α-Quasi-Lock Semantic Resolution Method Based on Lattice-Valued Logic

Xiaomei Zhong, Yang Xu
School of Mathematics, Southwest Jiaotong University, Chengdu, Sichuan 610031, P.R. China
E-mail: zhongxm2013@home.swjtu.edu.cn, xuyang@home.swjtu.edu.cn

Jun Liu, Shuwei Chen
School of Computing and Mathematics, University of Ulster, Northern Ireland, UK
E-mail: j.liu@ulster.ac.uk, chensw915@gmail.com

Abstract

Based on the general form of α-resolution principle for a lattice-valued logic with truth-values defined in a lattice-valued logical algebra structure - lattice implication algebra, the further extended α-resolution method in this lattice-valued logic is discussed in the present paper in order to increase the efficiency of the resolution method. Firstly, α-quasi-lock semantic resolution method in lattice-valued propositional logic LP(X) is established by combining the lock and semantic resolution simultaneously, and its theorems of soundness and conditional completeness are proved. Secondly, this α-quasi-lock semantic resolution method is extended into the corresponding lattice-valued first-order logic LF(X), and its soundness and conditional completeness are also established. This extended resolution method will provide a theoretical basis for automated soft theorem proving and program verification based on lattice-valued logic.

Keywords: α-Quasi-lock semantic resolution method; resolution-based automated reasoning; general form of α-resolution principle; lattice-valued logic; lattice implication algebra.

1. Introduction

Since resolution principle based on classical logic was proposed by Robinson in 1965 [1], resolution-based automated reasoning has been widely applied to various areas, such as mathematics, biology, engineering technologies. The study about resolution-based automated reasoning methods in classical logic attracted a lot of researchers’ interest and some important results about variation or extension of resolution principle have been achieved, such as, in 1965, Wos et al. [2] proposed the resolution strategy based on support sets, and its soundness and completeness were also obtained. In 1967, Slagle [3] established semantic resolution method and proved its soundness and completeness. Afterwards, Loveland [4] and Luckham [5] proposed linear resolution method by their respective views. In order to select the unique resolution literal during the process of resolution, Reiter [6] established ordered resolution method and ordered semantic resolution method, but the latter did not have completeness, even for ground clause sets. In order to improve the efficiency of resolution-based automated reasoning, in 1971, Boyer proposed lock resolution method in his doctoral thesis at the University of Texas, and proved its soundness and completeness theorems. In 1981, Huang [7] improved linear resolution method and established MOL resolution method, its soundness and completeness theorems were also proved. In 1979, 1985, 1987 and 1992, Liu [8-11] did in-depth research on the compatibility among semantic resolution method, linear
resolution method and lock resolution method, as well as the corresponding soundness and completeness. According to the improvement of semantic resolution method, Lu et al. [12] proposed colored resolution method and also proved its soundness and completeness theorems. In 2005, Cai [13] discussed the realization of the resolution method based on the strategy of support sets. In 2007, Meng et al. [14] improved the efficiency of resolution by checking the correlation of symbols in clauses.

From the above short review, in classical logic, there are mainly three kinds of resolution-based automated reasoning methods, i.e., lock resolution method, semantic resolution method and linear resolution method. Lock resolution method improves the efficiency of automated reasoning through limiting resolution literals to each literal equipped with a lock and implement resolution by the smallest lock rule. To some extent, this method can limit the generation of redundant clauses during the process of resolution and improve the efficiency of automated reasoning. Semantic resolution method and linear resolution method improve the efficiency of automated reasoning by limiting resolution clauses, which is to say that they limit resolution clauses by certain ways respectively, so as to reduce the number of redundant clauses occurring in the process of resolution and improve the efficiency of automated reasoning. In other words, these three kinds of resolution-based automated reasoning methods improve the efficiency of automated reasoning from two different views. If we can establish another method containing the benefits of the above three kinds of resolution-based automated reasoning methods, i.e., this new method reduces the generation of redundant clauses by limiting resolution clauses and literals simultaneously, then we can further improve the efficiency of resolution-based automated reasoning to some extent.

In another aspect, in the real world, people living in the environment with much uncertainty often need to make judgment with uncertainty (“soft conclusion”) based on uncertain environment, information with uncertainty (“soft premise”) and knowledge with uncertainty (“soft rules”). We call this “soft causal relationship” that “soft premise” and “soft rules” draw “soft conclusion” as a “soft theorem”. People often discover such “soft theorems”, and also need to verify their rationality (or correctness) through practice or methods. Non-classical logic has been a considerably useful formal tool for computer science and AI during the past decade. Many-valued logic is a powerful extension and development of classical logic, which aims to establish the logical foundation for “soft” information processing. Lattice-valued logic, as one of the most important many-valued logics, extends the chain-type truth-valued field to a general lattice in which the truth-values are incompletely comparable with each other. Lattice-valued logic is thus an important and promising research direction that provides an alternative logical approach to dealing with imprecision and incomparability as well [15]. As the automated reasoning method based on resolution principle for classical logic is an important class of automated reasoning methods in the field of “theorem machine proving”, in order to make machines can simulate people verifying these “soft theorems”, i.e., make machines automatically verify these “soft theorems” by reasoning, similar to the academic thinking of “theorem machine proving”, it is very important for us to establish an appropriate resolution principle in non-classical logics including many-valued logics even lattice-valued logics and some effective resolution methods based on them. With the progress of society, more and more uncertain information needs to be handled in the real world.

Taking the above ideas into consideration, the resolution principle based on lattice-valued logic with truth-value in a lattice-valued logical algebraic structure - lattice implication algebras was established by Xu et al. [16-17], which can be used to prove whether a lattice-valued logical formula is false at a truth-value level $\alpha$ (i.e., $\alpha$-false) or not in order to characterize incomparability and fuzziness. After that, some researchers did further research on the theory of resolution-based automated reasoning for the above lattice-valued logic and obtained some important results. For example, in 2007, Xu et al. [18, 19] discussed the relation between $\alpha$-resolution for lattice-valued propositional logic LP(X) and that for lattice-valued first-order logic LF(X), and pointed out the fact that $\alpha$-resolution for LF(X) can be equivalently transformed into that for LP(X). As an application of $\alpha$-resolution principle, Xu et al. [20] studied $\alpha$-resolution-based automated reasoning for linguistic truth-valued lattice-
valued propositional logic $\mathcal{L}_{\mathbf{TP}(X)}$. In 2008, Li [21] obtained some properties of $\alpha$-resolution-based automated reasoning in lattice-valued logic, and its soundness and weak completeness. In the same year, Xu et al. [24] proposed $\alpha$-generalized resolution principle based on lattice-valued propositional logic $\mathcal{L}(X)$, and its soundness and weak completeness were also established.

As a continuation of the above research work, on the basis of lock resolution method and semantic resolution method in classical logic, this paper will establish a lock semantic resolution method in lattice-valued logic. Furthermore, the efficiency of $\alpha$-resolution-based automated reasoning in lattice-valued logic will be improved by $\alpha$-lock resolution method.

2. Preliminaries

In the following, we will introduce some elementary concepts and conclusions of lattice-valued logic and the general form of $\alpha$-resolution principle. We refer the readers to [15, 23] for more details.

2.1. Lattice implication algebra

Definition 1. [15] Let $(L, \wedge, \vee, O, I)$ be a bounded lattice with an order-reversing involution $'$, and $O$ the greatest and the smallest element of $L$ respectively. Let $D: L \times L \to L$ be a mapping. $(L, \wedge, \vee, ', \to, O, I)$ is called a lattice implication algebra (LIA) if the following conditions hold for any $x, y, z \in L$:

$$(1) x \to (y \to z) = y \to (x \to z),$$
$$(2) x \to x = I,$$
$$(3) x \to y = y' \to x',$$
$$(4) x \to y = y \to x = I \iff x = y,$$
$$(5) (x \to y) \to z = (x \to z) \wedge (y \to z),$$
$$(6) (x \wedge y) \to z = (x \to z) \vee (y \to z).$$

Example 1. [15] (Łukasiewicz implication algebra on finite chain) Let $L_n = \{a_i | i = 1, 2, \ldots, n\}$, $a_1 < a_2 < \ldots < a_n$. For any $1 \leq j, k \leq n$, define

$$\begin{align*}
a_j \vee a_k &= a_{\max(j,k)}, \\
a_j \wedge a_k &= a_{\min(j,k)}, \ \\
(a_j)' &= a_{n-j+1}, \\
da_j \to a_k &= a_{\min(n-j+k, n)}.
\end{align*}$$

Then $(L_n, \vee, \wedge, ', \to, a_1, a_n)$ is a LIA.

Example 2. [25] Let $L_2 = \{b_1, b_2\}$, $b_1 < b_2$, $S_2 = (L_2, \vee, \wedge, ')$, $\to$, $b_1, b_2$ is also a Łukasiewicz implication algebra. For any $(a_i, b_i) \in L_2 \times L_2$, define

$$\begin{align*}
(a_i, b_i) \vee (a_j, b_j) &= (a_i \vee a_j, b_i \vee b_j), \\
(a_i, b_i) \wedge (a_j, b_j) &= (a_i \wedge a_j, b_i \wedge b_j), \ \\
(a_i, b_i)' &= (a_i', b_i'), \\
(a_i, b_i) \to (a_j, b_j) &= (a_i \to a_j, b_i \to b_j).
\end{align*}$$

Then $(L_2 \times L_2, \vee, \wedge, ')$, $(a_1, b_1)$, $(a_n, b_2)$ is a LIA, denoted as $S_2 \times S_2$.

2.2. Lattice-valued propositional logic $\mathcal{L}(X)$

Definition 2. [26] Let $X$ be the set of propositional variables, $(L, \vee, \wedge, \to, O)$ be a LIA, $T = L \cup \{', \to\}$ be a type with $ar() = 1$, $ar(\to) = 2$ and $ar(a) = 0$ for any $a \in L$. The proposition algebra of the lattice-valued propositional logic is calculated on the set $X$ of propositional
variables is the free $T$ algebra on $X$ and denoted by $LP(X)$.

**Definition 3.** [26] The set $F$ of formulas of lattice-valued propositional logic $LP(X)$ is the least set $Y$ satisfying the following conditions:

1. $X \subseteq Y$,
2. $L \subseteq Y$,
3. if $p, q \in Y$, then $'(p) \rightarrow (p, q) \in Y$,

where $X$ is the set of propositional variables, $L$ is the set of constants.

In the following, we denote $'(p)$ as $p'$ and $'(p, q)$ as $p \rightarrow q$.

**Definition 4.** [26] A mapping $v: LP(X) \rightarrow L$ is called a valuation of lattice-valued propositional logic $LP(X)$, if it is a $T$-homomorphism.

**Definition 5.** [27] Let $G \in F$ and $\alpha \in L$. If $\nu(G) \leq \alpha$ for any valuation $v$ of lattice-valued propositional logic $LP(X)$, we say $G$ is always less than or equal to $\alpha$ (or $G$ is $\alpha$-false), denoted by $G \leq \alpha$.

**Definition 6.** [15] A lattice-valued propositional logical formula $G$ in lattice-valued propositional logic system $LP(X)$ is called an extremely simple form, in short ESF, if a lattice-valued propositional logical formula $G'$ obtained by deleting any constant or literal or implication term occurring in $G$ is not equivalent to $G$.

**Definition 7.** [15] A lattice-valued propositional logical formula $G$ in lattice-valued propositional logic system $LP(X)$ is called an indecomposable extremely simple form, in short IESF, if the following two conditions hold:

1. $G$ is an ESF containing connectives $\rightarrow$ and $'$. at most,
2. for any $H \in F$, if $H \leq G$ in $LP(X)$, then $H$ is an ESF containing connectives $\rightarrow$ and $'$. at most, where $LP(X) = ( F, \vee, \wedge, ', \rightarrow)$ is a LIA, $F = F/\neg = \{ p \mid p \in F \}, \neg = \{ q \mid \text{for any valuation } v \text{ in } LP(X), v(q) = \neg v(p) \}, \text{for any } p, q \in F, \overline{p \lor q} = \overline{p} \lor \overline{q}, \overline{p \land q} = \overline{p} \land \overline{q}$, $(p \rightarrow p') = \overline{p'}, p \rightarrow q = \overline{p} \rightarrow q$.

**Definition 8.** [15] All the constants, literals and IESFs in $LP(X)$ are called generalized literals. Here, the definition of literal is the same as that in classical logic.

The disjunction of a finite number of generalized literals is a generalized clause.

**Definition 9.** [23] Let $C_i = p_{i1} \lor ... \lor p_{im}$ be generalized clauses of $LP(X)$, $H_i = \{ p_{i1}, ..., p_{im} \}$ the set of all disjuncts occurring in $C_i$, $i = 1, 2, ..., m$, $\alpha \in L$. For any $i \in \{1, 2, ..., m\}$, if there exist generalized literals $x_i \in H_i$ such that $x_1 \land x_2 \land ... \land x_m \leq \alpha$, then

$c_i(x_1 = \alpha) \lor c_i(x_2 = \alpha) \lor ... \lor c_i(x_m = \alpha)$

is called an $\alpha$-resolvent of $C_1, C_2, ..., C_m$, denoted by $R_{p_{i1}, ..., p_{im}}(c_i(x_1), c_i(x_2), ..., c_i(x_m)), x_1, x_2, ..., x_m$ are called an $\alpha$-resolution group.

**Definition 10.** [22] Let $C$ be a generalized clause in lattice-valued propositional logic $LP(X)$. $C$ is called a locked generalized clause if each disjunct occurring in $C$ is assigned a positive integer in its lower left corner (the same disjunct appearing in different locations can be labeled different positive integer). The positive integer is called a lock of the disjunct.

### 2.3. Lattice-valued first-order logic $LF(X)$

**Definition 11.** [17] Suppose $V$ and $F$ are the set of variable symbols and that of functional symbols in lattice-valued first-order logic $LF(X)$, respectively, the set of terms of $LF(X)$ is defined as the smallest set $\mathcal{T}$ satisfying the following conditions:

1. $V \subseteq \mathcal{T}$,
2. for any $n \in N$, if $f^{(n)} \in F$, then for any $t_0, t_1, ..., t_n \in \mathcal{T}$, $f^{(n)}(t_0, t_1, ..., t_n) \in \mathcal{T}$.

**Remark 1.** $f^{(0)}$ is specified as a constant symbol.

**Definition 12.** [17] Suppose $P$ is the predicate symbol set in lattice-valued first-order logic $LF(X)$. The set of atoms of $LF(X)$ is defined as the smallest set $\mathcal{A}$ satisfying the following condition:

For any $n \in N$, if $P^{(n)} \in P$, then $P^{(n)}(t_0, t_1, ..., t_n) \in \mathcal{A}$, for any $t_0, t_1, ..., t_n \in \mathcal{T}$.

**Remark 2.** $P^{(0)}$ is specified as a certain element in $L$. 
Definition 13. [17] The set of formulas of lattice-valued first-order logic $LF(X)$ is defined as the smallest set $\mathcal{F}$ satisfying the following conditions:

1. $\mathcal{F} \subseteq \mathcal{F}$
2. If $p, q \in \mathcal{F}$, then $p \land q \in \mathcal{F}$
3. If $p \in \mathcal{F}$ is a free variable in $p$, then $(\forall x)p$, $(\exists x)p$.

Remark 3. Note that $p \equiv p \lor O, p \lor q = (p \rightarrow q) \rightarrow q$, $p \land q = (p \lor q)$: $p \leftrightarrow q = (p \rightarrow q) \land (q \rightarrow p)$. Therefore, if $p, q \in \mathcal{F}$, then $p, p \lor q, p \land q, p \leftrightarrow q \in \mathcal{F}$.

Definition 14. [17] Suppose $G \in \mathcal{F}$, $F_G$ is the set of all functional symbols occurring in $G$, $P_G$ is the set of all predicate symbols occurring in $G$, and $D (\neq \phi)$ is the domain of interpretation. An interpretation of $G$ over $D$ is a tuple $I_D = (D, \mu_D, v_0)$, where,

- $\mu_D: F_G \rightarrow U_D = \{ f_D^{(a)}: D^a \rightarrow D \mid n \in N \}$,
- $f_D^{(0)} \mapsto f_D^{(0)}$, $f_D^{(0)}(D^0) = \{ f_D^{(0)} \} \subseteq D, D^{(0)}$ is a non-empty set,
- $f_D^{(0)} \mapsto f_D^{(n)} (n \in N^+)$,
- $v_D: P_G \rightarrow V_D = \{ P_D^{(a)}: D^a \rightarrow L \mid n \in N \}$,
- $p_D^{(0)} \mapsto p_D^{(0)}$, $p_D^{(0)}(D^0) = \{ P_D^{(0)} \} \subseteq L$,
- $p_D^{(0)} \mapsto p_D^{(n)} (n \in N^+)$.

In lattice-valued first-order logic $LF(X)$, the definitions of generalized literal and generalized clause are similar to those in lattice-valued propositional logic $LP(X)$.

Definition 15. [17] Let $G \in \mathcal{F}$, $\alpha \in L$. If $v_0(G) \leq \alpha$ for any interpretation $I_D = (D, \mu_D, v_0)$ in lattice-valued first-order logic $LF(X)$, $G$ is said to be $\alpha$-false, denoted by $G \leq \alpha$.

In the following, the definitions of substitution, renamed substitution, ground substitution, instance, ground instance are the same as those in classical logic.

Definition 16. [23] Let $C_j = p_1 \lor \ldots \lor p_m$ be generalized clauses without common variables in $LF(X)$, $H_i = \{ p_1, \ldots, p_m \}$ the set of all disjuncts occurring in $C_j$, $i = 1, 2, \ldots, m$, $\alpha \in L$. If there exist generalized literals $x_i \in H_i$ and a substitution $\sigma$ such that $x_i^\sigma \land x_j^\sigma \land \ldots \land x_m^\sigma \leq \alpha$, then

$C_1^\sigma(x_1^\sigma = \alpha) \lor C_2^\sigma(x_2^\sigma = \alpha) \lor \ldots \lor C_m^\sigma(x_m^\sigma = \alpha)$

is called an $\alpha$-resolvent of $C_1, C_2, \ldots, C_m$ denoted by $R_{\alpha-G}(C_i(x_i))$. $C_1(x_1), \ldots, C_m(x_m)$, $x_1, x_2, \ldots, x_m$ are called an $\alpha$-resolution group.

$\alpha$ occurring in the following is always less than $I$.

3. $\alpha$-Quasi-Lock Semantic Resolution for Lattice-Valued Propositional Logic $LP(X)$

Definition 17. Let $v_0$ be a valuation in lattice-valued propositional logic $LP(X)$, $\alpha \in L$. $E_1, \ldots, E_q$ are sets composed of some locked generalized clauses in $LP(X)$. The sequence $(N, E_1, \ldots, E_q)$ is called an $\alpha$-quasi-lock semantic clash ($\alpha$-QLS clash for short) w.r.t. $v_0$. If $N, E_1, \ldots, E_q$ satisfy the following conditions:

1. For any generalized clause $C_i \in E_i$, $v_0(C_i) \leq \alpha$, where $i = 1, 2, \ldots, q$.
2. For any disjunction $D_i = \vee C_i \in E_i$, $v_0(D_i)$ is an $\alpha$-false literal. $R_0$ is called the $\alpha$-QLS resolvent of this clash. $E_1, \ldots, E_q$ are called electrons and $N$ is called the core of this clash.

Remark 4. (1) For any generalized clause $C$ occurring in Definition 17, if there exists the same disjunct occurring in different locations of $C$, then retain the one with the smallest lock and delete others. For example, let $C = 4g \lor 5g \lor 4g \lor 8h \lor 8h$ be a locked generalized clause, if $g$ and $h$ are different disjuncts, then we rewrite $C$ as $4g \lor 8h$.

(2) For any disjunction $h_i$, occurring in $E_i$, $v_0(h_i) \leq \alpha$, $i = 1, 2, \ldots, q$. In fact, let generalized clause $C_i \in E_i$...
and $C_i = C_i' \lor h_i$; if $v_0(h_i) \leq \alpha$, then $v_0(C_i) \leq \alpha$, which is a contradict to $v_0(C_i) \leq \alpha$.

(3) For any generalized clause $C \in N$, $v_0(C) \leq \alpha$. In fact, if there exists a generalized clause $C' \in N$ such that $v_0(C') \leq \alpha$, then there is no $\alpha$-resolution literal occurring in $C'$ by condition (4) of Definition 17, which means there is no $\alpha$-QLS clash.

(4) Since the $\alpha$-QLS resolvent $R_q$ satisfying $v_0(R_q) \leq \alpha$, to obtain the $\alpha$-QLS resolvent as soon as possible, the previous $\alpha$-QLS resolvent must be involved in the next resolution, i.e., for the $i$-th resolution, $R_{i-1} \in N_i$. As the generalized clauses $C_1, C_2, \ldots, C_k$, $C_i \in \{C_1, C_2, \ldots, C_k\}$ and $R_{i-1}$ constitute resolution generalized clauses) not only occur in $E_i$, but also in $N_i, N_{i+1}, \ldots, N_{i+k}$, hence $N_i = \{R_{i-1}\} \cup N_{i+1}, N_{i+2}, \ldots, N_{i+k}.

(5) If $i=2$, then let $N_i' = A_1 \cup A_2$ and $A_i = \phi$. Otherwise, let $N_i' = A_1 \cup A_2, i = 3, \ldots, q$. The construction of $A_1$ and $A_2$ is as follows:

Step 1: Let $A' = \{R \mid R \in \{R_1, \ldots, R_q\}\}$. For any $R \in A'$, $R'$ satisfies the following conditions:

1. the $\alpha$-resolution literal of $R_{i-1}$ does not occur in $R'$,

2. the $\alpha$-resolution literals of $R_{i-2}$ and $R'$ do not come from the same generalized clause of $N_i$.

Step 2: For any generalized clause $D \in A'$, if there exists $D' \in A'$ such that each disjoint of $D$ is a disjoint of $D'$, then delete $D'$.

Step 3: After step 2, we can obtain a set, denoted by $A_1$.

Step 4: Let $A_2 = \{C \mid C \in N_i\}$. For any $C' \in A_2$, $C'$ satisfies the following conditions:

1. the $\alpha$-resolution literal of $R_{i-1}$ does not come from $C'$,

2. for any generalized clause $E \in A_1$, the $\alpha$-resolution literal of $E$ does not come from $C'$.

Example 3. Let $C_1 = (x \rightarrow y)', C_2 = \lnot (x \rightarrow y) \lor (y \rightarrow z)' \lor (s \rightarrow a_3)', C_3 = (s \rightarrow a_3) \lor (r \rightarrow t)$ be three locked generalized clauses in lattice-valued propositional logic $\mathcal{L}(X)$ based on $\mathcal{L}(X)$ and $S = C_1 \wedge C_2 \wedge C_3 \wedge C_4$, where $\mathcal{L}(X)$ is the Lukasiewicz implication algebra with nine elements, $x, y, z, r, s, t$ are propositional variables, $a_3, a_1 \in L_0$. Suppose $\alpha = a_1 \in L_0, v_0$ is the valuation of $\mathcal{L}(X)$ such that $v_0(x) = I, v_0(y) = 0, v_0(z) = 0, v_0(s) = I, v_0(r) = a_1, v_0(t) = a_2$; then we can obtain an $\alpha$-QLS clash ($N, E_1, \ldots, E_q$) by Definition 17.

In fact, since $v_0(C_1) = a_3 < \alpha$, $v_0(C_2) = a_3 > \alpha$, $v_0(C_3) = a_4 < \alpha$, so we can obtain an $\alpha$-QLS clash (w.r.t. $v_0$) ($N, E_1, E_2, E_3$): $N = \{C_1, C_2, C_3, C_4\}$, $E_1 = \{C_1\}$, $E_2 = \{C_1\}$, $E_3 = \{C_1\}$ and the $\alpha$-QLS resolvent $R_3$ of this clash is $\delta(r \rightarrow t) \lor \alpha$, where $R_1 = (y \rightarrow z)' \lor (s \rightarrow a_3)' \lor \alpha$, $N_3 = \{R_3\}$.

If we let $E_1 = \{C_1\}, E_2 = E_3 = \{C_1\}$, then there is no $\alpha$-QLS clash. Therefore, we can obtain the fact that $\alpha$-QLS clash is affected by the order of electrons.

Example 4. Let $C_1 = (x \rightarrow y), C_2 = \lnot (x \rightarrow z)' \lor \delta(s \rightarrow t)$, $C_3 = \lnot y' \lor \delta(y \rightarrow z), C_4 = \delta(s \rightarrow t)' \lor (r \rightarrow (a_2, b_1))$ be four locked generalized clauses in lattice-valued propositional logic $\mathcal{L}(X)$ based on $\mathcal{L}(X)$ and $S = C_1 \wedge C_2 \wedge C_3 \wedge C_4$, where $\mathcal{L}(X)$ is the same LIA with eighteen elements as that in Example 2. $x, y, z, r, s, t$ are propositional variables, $(a_2, b_1) \in L_0 \times L_2$. Suppose $\alpha = (a_2, b_1) \in L_0 \times L_2$, $v_0$ is the valuation of $\mathcal{L}(X)$ such that $v_0(x) = I, v_0(y) = v_0(z) = (a_2, b_1), v_0(s) = (a_2, b_1)$, $v_0(t) = (a_2, b_1), v_0(r) = (a_1, b_1)$, then we can obtain an $\alpha$-QLS clash ($N, E_1, \ldots, E_q$) by Definition 17.

In fact, since $v_0(C_1) = (a_2, b_1) < \alpha, v_0(C_3) = (a_2, b_1) / \alpha$ (here // means incomparable), $v_0(C_1) = I > \alpha, v_0(C_2) = (a_2, b_1) / \alpha$ so we can obtain an $\alpha$-QLS clash (w.r.t. $v_0$) ($N, E_1, E_2, E_3$): $N = \{C_2, C_3, C_4\}$, $E_1 = \{C_1\}$, $E_2 = \{C_1\}$, $E_3 = \{C_4\}$ and the $\alpha$-QLS resolvent $R_3$ of this clash is $\delta(r \rightarrow (a_2, b_1)) \lor \alpha$, where $R_1 = (y \rightarrow z) \lor \delta(s \rightarrow t) \lor \alpha, N_3 = \{R_3\}$.

Definition 18. Suppose $S = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where $C_1, C_2, \ldots, C_m$ are locked generalized clauses in lattice-valued propositional logic $LP(X), v_0$ is a valuation in $LP(X)$ and $\alpha \in L$. $\{\Phi_1, \Phi_2, \ldots, \Phi_q\}$ is called an $\alpha$-quasi-lock semantic resolution deduction (w.r.t. $v_0$) (or $\alpha$-QLS resolution deduction for short) from $S$ to generalized clause $\Phi_q$, if it satisfies the following conditions:

1. $\Phi_q$ is a generalized clause occurring in $S$ or
2. $\Phi_q$ is an $\alpha$-QLS resolvent, where the core and electrons of $\Phi_q$ are composed of $\Phi_i (i < q)$ or generalized clauses occurring in $S$.

Theorem 1. (Soundness) Suppose $S = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where $C_1, C_2, \ldots, C_m$ are locked generalized clauses in lattice-valued propositional logic $LP(X), \alpha \in L$. $v_0$ is a valuation in $LP(X)$ and $\{\Phi_1, \Phi_2, \ldots, \Phi_q\}$ is an $\alpha$-QLS resolution deduction (w.r.t. $v_0$) from $S$ to generalized clause $\Phi_q$. Co-published by Atlantis Press and Taylor & Francis Copyright: the authors
Proof. According to the soundness of the general form of \(\alpha\)-resolution principle in lattice-valued propositional logic \(LP(X)\) [23], we can obtain the result easily. \(\square\)

Theorem 2. (Conditional Completeness) Let \(S = C_1 \land C_2 \land \ldots \land C_m\), where \(C_1, C_2, \ldots, C_m\) are locked generalized clauses in lattice-valued propositional logic \(LP(X)\). \(v_0\) is a valuation of \(LP(X)\) and \(\alpha \in L\). If the following conditions hold:

1. \(S \leq \alpha\),
2. \(S^0 \neq \phi\), where \(S^0 = \{C_i \mid v_0(C_i) \leq \alpha, i \in \{1, 2, \ldots, m\}\}\),
3. there exists at least a locked generalized clause \(C_j \in \{C_1, C_2, \ldots, C_m\}\), such that for any disjunct \(g\) of \(C_j\), \(v_0(g) \leq \alpha\),

then there exists an \(\alpha\)-QLS resolution deduction (with respect to \(v_0\)) from \(S\) to an \(\alpha\)-false generalized clause.

Proof. Suppose \(H_j\) is the set of all disjuncts occurring in \(C_i\) and \([H_i] = w_i, i = 1, 2, \ldots, m\). Let \(K(S)\) be equal to the number of disjuncts occurring in \(S\) minus that of generalized clauses occurring in \(S\), i.e., \(K(S) = \sum_{i=1}^{m} w_i - m\).

We have the following two cases.

Case 1: If \(K(S) = 0\), then \(S\) is composed of unit generalized clauses, i.e., each generalized clause occurring in \(S\) includes only one generalized literal. Since \(S \leq \alpha\), so all generalized literals occurring in \(S\) compose an \(\alpha\)-resolution group. As the condition (2) of Theorem 2 holds, so we have \(S = S^0 \cup S^\Delta\) and \(S^0, S^\Delta \neq \phi\), where \(S^0 = \{C_i \mid C_i\) is a generalized clause occurring in \(S, v_0(C_i) \leq \alpha\}\), \(S^\Delta = \{C_i \mid C_i\) is a generalized clause occurring in \(S, v_0(C_i) \leq 1\}\). Let \(N = S^0, E = S^\Delta\), then \((N, E)\) is an \(\alpha\)-QLS clash and its \(\alpha\)-QLS resolvent is an \(\alpha\)-false generalized clause. Therefore the result holds.

Case 2: Suppose the result holds for \(K(S) < n, n > 0\). Now we need to prove the result for \(K(S) = n\).

1. Let \(K(S) = n\), so \(S\) has at least one non-unit generalized clause. Suppose \(t\) is the largest lock of \(\alpha\)-false disjuncts (under \(v_0\)) occurring in non-unit generalized clauses of \(S\). Let \(C_i = C_i^* \lor g\), where \(C_i^*\) is non-empty and \(v_0(g) \leq \alpha\).

Suppose \(S_1 = C_1 \land \ldots \land C_{t-1} \land C_t^* \land C_{t+1} \land \ldots \land C_m\), so \(S_1 \leq \alpha\) and \(K(S_1) < n\). According to induction hypothesis, there exists an \(\alpha\)-QLS resolution deduction \(D_t^*\) from \(S_1\) to an \(\alpha\)-false generalized clause.

2. If \(v_0(C_i^*) \leq \alpha\), then \(v_0(C_i) \leq \alpha\). In each \(\alpha\)-QLS clash \((N, E_1^*, \ldots, E_q^*)\) of \(D_t^*\), \(C_i^*\) can only be an element of electrons. If there exists \(k \in \{1, 2, \ldots, q\}\) such that \(C_i^* \in E_k^*\), then replace \(C_i^*\) with \(C_i\). Since the lock \(t\) of disjunct \(g\) is bigger than or equal to any other lock occurring in generalized clause \(C_i\), so the \(\alpha\)-QLS resolvent \(R_k\) of \(E_k\) and \(N_k^*\) is equal to \(R_k^* \lor g\), where \(E_k\) is the set obtained by replacing \(C_i^*\) occurring in \(E_k^*\) with \(C_i\), \(R_k^*\) is the \(\alpha\)-QLS resolvent of \(E_k^*\) and \(N_k^*\). Since \(v_0(g) \leq \alpha\), so \(g\) can not be the \(\alpha\)-resolution generalized literal of \(R_k(t = k, k+1, \ldots, q)\). Hence, after changing \(C_i^*\) to \(C_i\), the sequence \((N', E_1^*, \ldots, E_{t-1}^*, E_t^*, E_{t+1}^*, \ldots, E_q^*)\) is also an \(\alpha\)-QLS clash and its \(\alpha\)-QLS resolvent equals to \(R_k^* \lor g\).

Since disjuncts of an \(\alpha\)-false \(\alpha\)-QLS resolvent (under \(v_0\)) are composed of the following two parts:

(i) disjuncts occur in the core and are \(\alpha\)-false under \(v_0\),

(ii) disjuncts occur in non-unit generalized clauses of electrons and are not \(\alpha\)-resolution literals, so, in each \(\alpha\)-QLS clash \((N', E_1^*, \ldots, E_q^*)\) of \(D_t^*\), if there exists \(k \in \{1, 2, \ldots, q\}\) such that \(R_k \in E_k^*\) is an original \(\alpha\)-QLS resolvent, which is generated by the clash with \(C_i^*\) as an element of electrons, then after changing \(R_k\) to \(R_k^* \lor g\), the sequence \((N', E_1^*, \ldots, E_{t-1}^*, E_t^*, E_{t+1}^*, \ldots, E_q^*)\) is also an \(\alpha\)-QLS clash and its \(\alpha\)-QLS resolvent is equal to \(R_k^* \lor g\), where \(E_k\) is the set obtained by replacing \(R_k^*\) occurring in \(E_k^*\) with \(R_k^* \lor g\), and \(R_k^*\) is the \(\alpha\)-QLS resolvent of clash \((N', E_1^*, \ldots, E_q^*)\).

b. If \(v_0(C_i^*) \leq \alpha\), then \(C_i\) can only be an element occurring in the core of each \(\alpha\)-QLS clash \((N', E_1^*, \ldots, E_q^*)\) of \(D_t^*\). Since \(g\) is not the \(\alpha\)-resolution literal of each generalized clause occurring in the core \(N'\), so \((N, E_1^*, \ldots, E_q^*)\) is also an \(\alpha\)-QLS clash and its \(\alpha\)-QLS resolvent is equal to \(R_k^* \lor g\), where \(N\) is the set obtained by replacing \(C_i\) occurring in \(N'\) with \(C_i^* \lor g\), and \(R_k^*\) is the \(\alpha\)-QLS resolvent of clash \((N', E_1^*, \ldots, E_q^*)\).

Hence, after changing all \(C_i^*\) occurring in each \(\alpha\)-QLS clash of \(D_t^*\) to \(C_i\) and modifying the corresponding \(\alpha\)-QLS resolvent, we can obtain a resolution deduction \(D_t\). From the above discussion, we can get that \(D_t\) is an \(\alpha\)-QLS resolution deduction from \(S\) to an \(\alpha\)-false generalized clause or \(g\).
If \( D_1 \) is an \( \alpha \)-QLS resolution deduction from \( S \) to an \( \alpha \)-false generalized clause, then the result holds.

If \( D_1 \) is an \( \alpha \)-QLS resolution deduction from \( S \) to \( \gamma \), then let \( S_2 = \{ a_1 \wedge \ldots \wedge a_n \wedge \gamma \wedge \phi \} \). Obviously, \( S_2 \subseteq \alpha \) and \( \mathcal{L}(S_2) < \eta \). According to induction hypothesis, there exists an \( \alpha \)-QLS resolution deduction \( D_2 \) from \( S_2 \) to an \( \alpha \)-false generalized clause. Connecting \( D_1 \) and \( D_2 \), we can obtain an \( \alpha \)-QLS resolution deduction from \( S \) to an \( \alpha \)-false generalized clause.

2) If \( t \) occurring in \( l \) does not exist, then disjuncts occurring in non-unit generalized clauses of \( S \) are not \( \alpha \)-false under \( v_0 \). As \( S' \neq \emptyset \), so for any \( G \in S' \), \( G \) is a unit generalized clause. Since \( S \subseteq \alpha \), for any \( g_1, g_2, \ldots, g_m \in H_1 \times H_2 \times \ldots \times H_n \), we have \( g_1 \wedge g_2 \wedge \ldots \wedge g_m \subseteq \alpha \) and \( \mathcal{L}(S) \leq \mathcal{S}' \). Let \( N = \{ C_1, C_2, \ldots, C_p \} \subseteq \mathcal{S}' \). According to Definition 17, there exists an \( \alpha \)-QLS clash (w.r.t. \( v_0 \)) \((N, E_1, \ldots, E_p)\), and its \( \alpha \)-QLS resolvent is an \( \alpha \)-false generalized clause, where \( E_s \subseteq \mathcal{S}' \), \( s = 1, 2, \ldots, q \). Hence, the result holds. \( \square \)

Example 5. Let \( C_1 = (x \to y) \), \( C_2 = (x \to y' \vee (s \to t)) \), \( C_3 = y' \vee (y \to z) \vee (s \to (a_1, b_1)) \), \( C_4 = (s \to t') \vee (r \to (a_2, b_2))' \), \( C_5 = r \to (a_3, b_3) \) be five generalized clauses in lattice-valued propositional logic \((\mathcal{S}' \times \mathcal{S})\mathcal{P}(X)\) and \( S = C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \), where \( (a_1, b_1), (a_2, b_2), (a_3, b_3) \) occurring in \( 1 \) does not exist, then disjuncts \( C_{\alpha} \) and \( g \) are bound variables with the quantifier \( \forall \). For any generalized clauses \( C_1, C_2, \ldots, C_m \) in lattice-valued first-order logic \( \mathcal{L}(X) \), there exists at least a renamed substitution \( \varphi \) such that \( C_{\varphi}, C_{\varphi}, \ldots, C_{\varphi} \wedge C_{\varphi} \wedge C_{\varphi} \) have no common variables. Therefore, generalized clauses \( C_1, C_2, \ldots, C_m \) occurring in the following always have no common variables.

4. \( \alpha \)-Quasi-Lock Semantic Resolution for Lattice-Valued First-Order Logic \( \mathcal{L}(X) \)

Generalized clauses and generalized literals occurring in this section always belong to a generalized-Skolem standard form, i.e., for any generalized clause \( C \) and generalized literal \( g \), all variables of \( C \) and \( g \) are bound variables with the quantifier \( \forall \). For any generalized clauses \( C_1, C_2, \ldots, C_m \) in lattice-valued first-order logic \( \mathcal{L}(X) \), there exists at least a renamed substitution \( \varphi \) such that \( C_{\varphi}, C_{\varphi}, \ldots, C_{\varphi} \wedge C_{\varphi} \wedge C_{\varphi} \) have no common variables.
there exist instances \( g^{01}, g^{02} \) of \( g \) such that \( \nu_0(g^{01}) \leq \alpha \) and \( \nu_0(g^{02}) \leq \alpha \). Both \( \alpha \)-pure-false generalized literal and \( \alpha \)-para-false generalized literal w.r.t. \( I_D \) are called \( \alpha \)-false generalized literal w.r.t. \( I_D \).

**Definition 20.** Let \( I_D = (D, \mu_D, \nu_D) \) be an interpretation in lattice-valued first-order logic \( LF(X) \), \( \alpha \in L, N, E_1, ..., E_q \) are sets composed of some locked generalized clauses in \( LF(X) \). The sequence \((N, E_1, ..., E_q)\) is called an \( \alpha \)-quasi-lock semantic clash (\( \alpha \)-QLS clash for short) w.r.t. \( I_D \), if \( N, E_1, ..., E_q \) satisfy the following conditions:

1. For any generalized clause \( C_i \in E_i, \nu_D(C_i) \leq \alpha, i = 1, 2, ..., q, \)
2. Let \( R_0 = \bigvee C_i \). For any \( i = 1, 2, ..., q \), there exists an \( \alpha \)-resolution \( R_0 \) of \( N \) and \( E_i \), where \( N_i = N, N_2 = \{R_1\} \cup N_1 \), \( N_3 \subseteq N \) for any \( i = 3, ..., q \), \( N_i = \{R_{i-1}\} \cup N_2 \), \( N_j \subseteq N \cup \{R_{j-1}, R_{j-2}\} \).
3. For any generalized clause \( C_j \in E_j \) the \( \alpha \)-resolution literal \( g_j \) of \( C_j \) is the one that has the smallest lock among disjuncts occurring in \( C_j \), \( j = 1, 2, ..., q \).
4. For any generalized clause \( C_h \in N_p \) the \( \alpha \)-resolution literal \( g_h \) of \( C_h \) is the one which not only has at least a non-\( \alpha \)-false instance (w.r.t. \( I_D \)), and also has the smallest lock among disjuncts with non-\( \alpha \)-false instances (w.r.t. \( I_D \)) occurring in \( C_h \), \( j = 1, 2, ..., q \).
5. \( \nu_D(R_0) \leq \alpha \).

\( R_0 \) is called the \( \alpha \)-QLS resolvent of this clash. \( N \) is called the core and \( E_1, ..., E_q \) are called electrons of this clash.

**Remark 5.** (1) For any generalized clause \( C \) occurring in Definition 20, if there exists the same disjunct occurring in different locations of \( C \), then retain the one with the smallest lock and delete others.

2. For any disjunction \( h \) occurring in \( E_i \), \( \nu_0(h) \leq \alpha, i = 1, 2, ..., q \).

3. In general, there exists at least a generalized clause \( C \in N \) such that \( \nu_0(C) \leq \alpha \). In fact, if for any generalized clause \( C \in N, \nu_0(C) \leq \alpha \), then \( \nu_0(R_0) \leq \alpha \), i.e., there does not exist an \( \alpha \)-QLS clash. If \( R_0 \) is seen as an \( \alpha \)-QLS resolvent, then this \( \alpha \)-QLS clash is redundant by the construction of \( R_0 \).

**Example 6.** \( C_1 = (M(x_1) \rightarrow N(x_1))', C_2 = (M(b) \rightarrow N(y_1)) \vee (P(y_2)), C_3 = (Q(z_1) \rightarrow P(c))' \vee (R(z_2) \rightarrow S(c))' \) be four locked generalized clauses in lattice-valued first-order logic \( LF(X) \) based on \( \sim \) and \( S = C_1 \wedge C_2 \wedge C_3 \wedge C_4 \) where \( x_1, x_2, y_1, y_2, z_1, z_2 \) are variables and \( b, c \) are constants. Suppose \( \alpha = \alpha_1 \) and \( I_D = (D, \mu_D, \nu_D) \) is an interpretation of \( LF(X) \), where \( D = \{b, c\} \),

\[
\begin{array}{cccccccc}
  b & c & M(b) & M(c) & N(b) & N(c) & P(b) \\
  b' & c' & M(c) & M(b) & N(c) & N(b) & P(b) \\
\end{array}
\]

Then we can obtain an \( \alpha \)-QLS clash \((N, E_1, ..., E_q)\) by Definition 20.

In fact, since \( \nu_D(C_1) < \alpha \), \( \nu(D, C_2) > \alpha \), \( \nu(C_3) < \alpha \), so we can obtain an \( \alpha \)-QLS clash \((N, E_1, E_2)\): \( N = \{C_2\}, E_1 = \{C_1\}, E_2 = \{C_1\} \) and the \( \alpha \)-QLS resolvent \( R_2 \) of this clash is \((S(R_2) \rightarrow S(c))' \vee \alpha \), where \( R_1 = \{M(b) \rightarrow N(y_1) \vee (Q\rightarrow S(c))' \} \). If \( \nu_0(R_0) \leq \alpha \).

**Example 7.** \( C_1 = (M(x_1) \rightarrow N(a))' \vee (P(a) \rightarrow Q(v_3)), C_2 = (M(b) \rightarrow R(y_1))' \vee (S(c) \rightarrow T(y_2)), C_3 = (N(z_1) \rightarrow R(d)), C_4 = (S(u_1) \rightarrow T(d))' \) be four locked generalized clauses in lattice-valued first-order logic \( LF(X) \) based on \( LIA \), and \( S = C_1 \wedge C_2 \wedge C_3 \wedge C_4 \) where \( x_1, x_2, y_1, y_2, z_1, u_1 \) are variables and \( a, b, c, d \) are constants. Suppose \( \alpha = (a_0, b_0, c_0, d_0) \) is an interpretation of \( LF(X) \), where \( D = \{a, b, c, d\} \),

\[
\begin{array}{cccccccc}
  a & b & c & d & M(a) & M(b) & M(c) & M(d) \\
  a' & b' & c' & d' & M(a) & M(b) & M(c) & M(d) \\
\end{array}
\]

Then we can obtain an \( \alpha \)-QLS clash \((N, E_1, ..., E_q)\) by Definition 20.
In fact, since \( \nu_2(C_1) < \alpha, \nu_2(C_2) \parallel \alpha, \nu(C_3) > \alpha, \nu(C_4) < \alpha \), so we can obtain an \( \alpha \)-QLS clash (w.r.t. \( I_D \)) (N, E1, E2): \( N = \{ C_2, C_3 \}, E_1 = \{ C_1 \}, E_2 = \{ C_4 \} \) and the \( \alpha \)-QLS resolvent \( R_2 \) of this clash is \( \exists (P(a) \rightarrow Q(x_2)) \lor \alpha \), where \( R_1 = \{ S(c) \rightarrow T(v_3) \} \lor \exists (P(a) \rightarrow Q(x_3)) \lor \alpha, N_2 = \{ R_1 \} \).

**Definition 21.** Suppose \( S = C_1 \wedge C_2 \wedge \ldots \wedge C_m \) where \( C_1, C_2, \ldots, C_m \) are locked generalized clauses in lattice-valued first-order logic \( LF(X) \), \( \alpha \in L \) and \( I_D = (D, \mu_0, v_D) \) is an interpretation in \( LF(X) \). \{ \Phi_1, \Phi_2, \ldots, \Phi_t \} is an \( \alpha \)-quasi-lock semantic resolution deduction (w.r.t. \( I_D \)) (\( \alpha \)-QLS resolvent deduction for short) from \( S \) to generalized clause \( \Phi_s \) if it satisfies the following conditions:

1. \( \Phi_s \) is a generalized clause occurring in \( S \) or
2. \( \Phi_s \) is an \( \alpha \)-QLS resolvent (w.r.t. \( I_D \)), where the core and electrons of \( \Phi_s \) are composed of \( \Phi_i \) (\( i < \alpha \)) or generalized clauses occurring in \( S \).

**Theorem 3. (Soundness)** Suppose \( S = C_1 \wedge C_2 \wedge \ldots \wedge C_m \) where \( C_1, C_2, \ldots, C_m \) are locked generalized clauses in lattice-valued first-order logic \( LF(X) \), \( \alpha \in L \), \{ \Phi_1, \Phi_2, \ldots, \Phi_t \} is an \( \alpha \)-QLS resolvent deduction from \( S \) to generalized clause \( \Phi_s \). If \( \Phi_s \) is \( \alpha \)-false generalized clause, then \( S \models \alpha \), i.e., if \( \Phi_s \models \alpha \), then \( S \models \alpha \).

**Proof.** According to the soundness of the general form of \( \alpha \)-resolution principle in \( LF(X) \)[23], we can obtain the result easily. \( \square \)

**Theorem 4. (Lifting Lemma)** Let \( N, E_1, \ldots, E_q \) be sets composed of locked generalized clauses in lattice-valued first-order logic \( LF(X) \), \( I_D = (D', \mu_a, v_D) \) an interpretation in \( LF(X) \) and \( \alpha \in L \). \( N^* \) is the set obtained by replacing each locked generalized clause \( C \) occurring in \( N \) with an instance \( C^* \) of \( C \), and \( E_i^* \) is the set obtained by replacing each locked generalized clause \( C \) occurring in \( E_i \) with an instance \( C_i^* \) of \( C \). If \( (N^*, E_1^*, \ldots, E_q^*) \) is an \( \alpha \)-QLS clash (w.r.t. \( I_D \)) and write its \( \alpha \)-QLS resolvent as \( R_q^* \), then there exists an interpretation \( I_D = (D, \mu_0, v_D) \) of \( LF(X) \) such that \( (N, E_1, \ldots, E_q) \) is an \( \alpha \)-QLS clash (w.r.t. \( I_D \)), and \( R_q^* \) is an instance of \( R_q \), where \( R_q \) is the \( \alpha \)-QLS resolvent of \( (N, E_1, \ldots, E_q) \), i.e., Fig. 1 holds.

**Fig. 1 Transformation Diagram**

**Remark 6.** In fact, \( I_D = (D, \mu_0, v_D) \) and \( I_D^* = (D', \mu_0, v_D) \) satisfy the following conditions:

1. \( D = D' \).
2. For any function symbol \( f^{(0)} \) occurring in \( N \), \( E_1, \ldots, E_q \), \( f_D^{(0)} = f_D^{(0)} \).
3. Let \( (d_1, \ldots, d_n) \in D'(n \in N^*) \). For any \( n \)-ary function symbol \( f^{(n)} \) occurring in \( N, E_1, \ldots, E_q \):
   1. if \( f^{(n)}(d_1, \ldots, d_n) \) does not occur in \( I_D \), then \( \mu_D(f^{(n)}(d_1, \ldots, d_n)) = \mu_D(f^{(n)}(d_1, \ldots, d_n)) \), where \( (d_1, \ldots, d_n) \in D' \).
   2. if \( f^{(n)}(d_1, \ldots, d_n) \) does not occur in \( I_D \), then \( \mu_D(f^{(n)}(d_1, \ldots, d_n)) = \mu_D(f^{(n)}(d_1, \ldots, d_n)) \), where \( (d_1, \ldots, d_n) \in D' \).
4. Let \( (d_1, \ldots, d_n) \in D'(n \in N^*) \). For any \( n \)-ary predicate symbol \( p^{(n)} \) occurring in \( N, E_1, \ldots, E_q \):
   1. if \( p^{(n)}(d_1, \ldots, d_n) \) does not occur in \( I_D \), then \( \nu_D(p^{(n)}(d_1, \ldots, d_n)) = \nu_D(p^{(n)}(d_1, \ldots, d_n)) \).
   2. if \( p^{(n)}(d_1, \ldots, d_n) \) does not occur in \( I_D \), then \( \nu_D(p^{(n)}(d_1, \ldots, d_n)) = \nu_D(p^{(n)}(d_1, \ldots, d_n)) \), where \( (d_1, \ldots, d_n) \in D' \).

**Proof.** Since for any generalized clauses \( C_j^*, G_{ih}^* \) occurring in \( N^* \) and \( E_i^* \) respectively, there exist ground substitutions \( \xi_j \) and \( \xi_h \) such that \( C_j^* = C_j^*, G_{ih}^* \) and generalized clauses occurring in \( N \cup E_1 \cup \ldots \cup E_q \) have no common variables with each other, we can obtain \( C_j^* = C_j^*, G_{ih}^* = G_{ih}^* \), where \( \xi = \{ \xi_1, \ldots, \xi_q \} \).
Let $S = C_1 \land C_2 \land \ldots \land C_m$, where $C_1, C_2, \ldots, C_m$ are locked general clauses in the lattice-valued first-order logic $\mathsf{LF}(X)$. $\alpha \in L$, $I_0 = (D, \mu_0, \nu_0)$ is an interpretation in $\mathsf{LF}(X)$ and for any disjunct $g$ occurring in $S$, $g$ is not an $\alpha$-para-false disjunct (w.r.t. $I_0$). If the following conditions hold:

1. $S \subseteq \alpha$,

2. $S^* \not\subseteq \phi$, where $S^* = \{C_i \mid \nu_0(C_i) \leq \alpha, i \in \{1, 2, \ldots, m\}\}$,

3. there exists at least a locked generalized clause $C_i$ occurring in $S$ such that $\nu_0(g) \not\equiv \alpha$ for any disjunct $g$ of $C_i$,

then there exists an $\alpha$-QLS resolution deduction (w.r.t. $I_0$) from $S$ to an $\alpha$-QLS generalized clause.

Proof. Since $S \subseteq \alpha$, according to Herbrand theorem [15], there exists a finite ground instance set $S'$ such that $S' \subseteq \alpha$, where $S'$ is the conjunction of all ground instances in $S$. As for any disjunct $g$ of $C_i$, $\nu_0(g) \not\equiv \alpha$, so for any disjunct $g^0$ of $C_0$, $\nu_0(g^0) \not\equiv \alpha$, where $C_0$ is a ground instance of $C_i$, $C_0 \subseteq S'$ and $I_0 = (D^*, \mu_0, \nu_0)$ is an interpretation in $\mathsf{LF}(X)$, $D' = D$, $\nu_0 = \nu_0$, $\mu_0$ adds the interpretation of the constants only occurring in $S'$ and not in $S$ based on $\mu_0$. Since $S \not\subseteq \phi$ and there is no $\alpha$-para-false disjunct (w.r.t. $I_0$) occurring in $S$, so for any generalized clause $C_i \in S'$, $\nu_0(C_i) \not\equiv \alpha$, where $C_0$ is a ground instance of $C_i$, $C_0 \subseteq S'$. According to Theorem 2, there exists an $\alpha$-QLS resolution deduction (w.r.t. $I_0$).
ω* from Sα to an α-false generalized clause. Moreover,
we can use Theorem 4 to lift ω* to an α-QLS
resolution deduction (w.r.t. Iℝ) from S to an α-false
generalized clause.

Example 8. Let C1 = (M(f(x1)) → N(x2)) ∨ (N(x3) → N(a)),
C2 = (M(f(b)) → P(y1)) ∨ (Q(y2) → R(c))', C3 =
(N(z1))' ∨ (N(a) → P(z2)) ∨ (S(z3) → T(d))', C4 = (Q(d) → R(u1)) ∨ (Q(u2) → R(c)), C5 = (T(v1) → W(a))' be
generalized clauses in lattice-valued first-order
logic ( (X × X) F(X) and S = C1 ∨ C2 ∨ C3 ∨ C4 ∨ C5,
where x1, x2, x3, y1, y2, z1, z2, z3, u1, u2, v1 are variables
and a, b, c, d are constants. If α = (a0, b0) ∈ LφL3 then
there exists an α-QLS resolution deduction from S to an
α-false generalized clause.

Equip S with the following locks:

C1 = (M(f(x1)) → N(x2)) ∨ (M(x3) → N(a))
C2 = (M(f(b)) → P(y1)) ∨ (Q(y2) → R(c))'
C3 = (N(z1))' ∨ (N(a) → P(z2)) ∨ (S(z3) → T(d))'
C4 = (Q(d) → R(u1)) ∨ (Q(u2) → R(c))
C5 = (T(v1) → W(a))'.

As there exists ground substitution θ = {b/x1, a/x2, f
(b)x2, c/y1, d/y2, a/z1, c/z2, c/u1, d/u2, d/v1} such that
C1θ = M(f(b)) → N(a), C2θ = (M(f(b)) → P(c))' ∨ (Q(d) → R(c))', C3θ = (N(a))' ∨ (N(a) → P(c))' ∨
(S(c) → T(d))', C4θ = (Q(d) → R(c)), C5θ = (T(d) →
W(a)'), i.e., Sθ = C1θ ∨ C2θ ∨ C3θ ∨ C4θ ∨ C5θ, according
to Theorem 4.3, we only need to prove that there exists
an α-QLS resolution deduction from Sα to an α-false
generalized clause.

Suppose I₂p = (D', μp, v₂p) is an interpretation of
((X × X) F(X), where D' = {a, b, c, d},

a b c d f(b) M(a) M(b) M(c) M(d)
abc f(b) a b c d a (a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
N(a) P(c) Q(d) R(c) S(c) T(d) W(a)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂) (a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)

Hence, v₂p(C1θ) / α, v₂p(C2θ) < α, v₂p(C3θ) > α,
v₂p(C4θ) / α, v₂p(C5θ) < α. Moreover, we can obtain the
following α-QLS resolution deduction ω*:

(1') (M(f(b)) → N(a))
(2') (M(f(b)) → P(c))' ∨ (Q(d) → R(c))'
(3') (N(a))' ∨ (N(a) → P(c)) ∨ (S(c) → T(d))'
(4') (Q(d) → R(c))
(5') (T(d) → W(a))'
(6') (Q(d) → R(c))' ∨ (N(a))' ∨ (S(c) → T(d))' ∨ α
by (1'), (2'), (3')
(7') (N(a))' ∨ (S(c) → T(d))' ∨ α
by (4'), (6')
(8') (Q(d) → R(c))' ∨ (S(c) → T(d))' ∨ α
by (1'), (2'), (7')
(9') (S(c) → T(d))' ∨ α
by (4'), (8')
(10') α
by (1'), (5'), (9')

Therefore, ω* is an α-QLS resolution deduction (w.r.t.
I₂p) from Sα to an α-false generalized clause.

Let Iₐ = (D, μₐ, vₐ) be an interpretation of
((X × X) F(X), where D = {a, b, c, d},

a b c d f(a) f(b) f(c) f(d)
abc (a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
M(a) M(b) M(c) M(d)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
N(a) N(b) N(c) N(d)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
P(a) P(b) P(c) P(d)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
Q(a) Q(b) Q(c) Q(d)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
R(a) R(b) R(c) R(d)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
S(a) S(b) S(c) S(d)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
T(a) T(b) T(c) T(d) W(a)
(a₁, b₁) (a₁, b₂) (a₂, b₁) (a₂, b₂)
O O O O O O O O O O
(a₁, b₁)

According to Theorem 4, after replacing C0θ (i =
1, 2, . . . , 5) occurring in ω* with Cₐ, we can obtain an α-
QLS resolution deduction (w.r.t. Iₐ) ω from S to an α-
false generalized clause as follows:

(1) (M(f(x1)) → N(x2)) ∨ (M(x3) → N(a))
(2) (M(f(b)) → P(y1))' ∨ (Q(y2) → R(c))'
(3) (N(z1))' ∨ (N(a) → P(z2)) ∨ (S(z3) → T(d))'
(4) (Q(d) → R(u1)) ∨ (Q(u2) → R(c))
(5) (T(v1) → W(a))'
(6) (Q(y2) → R(c))' ∨ (N(z1))' ∨ (S(z3) → T(d))' ∨ α
In fact, there are the following five α-QLS clashes \((N, E_1, \ldots, E_i)\) occurring in \(\omega\):

1. \(N^1_1 = \{C_1, C_3\}, E^1_1 = \{C^*_1\}\) and the α-QLS resolvent \(R_1^1\) of \((N^1_1, E^1_1)\) is \(\delta(Q(v_1) \rightarrow R(c))' \vee \delta(S(z_1))' \vee \delta(S(z_1) \rightarrow T(d))' \vee \alpha\).

2. \(N^2_1 = \{C_1, E^2_1 = \{R^1_1\}\) and the α-QLS resolvent \(R_1^2\) of \((N^2_1, E^2_1)\) is \(\delta(N(z_1))' \vee \delta(S(z_1)) \rightarrow T(d))' \vee \alpha\).

3. \(N^3_1 = \{C_1, E^3_1 = \{C^*_1, R^1_1\}\) and the α-QLS resolvent \(R_1^3\) of \((N^3_1, E^3_1)\) is \(\delta(Q(v_1) \rightarrow R(c))' \vee \delta(S(z_1)) \rightarrow T(d))' \vee \alpha\).

4. \(N^4_1 = \{C_1, E^4_1 = \{R^1_1\}\) and the α-QLS resolvent \(R_1^4\) of \((N^4_1, E^4_1)\) is \(\delta(S(z_1)) \rightarrow T(d))' \vee \alpha\).

5. \(N^5_1 = \{C_1, E^5_1 = \{C^*_1, R^4_1\}\) and the α-QLS resolvent \(R_1^5\) of \((N^5_1, E^5_1)\) is \(\alpha\).

**Remark 7.** The main difference between quasi-lock semantic resolution (QLS for short) resolution in classical logic and α-QLS resolution in lattice-valued logic are the following two aspects:

1. Electrons and core of QLS clash in classical logic are clauses, but electrons and core of α-QLS clash are sets composed of generalized clauses.

2. Each resolution pair of QLS clash are composed of two literals, but each α-Resolution group of α-QLS clash may include more than two generalized literals.

Because of the above difference, for some false clause sets in classical logic, which do not have the completeness of QLS resolution, may be α-QLS resolved into empty clause. For example:

**Example 9.** Let \(S = \{P(a), \neg P(x) \lor Q(y), \neg Q(b)\}\) be a clause set in classical logic, written as \(S = P(a) \lor \neg P(x) \lor Q(y) \lor Q(b)\). Obviously, \(S\) is false and equip \(S\) with locks as follows:

1. \(P(a)\),

2. \(\neg P(x) \lor Q(y)\).

(3) \(\neg Q(b)\).

Let the interpretation \(I = \{\neg P(a), P(b), Q(a), \neg Q(b)\}\).

In fact, we can obtain that (1), (2) are false under \(I\) and (3) is true under \(I\). Hence, only (3) is qualified to become the core. So we have the following two cases:

Case 1: According to QLS resolution in classical logic, we can obtain a QLS clash ((2), (3)) and the QLS resolvent of this QLS clash is \(\neg P(x)\). Since \(\neg P(x) \lor \neg Q(b)\) is not a resolution pair, so there is not other QLS clash. Hence, there does not exist a QLS resolution deduction form \(S\) to empty clause.

Case 2: According to α-QLS resolution, there exist two α-QLS clashes \((N, E_1, \ldots, E_i)\) as follows:

1. \(N^1_1 = \{3\}, E^1_1 = \{2\}\) and the α-QLS resolvent \(R_1^1\) of \((N^1_1, E^1_1)\) is \(\neg P(x)\).

2. \(N^1_1 = \{3\}, E^1_1 = \{1, R^1_1\}\) and the α-QLS resolvent \(R_1^2\) of \((N^1_1, E^1_1)\) is empty clause.

Therefore, there exists an α-QLS resolution deduction form \(S\) to empty clause.

In general, α-QLS resolution in lattice-valued first-order logic LF(X) can not be equivalently transformed into that for lattice-valued propositional logic LP(X), which means that the lifting lemma is usually not true. But we can obtain the conclusion under some special cases.

**5. Conclusions**

Combined with the benefits of lock resolution method and semantic resolution method in classical logic, α-quasi-lock semantic resolution method for a lattice-valued logic with truth-valued defined in a lattice-valued logical algebraic structure- lattice implication algebras (LIA) was discussed. Concretely, on the basis of the general form of α-resolution principle, α-quasi-lock semantic resolution method based on lattice-valued propositional logic LP(X) was established, and its soundness theorem and condition completeness theorem were proved. Secondly, the corresponding α-quasi-lock semantic resolution method in lattice-valued first-order logic LF(X) was proposed, its soundness and condition completeness were also established. This will become the theoretical foundation for automated reasoning in lattice-valued logic based on LIA. Meanwhile, this method can be used in areas such as automated theorem
proving, program verification, and engineering technologies.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of P. R. China (Grant No. 60875034, 61175055, 61100046); the project TIN2009-0828; Sichuan Key Technology Research and Development Program of China (Grant No. 2011FZ0051) and Wireless Administration of Ministry of Industry and Information Technology of China ([2011]146).

References

6. R. Reiter, Two results on ordering for resolution with merging and linear format, J. ACM. 18(4) (1971) 630-646.
22. X. X. He, Y. Xu and J. Liu, α-Satisfiability and α-Lock Resolution for a Lattice-Valued Logic LP(X), in Proc. the 5th International Conference on Hybrid Artificial Intelligence Systems, (San Sebastián, Spain, Jun.23-25, 2010), pp. 320-327.