# $\alpha$-Satisfiability and $\alpha$-Lock Resolution for a Lattice-Valued Logic LP(X)


**Xingxing HE, Yang XU, Xiaomei ZHONG**

*Southwest Jiaotong University, Chengdu, Sichuan, China*

**Jun LIU**

*University of Ulster, Northern Ireland, UK*

**Luis MARTINEZ**

*Department of Computing, University of Jaén, E-23071 Jaén, Spain*

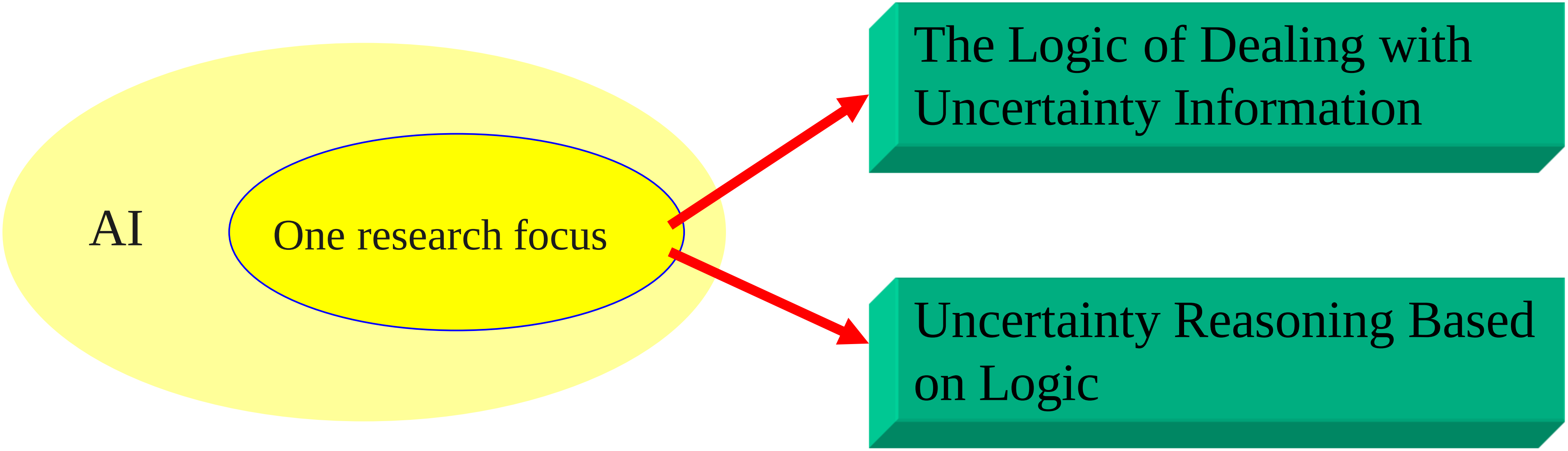
**Da RUAN**

*Belgian Nuclear Research Centre (SCK.CEN) and Ghent University, Belgium*

# Outlines

- **Introduction**
- **Academic Background and Ideas**
- **Focused Technical Works**
- **Ongoing Research and Prospects**
- **Conclusion**

# Research View and Orientation




**Logic Based Intelligent Systems**

# Study of logic foundation for uncertainty reasoning: especially incomparability

- **Key ideas**

  Intelligent information processing → Uncertain Information → Uncertainty Reasoning → Need for establishing strict logic foundation → Non-Classical logic → Incomparable information → Lattice-valued logic system with truth-valued in a lattice

**Lattice + Logic**

- **Logical algebraic structure – lattice implication algebras (LIA)**

  Combining **lattice** and **implication** algebra, non-chain structure

- **Lattice-valued logic systems based on LIA**

  Incomparable information → Relation with fuzzy logic → Universal Algebra → Truth-valued attached → Syntax and semantics extension → Complete and Sound lattice-valued logic system

# Academic routine since 1993

- Lattice-valued logical algebra — **Lattice Implication Algebra** (LIA)
  - **Y. Xu, Lattice implication algebra, *Journal of Southwest Jiaotong University* (in Chinese), 1993, 1, pp. 20-27.**

↓

- **Structure and properties** of LIA
- **Lattice-valued algebraic logic** — lattice-valued logic based on LIA
- **Approximate reasoning** based on lattice-valued logic
- **Automated reasoning** based on lattice-valued logic

# A lattice-valued logical algebra -- lattice implication algebra (LIA)

**Definition** (LIA) Let $(L, \vee, \wedge, ')$ be a bounded lattice with an order-reversing involution " $'$ " and the universal bounds O, I, : $L \times L \to L$ be a mapping. $(L, \vee, \wedge, ', \to)$ is called a **lattice implication algebra** (LIA) if the following conditions hold for all $x, y, z \in L$:

($I_1$) $x \to (y \to z) = y \to (x \to z)$ (exchange property)

($I_2$) $x \to x = \mathrm{I}$ (identity)

($I_3$) $x \to y = y' \to x'$ (contraposition or contrapositive symmetry)

($I_4$) $x \to y = y \to x = \mathrm{I}$ implies $x = y$ (equivalency)

($I_5$) $(x \to y) \to y = (y \to x) \to x$

($I_6$) $x \to (y \vee z) = (x \to y) \vee (x \to z)$ (implication $\vee$ -distributivity)

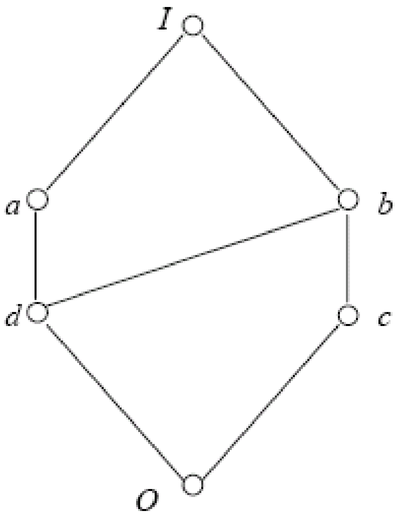
($I_7$) $x \to (y \wedge z) = (x \to y) \wedge (x \to z)$ (implication $\wedge$ -distributivity)

# Examples of LIA

Boolean algebra and Lukasiewicz algebra are all LIAs. A class of all LIAs form a proper class, which means many LIAs can be constructed and there are at least countable LIAs which can be constructed in [0, 1]

| $x$ | $x'$ |
|---|---|
| O | I |
| $a$ | $c$ |
| $b$ | $d$ |
| $c$ | $a$ |
| $d$ | $b$ |
| I | O |

| $\rightarrow$ | O | $a$ | $b$ | $c$ | $d$ | I |
|---|---|---|---|---|---|---|
| O | I | I | I | I | I | I |
| $a$ | $c$ | I | $b$ | $c$ | $b$ | I |
| $b$ | $d$ | $a$ | I | $b$ | $a$ | I |
| $c$ | $a$ | $a$ | I | I | $a$ | I |
| $d$ | $b$ | I | I | $b$ | I | I |
| I | O | $a$ | $b$ | $c$ | $d$ | I |




→ **Non-chain LIA**

# Book published (2003)

- Xu, Y., Ruan, D., Qin, K.Y., and Liu, J., *Lattice-Valued Logic – An Alternative Approach to Treat Fuzziness and Incomparability*, Springer-Verlag, Heidelberg, July, 2003, 390 pages.
- ISBN-3-540-40175-X

# The main focus of this paper: resolution-based automated reasoning

- Feature and properties of the logical formula which includes **constants** in LP(X)
- **Simplify** the structure of the generalized literals in LP(X)
- **Improve the efficiency** of α- resolution in lattice-valued logic, an **α-lock resolution** method based on LP(X) is proposed and the soundness and weak completeness of this method has been proved

# The essence of classical automated reasoning methods

- The kernel problem in classical automated reasoning
  - $A_1, \ldots, A_n \Rightarrow B$ ? *or if* $A_1 \wedge \ldots \wedge A_n \rightarrow B$ *is a Theorem*?
- The problem is transformed into validating the unsatisfiability of a logical formula variation of this theorem
  - $A_1 \wedge \ldots \wedge A_n \rightarrow B$ *is a theorem iff* $A_1 \wedge \ldots \wedge A_n \vee \sim B$ *is unsatisfiable*
- An algorithm needs to be constructed to prove the unsatisfiability of this logical formula
- The resolution method is of great importance on mechanical theorem proving in classical logic

# The α-automated reasoning algorithm in LP(X)

- **Definition ($\alpha$-false )** Let $\varphi$ be a generalized logic formula in LP(X). $\varphi$ is said to be always false at a truth-value level $\alpha$ ($\alpha$-false in short) if for an arbitrary valuation $\gamma$ such that $\gamma(\varphi)\leq\alpha$.

- **An α-Automated reasoning algorithm in LP(X) can be obtained as the similar way in two-valued logic**
  - **search and delete the <u>$\alpha$-false pairs</u>**

- **Soundness and completeness)** $S \leq \alpha$ iff the α-automated reasoning algorithm in LP(X) terminates on α-empty clause.

# About a generalized conjunctive normal form in LP(X)

- **Definition 7** (***an extremely simple form f***, in short ESF) if an *L*-valued propositional logical formula $f^*$ obtained by deleting any constant or literal or implication term appearing in *f* is not equivalent to *f*.

- **Definition 8** (***an indecomposable extremely simple form***, in short IESF) if *f* is an ESF containing no connectives other than implication connectives.

- **Definition 9** All the constants, literals and IESF´s are called ***generalized literals***.

- **Definition 10** An *L*-valued propositional logical formula *G* is called ***a generalized clause***, if *G* is a formula of the form:

  $$G=g_1\vee\ldots\vee g_i\vee\ldots\vee g_n$$

  where $g_i$ ($i$=1,... , $n$) are generalized literals.

- A conjunction of finite generalized clauses is called ***a generalized conjunctive normal form***.

# α-Resolution Principle

**Definition 12.** [6] *(α-Resolution). Let $\alpha \in L$, and $G_1$ and $G_2$ be two generalized clauses of the forms:*

$$G_1 = g_1 \vee \ldots \vee g_i \vee \ldots \vee g_m$$
$$G_2 = h_1 \vee \ldots \vee h_j \vee \ldots \vee h_n$$

*If* $g_i \wedge h_j \leq \alpha$

$$G = g_1 \vee \ldots \vee g_{i-1} \vee \ldots \vee g_{i+1} \vee \ldots \vee h_1 \vee \ldots \vee h_{j-1} \vee \ldots \vee h_{j+1} \vee \ldots \vee h_n$$

*is called an α-resolvent of $G_1$ and $G_2$, denoted by $G = R_\alpha(G_1, G_2)$, and $g_i$ and $h_j$ form an α-resolution pair, denoted by $(g_i, h_j) - \alpha$. Generation of an α-resolvent from two clauses, called α-resolution, is the sole rule of inference of the α-resolution principle.*

# Simplify the structure of the generalized literals in LP(X)

- $\alpha$ -Valid Rule
- Unit generalized literal rule
- Pure generalized literal rule
- Splitting rule

# $\alpha$ -Lock resolution method in *LP*(*X*)

**Definition 16.** *Let $G$ be a generalized clause in $L_nP(X)$, each occurrence of a generalized literal in $G$ is assigned a positive integer in the lower left corner (the same generalized literals can be labeled different positive integer), this specific generalized clause $G$ is called a lock generalized clause, and the positive integer in the generalized literal is called a lock index.*

**Definition 17.** *Let $G$ be a lock generalized clause in $L_nP(X)$. Suppose that $G$ contains generalized literals which have the same name with different indices, then delete the generalized literals with larger indices. This process is called amalgamation.*

**Definition 18.** *Let $G_1$ and $G_2$ be two generalized clauses in $L_nP(X)$, $\alpha \in L_n$. $G = R_{\alpha L}(G_1, G_2)$ is called an $\alpha$-lock resolvent of $G_1$ and $G_2$ if it satisfies the following conditions.*

(1) *$G$ is the $\alpha$-resolvent of $G_1$ and $G_2$.*

(2) *The $\alpha$-resolvent generalized literals in $G_1$ and $G_2$ have the minimal indices respectively.*

# α -Lock resolution method in *LP*(*X*)

**Definition 19.** *Let $S$ be a finite generalized clause set in $L_nP(X)$, and all generalized literals in $S$ are assigned lock indices. An $\alpha$-resolution deduction from $S$ is called an $\alpha$-lock deduction if each $\alpha$-resolution in the deduction process is an $\alpha$-lock resolution. An $\alpha$-lock deduction of from $S$ to $\alpha$-empty clause is called an $\alpha$-lock proof of $S$.*

**Theorem 5.** (Soundness Theorem). *Let $S$ be a finite generalized clause set in $L_nP(X)$, and all generalized literals in $S$ are assigned lock indices. $\{D_1, D_2, \ldots, D_m\}$ is an $\alpha$-lock resolution deduction from $S$ to a generalized clause $D_m$. If $D_m \le \alpha$, then $S \le \alpha$.*

**Theorem 6.** (weak completeness theorem). *Let $S$ be a finite generalized clause set in $L_nP(X)$, and all generalized literals in $S$ are assigned lock indices. Let $\alpha \in L_n$ and $\vee_{a \in L_n}(a \wedge a') \le \alpha < I$. If $S \le \alpha$, then there exists an $\alpha$-lock deduction of from $S$ to $\alpha$-empty clause.*

# Potential applications

- Machine intelligence needs the investigation of linguistic valued uncertainty reasoning
  - Human beings bound to express ourselves in a natural language that uses words
  - A nice feature of linguistic term set
    - Their values are structured, makes it possible to compute the representations of composed linguistic values from those of their composing parts
- Lattice-based linguistic truth-valued algebra
- **Symbolic approach - direct computation on linguistic values**
- **Computing with Words $\Rightarrow$ Reasoning with words**

# Linguistic-valued logic scheme

- In general, we conjecture that the domain of a linguistic-valued algebra (LA) can be represented as a lattice. Thus, a linguistic-valued logic is a logic in which the truth degree of an assertion is a linguistic value in LA.
- Use natural language to express a logic in which the truth values of propositions are expressed as linguistic values in natural language terms such as *true*, *very true*, *less true*, *very false*, *false*, etc., instead of a numerical scale.

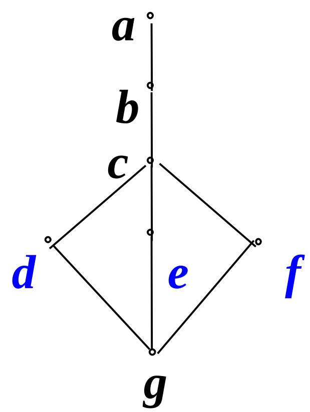
↓

**Reasoning with words**

# Some values of linguistic variable cannot be strictly linearly ordered

- **Linguistic variables take natural language words or labels as values**
- **Some words seem difficult to distinguish their boundary**
- **There are some vague "overlap district" among some words**

Fig. 1 The ordering relationships in linguistic terms:




*a=very True, b=more True, c=True, d=Approximately True*
*e=possibly True, f=more or less True, g=little True*

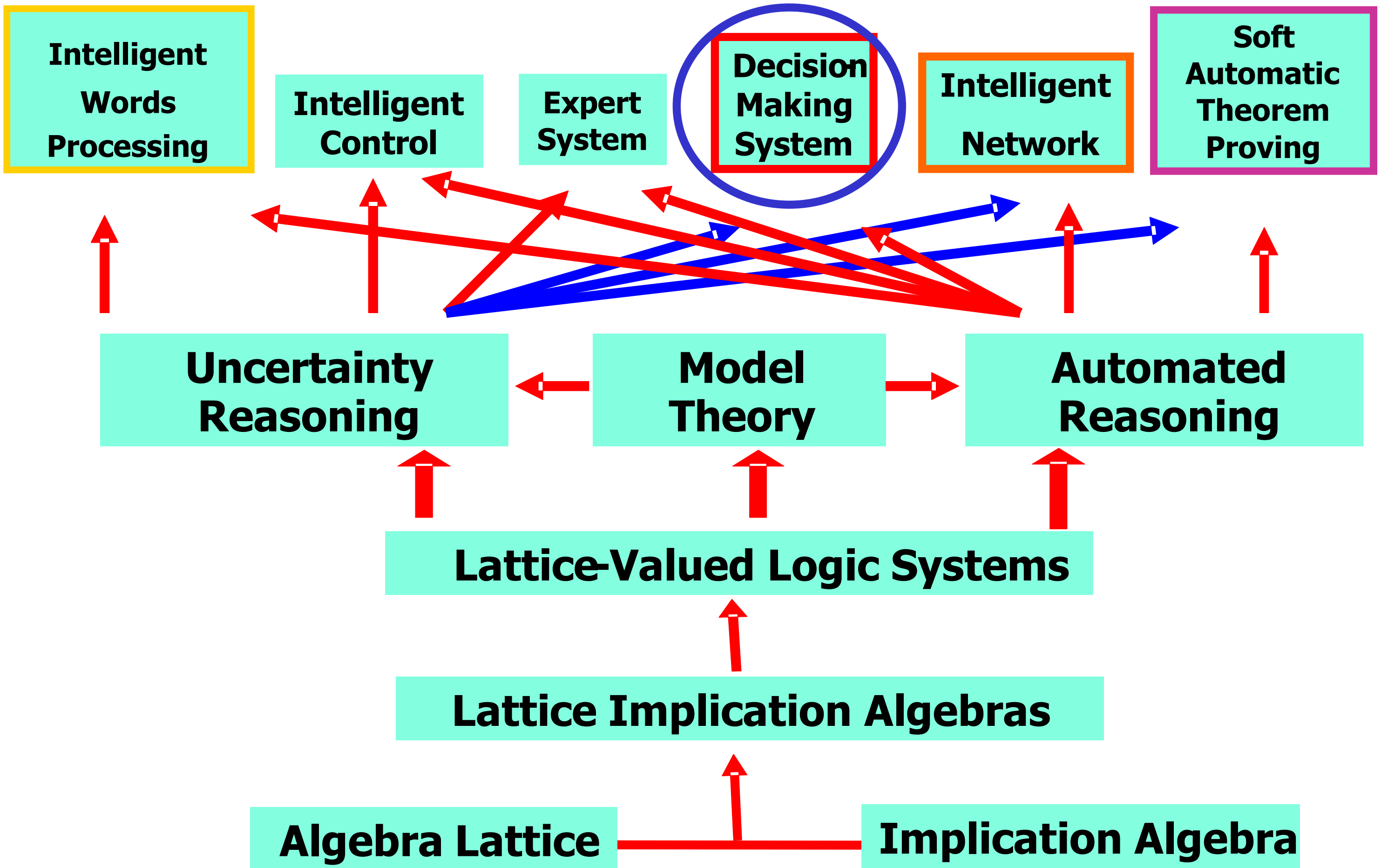
# Lattice-valued logic algebra can be used to construct linguistic value algebra

- It should be suitable to represent the linguistic values by a partially ordered set or lattice.
- LIA is an extension of Boolean algebra by combining a lattice and the implication operator
- The axiomatic definition of implication operator
- The operations can be decided upon the elements and their orders are given.
- **LIA used to construct linguistic value algebra with lattice order**

# Lattice-valued linguistic based automated reasoning and decision making

- **Representing linguistic terms**
  - Linguistic truth-value lattice-implication algebra
  - Linguistic atom term, logically composed terms, modified terms with a set of linguistic modifiers (hedges)
  - Their ordering relationship
  - Structure and characteristic
- **Lattice-valued linguistic resolution-based automated reasoning**
  - Structure and transformation, resolution principle, structure of resolution field, algorithm and programming
- **Application in decision making**

# A sketch map on research views, activities and directions



# Thank you !