to B$
Value	A°= [0,1] \ A	$A \cap B$	$A \cup B$	$A° \cup B$, $(A \backslash B)°$
Set theoretically	complement of A	intersection	union	complement of difference

Table 3
The basic logical operators of Interval-valued logic

3.2 Non-Logical Operators

We introduce the interval operator Mirror with which we can "interchange" the ends of the intervals.

Definition 1. Let $A = \bigcup_{i=1}^{j} \langle a_{i,1}, a_{i,2} \rangle$ be an interval-value. Then set $\text{Mirror}(A) = \bigcup_{i=1}^{j} \langle 1 - a_{i,2}, 1 - a_{i,1} \rangle$.

The Product operator has two arguments and gives a kind of product of them.

Definition 2. Let $A = \bigcup_{i=1}^{k} \langle a_{i,1}, a_{i,2} \rangle$ and $B = \bigcup_{j=1}^{l} \langle b_{j,1}, b_{j,2} \rangle$ be two interval values with number k and l components respectively. Then let the value of C=A*B be the following: let the number of components of C be $k \cdot l$. For this process we can use double indices for the components of C. Let the starting and ending point of the ijth component be $a_{i1}+b_{j1}(a_{i2}-a_{i1})$ and $a_{i1}+b_{j2}(a_{i2}-a_{i1})$ respectively. (An endpoint belongs to the interval iff the original endpoints belong to the interval-values of the arguments.)

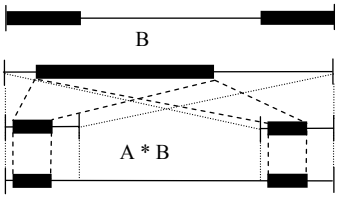

Figure 2
Example for Product of interval-values

As we can see in the definition, this product looks like the Cartesian product; the indexes works in the same way. The component ij of C comes from the ith component of A and the jth component of B.

Now we will examine the reverse operator of Product. Generally it has no inverse; we can show interval-value which cannot occur as a result of a product with a given operand. We need a reverse operator, but not in every case. So we can restrict our analysis to interval values of the type $[0,x]$. Let the interval value of A be $[0,a]$ and the interval value of B be $[0,b]$. Then the value of their product A*B is $[0,ab]$. In this case the product is symmetric. As we can see the type of the result is also $[0,x]$. The question is what the other argument of the product is, if the result and an argument are given. We define the operator Divide in the following way.

Definition 3. Given the interval-values A and B in the form [0,*a*] and [0,*b*] respectively. If *b*>*a* then let Divide(B,A)=T. In each of the other cases let Divide(B,A) = [0,*b*/*a*].

To have connections between the many-valued (fuzzy) logics and the interval-valued system we need operators connect the real numbers to interval-values and vice-versa. They will be helpful in simulation of the traditional fuzzy systems.

Definition 4. The ZeroStart operator assigns the interval-value [0,*x*) to the real number *x* between 0 and 1.

Using ZeroStart and Mirror one can create interval-values type (1-*x*,1].

Definition 5. The Measure operator assigns the real value of the sum of the length of components to an interval-value. (We use the Lebesgue measure.)

Definition 6. Let A be an interval-value. FromZeroMeasure(A) is the real value *x*, where *x* is the maximal value such that (0,*x*) is part of A.

Finally, in the Interval-valued logic we introduce shift operator with two interval-valued arguments.

Definition 7. Let $A = \bigcup_{i=1}^{j} \langle a_{i,1}, a_{i,2} \rangle$, and let *b* = FromZeroMeasure(B). Then $\mathrm{Lshift}(A,B) = \bigcup_{i=1}^{j} \langle \max(a_{i,1} - b, 0), \max(a_{i,2} - b, 0) \rangle$. (So the value of each point *p* in A goes to the point (max(*p*-*b*,0)) in Lshift(A,B). The value in point 0 is the value of point *b* in A, the other part (above 1-*b*) remains 0, i.e. empty.

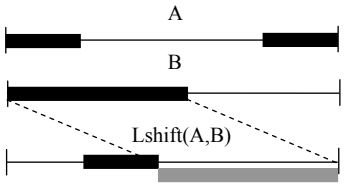

Figure 3
Example for the operator Lshift

3.3 Properties of the System

The Interval-values have a natural partial order, namely the set-theoretical subset relation. It is connected to the implication: $A \to B = T$ iff $B \supseteq A$. Having T as true and $\perp$ as false the logical laws of this system are the same formulae as the laws of the classical logic. The logical operations act in a vertical way (pointwise), the non-logical operators (such as the measure operators) act somehow in a horizontal

way. Several other properties are detailed in [7]. The Interval-valued logic with non-logical operators forms a universal computing device ([8]).

4 Simulation of Fuzzy Systems by Interval-Values

In this section we present our main results.

4.1 Belnap's Four-Valued System

We can make the following mapping: f = ⊥, u = [0,1/2] , ± = (1/2,1], t = T (see Fig. 4 as well). Then the operators of Belnap's system can be simulated in the following way.

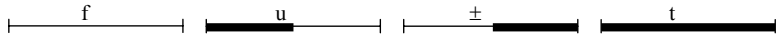

Figure 4
Belnap's four values with interval values

Negation: Let us make the interval negation of the operand and after this use the Mirror operator.

Conjunction and *disjunction* work in a natural way; we use the same interval connectives and we translate back the result to {f, u, ±, t} values.

Unary operators: 'xor' on intervals and we need a unit interval, namely u)

Et(A) = A ⊕ T (where T can be written as $u \vee \neg u$),

Eu(A) = A ⊕ u (unit)

E±(A) = A ⊕ ¬u

Ef(A) = A ⊕ ⊥ (⊥ can be written as $u \wedge \neg u$).

4.2 Post's Systems

The truth-value x of the Post system is represented by the interval-value $[0,x]$. To represent the negation we need a parameter interval-value, which represents the unit $(1/m)$. We represent the operators of this system in the following way:

The *negation* of the Post system is the same as the following expression in interval-valued logic:

Zerostart(FromZeroMeasure(¬ZeroStart(x)))∨Lshift(ZeroStart(x),ZeroStart($1/m$)).
Its Measured value is exactly the value of ¬x in Post logic.

The connectives logical *or* and *and* can be interpreted in a natural way as we can see in Figure 5. We can use the correspondent logical connectives of interval-values between the interval-values of arguments, and we can use Measure the get the result.

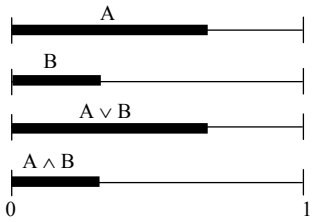

Figure 5
Conjunction and disjunction in Post logic

4.3 Heyting's and Gödel's Logics

We will interpret only the infinite-valued Gödelian logic; the finite valued systems (as specially the Heyting's one as well) work in the same way. Represent the real number x by the interval-value $[0,x]$ as we defined it with ZeroStart.

Procedure 1

1 Use the ZeroStart operator for all arguments.

2 Use the correspondent logical operator of interval-valued logic.

3 Use the FromZeroMeasure operator to read the result.

In Figures 5, 6 and 7 one can see examples for the basic connectives.

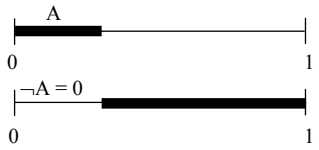

Figure 6
Interpretation of Gödelian negation with interval-values

The value $\neg 0 = 1$, because in this case the result interval-value is T, which is 1 FromZeroMeasured. The result for any other argument is 0 as we can see in the Figure 6. The logical *an*' and *or* work in the same way as in the Post-system. (See Fig. 5, the results are the same whether we use Measure or FromZeroMeasure of the interval-values type $[0,x]$.)

If A>B then FromZeroMeasured $A \rightarrow B$ is really the same as B, while $B \rightarrow A$ is T, i.e. in case of $A \geq B$, the truth-value of $B \rightarrow A$ is 1.

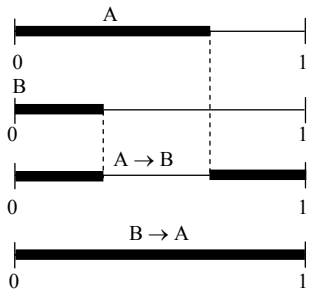

Figure 7
Gödelian implication with intervals

So, in this system we loose the parts of the interval-values which are not connected to the point 0 after a logical operation, and it occurs after negation and implication. It means that this system is intuitionistic. We can use only the parts $\langle 0,x\rangle$ of the resulted interval-values. In this way it is easy to read (and translate back to Gödelian logic) the result of any kind of logical operation.

The interval-valued logic can interpret why the classical double-negation law does not work, and how we can embed classical logic into the intuitionistic system (we must use double negation in front of all classical formulae and we get back the well-known classical tautologies with two values.)

4.4 Łukasiewicz-Type Logics

Now we can show how we can interpret the Łukasiewicz-style operators.

Procedure 2

1 Use ZeroStart operator for the arguments.

2 In case of conjunction (&) or disjunction (+) use the Mirror operation for the interval-value representing the second argument.

3 Use the correspondent logical operator of interval-valued logic.

4 Use the Measure operator to read the result.

In details:

Negation: $x \Rightarrow [0,x] \Rightarrow (x,1] \Rightarrow 1\text{-}x$, as we want.

As we can see in Figure 8, if the sum of values A and B is not greater than 1, then the result of their *conjunction* is the empty interval $\perp$. Their *disjunction* has the same value as their sum. If the sum of the lengths of arguments is greater than 1 (A, C in Figure 8) then their *conjunction* has the value which is 1 less than their sum. In this case the result of their *operation* + is the whole interval T.

Implication: we can write the following form: $A \rightarrow B = \min(1, 1 - A + B)$.

In this case our interval-value after the interval implication is same as in the Gödelian case, but now we use the Measure operation instead of FromZeroMeasure. So in the case of $A \leq B$ the result is T, which means 1 in Measured value. In the case of $A > B$ our interval result contains two interval components, the second of which has its length (1-A) missing at FromZeroMeasure, but the Measured result is (1-A+B).

The results of the infinite Łukasiewicz-system still work in finite valued Łukasiewicz-systems when we restrict our interval values to only the values which can be obtained by the sum of correspondent intervals (they have starting and ending points at 0, 1/(*k*-1), ..., (*k*-2)/(*k*-1), 1).

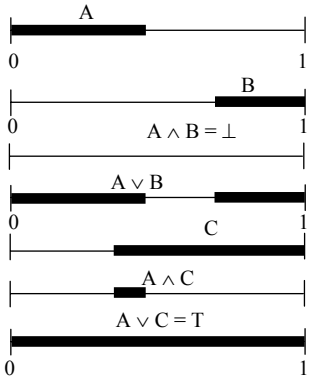

Figure 8

Calculating the Łukasiewicz-style connectives with intervals

4.5 The Product Logic

Let the used interval-values be the ZeroStart values of the values at the product logic.

The *negation* is same as many systems before, hence we can interpret it in the same way: we get the interval-value with ZeroStart, we use the interval negation, and we use Measure operator to get the result.

The logical *and* of product logic works in the following way: we get the interval values of the parameters using ZeroStart, we make the Product (*) of them, and we use the Measure operator.

We can make the logical '*or*' in one of the following ways (they are equivalent):

- using the DeMorgan law: make the negations of the interval-values of the parameters, then make the Product of these values, and then make the negation of the result. With Measure operator it is the result.

- we can use the interval or with one of the operands and the Product of its negation and the other operand.

The *implication* works in the following way: use ZeroStart for the arguments, after this we use Divide(second argument, first argument) and use Measure to get the result in real numbers.

Conclusions

In this paper a many-valued logic is presented, namely the interval-valued logic. The logical values of the system are contains points and intervals over the unit interval [0,1]. Logical and non-logical operators are defined, and by their help the simulation of several many-valued and fuzzy systems are showed. We can conclude that the interval-valued logical system works as a generalisation of the many-valued and fuzzy systems. One can use this general system with a high flexibility due to the non-logical operators.

References

[1] N. D. Belnap: A useful four-valued logic, Modern Uses of Multiple-valued Logic, (editors: J. M. Dunn-G. Epstein), Reidel 1975, pp. 9-37

[2] K. Gödel: Zum intuitioonischen Aussagenkalkül. Anzeigner Akademie der Wissenschaften im Wien, Mathematish-Naturwissenschaftliche Klasse, **69**, 1932, pp. 65-66

[3] P. Hájek: Metamathematics of Fuzzy Logic, (Trends in Logic **4**), Kluwer Academic Publ., Dordrecht 1998

[4] A. Heyting: Die formale Regeln der intuitionischen Logik, Sitzungsberchte der preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse, 1930

[5] J. Łukasiewicz: Logice trójwartościowej, Ruch Filozoficzny, r. **5** N 9 Lwów, 1920, pp. 169-171

[6] B. Nagy: Intervallum logika (fuzzy logikai vizsgálódások), Különdíjas OTDK dolgozat, (in Hungarian), Proceedings of Nationwide Scientific Student Conference, Eger Hungary, 1997

[7] B. Nagy: Interval-valued logic as generalization of many valued logics, Technical Report 2002/20, Institute of Mathematics and Informatics, University of Debrecen 2002, 36 pages

[8] B. Nagy: An Interval-valued Computing Device, Proceedings of Computability in Europe: New Computational Paradigms, Amsterdam, 2005, pp. 166-177

[9] E. L. Post: Introduction to a General Theory of Elementary propositions, American Journal of Mathematics **43** N 3 1921, pp. 163-185

[10] P. Vojtas: Propositional Abduction with Uncertainty, Preprint 14/1997, P. J. Safárik University Kosice, 1997, 12 pages