3

Resolution Principle and Fuzzy Logic

Hashim Habiballa
University of Ostrava
Czech Republic

1. Introduction

Fuzzy Predicate Logic with Evaluated Syntax (FPL) (Novák, V.) is a well-studied and wide-used logic capable of expressing vagueness. It has a lot of applications based on robust theoretical background. It also requires an efficient formal proof theory. However the most widely applied resolution principle (Dukić, N.) brings syntactically several obstacles mainly arising from normal form transformation. FPL is associating with even harder problems when trying to use the resolution principle. Solutions to these obstacles based on the non-clausal resolution (Bachmair, L.) were already proposed in (Habiballa, H.).

In this article it would be presented a natural integration of these two formal logical systems into fully functioning inference system with effective proof search strategies. It leads to the refutational resolution theorem prover for FPL ($RRTP_{FPL}$). Another issue addressed in the paper concerns to the efficiency of presented inference strategies developed originally for the proving system. It is showed their perspectives in combination with standard proof-search strategies. The main problem for the fuzzy logic theorem proving lies in the large amount of possible proofs with different degrees and there is presented an algorithm (Detection of Consequent Formulas - DCF) solving this problem. The algorithm is based on detection of such redundant formulas (proofs) with different degrees.

The article presents the method which is the main point of the work on any automated prover. There is a lot of strategies which makes proofs more efficient when we use refutational proving. We consider well-known strategies - orderings, filtration strategy, set of support etc. One of the most effective strategies is the elimination of consequent formulas. It means the check if a resolvent is not a logical consequence of a formula in set of axioms or a previous resolvent. If such a condition holds it is reasonable to not include the resolvent into the set of resolvents, because if the refutation can be deduced from it, then so it can be deduced from the original resolvent, which it implies of.

2. First-order logic

For the purposes of $(RRTP_{FPL})$ it will be used generalized principle of resolution, which is defined in the research report (Bachmair, L.). There is a propositional form of the rule defined at first and further it is lifted into first-order logic. It is introduced the propositional form of the general resolution.

General resolution - propositional version

$$\frac{F[G] \quad F'[G]}{F[G/\bot] \vee F'[G/\top]} \tag{1}$$

where the propositional logic formulas F and F' are the premises of inference and G is an occurrence of a subformula of both F and F'. The expression $F[G/\bot] \vee F'[G/\top]$ is the resolvent of the premises on G. Every occurrence of G is replaced by false in the first formula and by true in the second one. It is also called F the positive, F′ the negative premise, and G the resolved subformula.

The proof of the soundness of the rule is similar to clausal resolution rule proof. Suppose the Interpretation I in which both premises are valid. In I, G is either true or false. If G ($\neg G$) is true in I, so is $F'[G/\top]$ ($F[G/\bot]$).

Revised version of the paper which forms the core of the handbook (Bachmair, L.) is closely related with notion of selection functions and ordering constraints. By a selection functions it is meant a mapping S that assigns to each clause C a (possibly empty) multiset $S(C)$ of negative literals in C. In other words, the function S selects (a possibly empty) negative subclause of C. We say that an atom A, or a literal $\neg A$, is selected by S if $\neg A$ occurs in $S(C)$. There are no selected atoms or literals if $S(C)$ is empty. Lexicographic path ordering can be used as an usual ordering over a total precedence. But in this case the ordering is admissible if predicate symbols have higher precedence than logical symbols and the constants $\top$ and $\bot$ are smaller than the other logical symbols. It means the ordering is following $A \succ \equiv \succ \supset \succ \neg \succ \vee \succ \wedge \succ \top \succ \bot$. The handbook also addresses another key issues for automated theorem proving - the efficiency of the proof search. This efficiency is closely related with the notion of *redundancy*.

If we want to generalize the notion of resolution and lift it into first-order case we have to define first the notion of selection function for general clauses. General clauses are multisets of arbitrary quantifier-free formulas, denoting the disjunction of their elements. Note that we can also work with a special case of such general clause with one element, which yields to a standard quantifier-free formula of first-order logic. A (general selection) function is a mapping S that assigns to each general clause C a (possibly empty) set C of non-empty sequences of (distinct) atoms in C such that either $S(C)$ is empty or else, for all interpretations I in which C is false, there exists a sequence $A_1, ..., A_k$ in $S(C)$, all atoms of which are true in I. A sequence $A_1, ..., A_k$ in $S(C)$ is said to be *selected* (by S).

We have to define the notion of polarity for these reasons according to the handbook (Bachmair, L.). It is based on the following assumption that a subformula F' in $E[F']$ is *positive* (resp. *negative*), if $E[F'/\top]$ (resp. $E[F'/\bot]$) is a tautology. Thus, if F' is *positive* (resp. *negative*) in E, F' (resp. $\neg F'$) logically implies E. Even it should seem that determining of the polarity of any subformula is NP-complete (hard) problem, we can use syntactic criteria for this computation. In this case the complexity of the algorithm is linear (note that we base our theory on similar syntactic criteria below - structural notions definition).

Proposition 1. *Polarity criteria*

1. *F is a positive subformula of F.*
2. *If $\neg G$ is a positive (resp. negative) subformula of F, then G is a negative (resp. positive) subformula of F.*
3. *If $G \vee H$ is a positive subformula of F, then G and H are both positive subformulas of F.*
4. *If $G \wedge H$ is a negative subformula of F, then G and H are both negative subformulas of F.*
5. *If $G \rightarrow H$ is a positive subformula of F, then G is a negative subformula and H is a positive subformula of F.*
6. *If $G \rightarrow \bot$ is a negative subformula of F, then G is a positive subformula of F.*

7. F is positive in a clause C if it is an element of C.

Note that this proposition applies both to formulas and clauses and allows us to determine polarity of any subformula in a formula. It is safe to *select any sequence of negative atoms* in a general clause, since a negative atom cannot be false in an interpretation the clause is false. With the notion of the polarity as a selection function there is possible to state another notion of General resolution based on orderings applied to clauses.

General ordered resolution with selection $O_S^{\succ}$

$$\frac{C_1(A_1)...C_n(A_n) \quad D(A_1,...,A_n)}{C_1(\bot)...C_n(\bot) \quad D(\top,...,\top)} \tag{2}$$

where (i) either $A_1,...,A_n$ is selected by S in D, or else $S(D)$ is empty, $n = 1$, A_1 is maximal in D, (ii) each atom A_i is maximal in C_i, and (iii) no clause C_i contains a selected atom.

According to the (Bachmair, L.) an inference system based on this rule is refutationally complete. When trying to extend this into the first-order case we to use lifting lemma.

Lemma 1. *Lifting lemma*
Let M be a set of clauses and $K = G(M)$ (set of ground instances). If

$$\frac{C_1...C_n \quad C_0}{C}$$

is an inference in $O_{S_M}^{\succ}(K)$ then there exist clauses C_i' in M, a clause C', and a ground substitution σ such that

$$\frac{C_1'...C_n' \quad C_0'}{C'}$$

is an inference in $O_S^{\succ}(M)$, $C_i = C_i'\sigma$, and $C = C'\sigma$.

Example 1. ***General resolution - polarity based selection***
1. $\neg a \vee \neg b \vee c$ (axiom),
2. a (axiom), 3. b (axiom)
4. $\bot \quad \vee \quad \neg\top \vee \neg b \vee c$ (a is a negative atom in (1) - selected in (1) as negative premise, and (2) as positive premise respectively) $\Rightarrow \neg b \vee c$
5. $\bot \quad \vee \quad \neg\top \vee c$ (b is a negative atom in (4) - selected in (4) as negative premise, and (3) as positive premise respectively) $\Rightarrow c$

In the example we used the notion of polarity as a selection function. For example in the line 4 we select the atom a upon negative polarity (according the proposition criteria 1, 3 and 2 - level ordered) in formula 1 (it means 1. is a negative premise).

Further we can observe the behavior of the rule within the frame of clausal form resolution. Consider following table showing various cases of resolution on clauses.

Example 2. ***General resolution with equivalence***
1. $a \wedge c \leftrightarrow b \wedge d$ (axiom), 2. $a \wedge c$ (axiom), 3.$\neg[b \wedge d]$ (axiom) - negated goal
4. $[a \wedge \bot] \vee [a \wedge \top]$ (resolvent from (2), (2) on c) $\Rightarrow a$
5. $[a \wedge \bot] \vee [a \wedge \top \leftrightarrow b \wedge d]$ ((2), (1) on c) $\Rightarrow a \leftrightarrow b \wedge d$
6. $\bot \vee [\top \leftrightarrow b \wedge d]$ ((4), (5) on a) $\Rightarrow b \wedge d$
7. $\bot \wedge d \vee \top \wedge d$ ((6), (6) on b) $\Rightarrow d$
8. $b \wedge \bot \vee b \wedge \top$ ((6), (6) on d) $\Rightarrow b$

Premise1	Premise2	Resolvent	Simplified	Comments
$a \vee b$	$b \vee c$	$(a \vee \bot) \vee (\top \vee c)$	$\top$	no compl. pair
$a \vee \neg b$	$b \vee c$	$(a \vee \top) \vee (\top \vee c)$	$\top$	redundant inference
$a \vee b$	$\neg b \vee c$	$(a \vee \bot) \vee (\bot \vee c)$	$a \vee c$	clausal resolution
$a \vee \neg b$	$\neg b \vee c$	$(a \vee \top) \vee (\bot \vee c)$	$\top$	no compl. pair

Table 1. Clausal resolution in the context of the non-clausal resolution

9. $\bot \vee \neg[\top \wedge d]$ *((8), (3) on b)* $\Rightarrow \neg d$
10. $\bot \vee \neg\top$ *((7), (9) on d)* $\Rightarrow \bot$ *(refutation)*

When trying to refine the general resolution rule for fuzzy predicate logic, it is important to devise a sound and complete unification algorithm. Standard unification algorithms require variables to be treated only as universally quantified ones. We will present a more general unification algorithm, which can deal with existentially quantified variables without the need for those variables be eliminated by skolemization. It should be stated that the following unification process does not allow an occurrence of the equivalence connective. It is needed to remove equivalence by rewrite rule: $A \leftrightarrow B \Leftrightarrow [A \rightarrow B] \wedge [B \rightarrow A]$.

We assume that the language and semantics of FOL is standard. We use terms - individuals $(a, b, c, ...)$, functions (with n arguments) $(f, g, h, ...)$, variables $(X, Y, Z, ...)$, predicates(with n arguments) $(p, q, r, ...)$, logical connectives $(\wedge, \vee, \rightarrow, \neg)$, quantifiers $(\exists, \forall)$ and logical constants $(\bot, \top)$. We also work with standard notions of logical and special axioms (sets LAx, SAx), logical consequence, consistency etc. as they are used in mathematical logic.

Definition 1. ***Structural notions of a FOL formula***
Let F be a formula of FOL then the structural mappings Sub (subformula), Sup (superformula), Pol (polarity) and Lev (level) are defined as follows:

$F = G \wedge H$ *or* $F = G \vee H$	$Sub(F) = \{G, H\}$, $Sup(G) = F$, $Sup(H) = F$ $Pol(G) = Pol(F)$, $Pol(H) = Pol(F)$
$F = G \rightarrow H$	$Sub(F) = \{G, H\}$, $Sup(G) = F$, $Sup(H) = F$ $Pol(G) = -Pol(F)$, $Pol(H) = Pol(F)$
$F = \neg G$	$Sub(F) = \{G\}$, $Sup(G) = F$ $Pol(G) = -Pol(F)$
$F = \exists \alpha G$ *or* $F = \forall \alpha G$ *(*α *is a variable)*	$Sub(F) = \{G\}$, $Sup(G) = F$ $Pol(G) = Pol(F)$

$Sup(F) = \emptyset \Rightarrow Lev(F) = 0, Pol(F) = 1,$
$Sup(F) \neq \emptyset \Rightarrow Lev(F) = Lev(Sup(F)) + 1$
For mappings Sub and Sup reflexive and transitive closures Sub^* *and* Sup^* *are defined recursively as follows:*
1. $Sub^*(F) \supseteq \{F\}$, $Sup^*(F) \supseteq \{F\}$
2. $Sub^*(F) \supseteq \{H | G \in Sub^*(F) \wedge H \in Sub(G)\}$, $Sup^*(F) \supseteq \{H | G \in Sup^*(F) \wedge H \in Sup(G)\}$

Example: $A \rightarrow B$ - $Pol(A) = -1$, $Pol(B) = 1$, $Lev(A) = 1$

These structural mappings provide framework for assignment of quantifiers to variable occurrences. It is needed for the correct simulation of skolemization (the information about a variable quantification in the prenex form). Subformula and superformula mappings and its closures encapsulate essential hierarchical information of a formula structure. Level gives

the ordering with respect to the scope of variables (which is also essential for skolemization simulation - unification is restricted for existential variables). Polarity enables to decide the global meaning of a variable (e.g. globally an existential variable is universal if its quantification subformula has negative polarity). Sound unification requires further definitions on variable quantification. We will introduce notions of the corresponding quantifier for a variable occurrence, substitution mapping and significance mapping (we have to distinguish between original variables occurring in special axioms and newly introduced ones in the proof sequence).

Definition 2. ***Variable assignment, substitution and significance***
Let F be a formula of FOL, $G = p(t_1, ..., t_n) \in Sub^*(F)$ *atom in F and* α *a variable occurring in* t_i*. Variable mappings Qnt(quantifier assignment), Sbt (variable substitution) and Sig(significance) are defined as follows:*

$Qnt(\alpha) = Q\alpha H, where Q = \exists \vee Q = \forall, H, I \in Sub^*(F), Q\alpha H \in Sup^*(G),$
$\forall Q\alpha I \in Sup^*(G) \Rightarrow Lev(Q\alpha I) < Lev(Q\alpha H).$
$F[\alpha/t']$ *is a substitution of term* t' *into* α *in* $F \Rightarrow Sbt(\alpha) = t'.$
A variable α *occurring in* $F \in LAx \cup SAx$ *is significant w.r.t. existential substitution,* $Sig(\alpha) = 1$ *iff variable is significant,* $Sig(\alpha) = 0$ *otherwise.*

Example: $\forall x(\forall x A(x) \rightarrow B(x))$ - $Qnt(x) = \forall x A(x)$, for x in $A(x)$ and $Qnt(x) = \forall x(\forall x A(x) \rightarrow B(x))$, for x in $B(x)$.

Note that with Qnt mapping (assignment of first name matching quantifier variable in a formula hierarchy from bottom) we are able to distinguish between variables of the same name and there is no need to rename any variable. Sbt mapping holds substituted terms in a quantifier and there is no need to rewrite all occurrences of a variable when working with this mapping within unification. It is also clear that if $Qnt(\alpha) = \emptyset$ then α is a free variable. These variables could be simply avoided by introducing new universal quantifiers to F. Significance mapping is important for differentiating between original formula universal variables and newly introduced ones during proof search (an existential variable can't be bounded with it).

Before we can introduce the standard unification algorithm, we should formulate the notion of global universal and global existential variable (it simulates conversion into prenex normal form).

Definition 3. ***Global quantification***
Let F be a formula without free variables and α *be a variable occurrence in a term of F.*

1. α *is a global universal variable* $(\alpha \in Var_\forall(F))$ *iff* $(Qnt(\alpha) = \forall \alpha H \wedge Pol(Qnt(\alpha)) = 1)$ *or* $(Qnt(\alpha) = \exists \alpha H \wedge Pol(Qnt(\alpha)) = -1)$
2. α *is a global existential variable* $(\alpha \in Var_\exists(F))$ *iff* $(Qnt(\alpha) = \exists \alpha H \wedge Pol(Qnt(\alpha)) = 1)$ *or* $(Qnt(\alpha) = \forall \alpha H \wedge Pol(Qnt(\alpha)) = -1)$

$Var_\forall(F)$ *and* $Var_\exists(F)$ *are sets of global universal and existential variables.*

Example: $F = \forall y(\forall x A(x) \rightarrow B(y))$ - x is a global existential variable, y is a global universal variable.

It is clear w.r.t. skolemization technique that an existential variable can be substituted into an universal one only if all global universal variables over the scope of the existential one have been already substituted by a term. Skolem functors function in the same way. Now we can define the most general unification algorithm based on recursive conditions (extended unification in contrast to standard MGU).

Definition 4. ***Most general unifier algorithm***
Let $G = p(t_1, ..., t_n)$ and $G' = r(u_1, ..., u_n)$ be atoms. Most general unifier (substitution mapping) MGU(G, G') = σ is obtained by following atom and term unification steps or the algorithm returns fail-state for unification. For the purposes of the algorithm we define the Variable Unification Restriction (VUR).

Variable Unification Restriction

Let F_1 be a formula and α be a variable occurring in F_1, F_2 be a formula, t be a term occurring in F_2 and β be a variable occurring in F_2. Variable Unification Restriction (VUR) for (α,t) holds if one of the conditions 1. and 2. holds:

1. *α is a global universal variable and $t \neq \beta$, where β is a global existential variable and α not occurring in t (non-existential substitution)*
2. *α is a global universal variable and $t = \beta$, where β is a global existential variable and $\forall F \in Sup^*(Qnt(\beta))$, $F = Q\gamma G$, $Q \in \{\forall, \exists\}$, γ is a global universal variable, $Sig(\gamma) = 1 \Rightarrow (Sbt(\gamma) = r') \in \sigma$, r' is a term (existential substitution).*

Atom unification

1. *if $n = 0$ and $p = r$ then $\sigma = \emptyset$ and the unifier exists (success-state).*
2. *if $n > 0$ and $p = r$ then perform term unification for pairs $(t_1, u_1), \ldots, (t_n, u_n)$; If for every pair unifier exists then $MGU(G, G') = \sigma$ obtained during term unification (success state).*
3. *In any other case unifier does not exist (fail-state).*

Term unification (t', u')

1. *if $u' = \alpha$, $t' = \beta$ are variables and $Qnt(\alpha) = Qnt(\beta)$ then unifier exists for (t', u') (success-state) (occurrence of the same variable).*
2. *if $t' = \alpha$ is a variable and $(Sbt(\alpha) = v') \in \sigma$ then perform term unification for (v', u'); The unifier for (t', u') exists iff it exists for (v', u') (success-state for an already substituted variable).*
3. *if $u' = \alpha$ is a variable and $(Sbt(\alpha) = v') \in \sigma$ then perform term unification for (t', v'); The unifier for (t', u') exists iff it exists for (t', v') (success-state for an already substituted variable).*
4. *if $t' = a$, $u' = b$ are individual constants and a = b then for (t',u') unifier exists (success-state).*
5. *if $t' = f(t'_1, ..., t'_m)$, $u' = g(u'_1, ..., u'_n)$ are function symbols with arguments and $f = g$ then unifier for (t', u') exists iff unifier exists for every pair (t'_1, u'_1), ..., (t'_n, u'_n) (success-state).*
6. *if $t' = \alpha$ is a variable and VUR for (t', u') holds then unifier exists for (t', u') holds and $\sigma = \sigma \cup (Sbt(\alpha) = u')$ (success-state).*
7. *if $u' = \alpha$ is a variable and VUR for (u', t') holds then unifier exists for (t', u') holds and $\sigma = \sigma \cup (Sbt(\alpha) = t')$ (success-state).*
8. *In any other case unifier does not exist (fail-state).*

$MGU(A) = \sigma$ for a set of atoms $A = \{G_1, \ldots, G_k\}$ is computed by the atom unification for $(G_1, G_i), \sigma_i = MGU(G_1, G_i), \forall i, \sigma_0 = \emptyset$, where before every atom unification (G_1, G_i), σ is set to σ_{i-1}.

With above defined notions it is simple to state the general resolution rule for FOL (without the equivalence connective). It conforms to the definition from (Bachmair, L.).

Definition 5. ***General resolution for first-order logic*** (GR_{FOL})

$$\frac{F[G_1, , ..., G_k] \quad F'[G'_1, ..., G'_n]}{F\sigma[G/\bot] \vee F'\sigma[G/\top]} \tag{3}$$

where $\sigma = MGU(A)$ is the most general unifier (MGU) of the set of the atoms $A = \{G_1, \ldots, G_k, G'_1, \ldots, G'_n\}$, $G = G_1\sigma$. For every variable α in F or F', $(Sbt(\gamma) = \alpha) \cap \sigma = \emptyset \Rightarrow Sig(\alpha) = 1$ in F or F' iff $Sig(\alpha) = 1$ in $F\sigma[G/\bot] \vee F'\sigma[G/\top]$. F is called positive and F' is called negative premise, G represents an occurrence of an atom. The expression $F\sigma[G/\bot] \vee F'\sigma[G/\top]$ is the resolvent of the premises on G.

Note that with Qnt mapping we are able to distinguish variables not only by its name (which may not be unique) but also with this mapping (it is unique). Sig property enables to separate variables, which were not originally in the scope of an existential variable. When utilizing the rule it should be set the Sig mapping for every variable in axioms and negated goal to one. We present a very simple example of existential variable unification before we introduce the refutational theorem prover for FOL.

Example 3. *Variable Unification Restriction*
We would try to prove if $\forall X \exists Y p(X, Y) \vdash \exists Y \forall X p(X, Y)$? We will use refutational proving and therefore we will construct a special axiom from the first formula and negation of the second formula:
$F_0 : \forall X \exists Y p(X, Y)$. $F_1(\neg query) : \neg \exists Y \forall X p(X, Y)$.
There are 2 trivial and 2 non-trivial combinations how to resolve F_0 and F_1 (combinations with the same formula as the positive and the negative premise could not lead to refutation since they are consistent):
Trivial cases: $R[F_1 \& F_1] : \bot \vee \top$ and $R[F_0 \& F_0] : \bot \vee \top$. Both of them lead to $\top$ and the atoms are simply unifiable since the variables are the same.
Non-trivial cases:$[F_1 \& F_0]$: no resolution is possible.
$Y \in Var_\forall(F_1)$ and $Y \in Var_\exists(F_0)$ can't unify since VUR for (Y, Y) does not hold - there is a variable $X \in Sup^*(Qnt(Y))$(over the scope), $X \in Var_\forall(F_0), Sbt(X) = \emptyset$); the case with variable X is identical.
$[F_0 \& F_1]$: no resolution is possible (the same reason as above).
No refutation could be derived from F_0 and F_1 due to VUR.

Further we would like to prove $\exists Y \forall X p(X, Y) \vdash \forall X \exists Y p(X, Y)$.
F0 : $\exists Y \forall X p(X, Y)$. F1 ($\neg$query) : $\neg \forall X \exists Y p(X, Y)$
In this case we can simply derive a refutation:
$R[F_1 \& F_0] : \bot \vee \neg\top(refutation)$
$X \in Var_\forall(F_0)$ and $X \in Var_\exists(F_1)$ can unify since VUR for (X, X) holds - there is no global universal variable over the scope of X in F_1; $Sbt(X) = X$ and $Sbt(Y) = Y$.

3. Fuzzy predicate logic and refutational proof

The fuzzy predicate logic with evaluated syntax is a flexible and fully complete formalism, which will be used for the below presented extension (Novák, V.). In order to use an efficient form of the resolution principle we have to extend the standard notion of a proof (provability value and degree) with the notion of refutational proof (refutation degree). Propositonal version of the fuzzy resolution principle has been already presented in (Habiballa, H.). We suppose that set of truth values is Łukasiewicz algebra. Therefore we assume standard notions of conjunction, disjunction etc. to be bound with Łukasiewicz operators.

We will assume Łukasewicz algebra to be

$$\mathcal{L}_{Ł} = \langle [0,1], \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle$$

where [0, 1] is the interval of reals between 0 and 1, which are the smallest and greatest elements respectively. Basic and additional operations are defined as follows:

$$a \otimes b = 0 \vee (a + b - 1) \quad a \rightarrow b = 1 \wedge (1 - a + b) \quad a \oplus b = 1 \wedge (a + b) \quad \neg a = 1 - a$$

The biresiduation operation $\leftrightarrow$ could be defined $a \leftrightarrow b =_{df} (a \rightarrow b) \wedge (b \rightarrow a)$, where $\wedge$ is infimum operation. The following properties of $\mathcal{L}_{Ł}$ will be used in the sequel:
$a \otimes 1 = a, a \otimes 0 = 0, a \oplus 1 = 1, a \oplus 0 = a, a \rightarrow 1 = 1, a \rightarrow 0 = \neg a, 1 \rightarrow a = a, 0 \rightarrow a = 1$
The syntax and semantics of fuzzy predicate logic is following:

- terms $t_1, ..., t_n$ are defined as in FOL
- predicates with $p_1, ..., p_m$ are syntactically equivalent to FOL ones. Instead of 0 we write $\bot$ and instead of 1 we write $\top$, connectives - & (Łukasiewicz conjunction), ∇ (Łukasiewicz disjunction), $\Rightarrow$ (implication), $\neg$ (negation), $\forall X$ (universal quantifier),$\exists X$ (existential quantifier) and furthermore by F_J we denote set of all formulas of fuzzy logic in language J
- FPL formulas have the following semantic interpretations (D is the universe): Interpretation of terms is equivalent to FOL, $\mathcal{D}(p_i(t_{i_1}, ..., t_{i_n})) = P_i(\mathcal{D}(t_{i_1}), ..., \mathcal{D}(t_{i_n}))$ where P_i is a fuzzy relation assigned to p_i, $\mathcal{D}(\mathfrak{a}) = a$ for $a \in [0,1]$, $\mathcal{D}(A \,\&\, B) = \mathcal{D}(A) \otimes \mathcal{D}(B)$, $\mathcal{D}(A \nabla B) = \mathcal{D}(A) \oplus \mathcal{D}(B)$, $\mathcal{D}(A \Rightarrow B) = \mathcal{D}(A) \rightarrow \mathcal{D}(B)$, $\mathcal{D}(\neg A) = \neg \mathcal{D}(A)$, $\mathcal{D}(\forall X(A)) = \bigwedge \mathcal{D}(A[x/d] | d \in D)$, $\mathcal{D}(\exists X(A)) = \bigvee \mathcal{D}(A[x/d] | d \in D)$
- for every subformula defined above $Sub, Sup, Pol, Lev, Qnt, Sbt, Sig$ and other derived properties defined for classical logic hold (where the classical FOL connective is presented the Łukasiewicz one has the same mapping value).

Graded fuzzy predicate calculus assigns grade to every axiom, in which the formula is valid. It will be written as a/A where A is a formula and a is a syntactic evaluation. We use several standard notions defined in (Novák, V.) namely: inference rule, formal fuzzy theory with set of logical and special axioms, evaluated formal proof.

Definition 6. *Inference rule*
An n-ary inference rule r in the graded logical system is a scheme

$$r : \frac{a_1/A_1, ..., \ a_n/A_n}{r^{evl}(a_1, ..., a_n)/r^{syn}(A_1, ..., A_n)} \tag{4}$$

using which the evaluated formulas $a_1/A_1, ..., \ a_n/A_n$ *are assigned the evaluated formula* $r^{evl}(a_1, ..., a_n)/r^{syn}(A_1, ..., A_n)$. *The syntactic operation* r^{syn} *is a partial n-ary operation on* F_J *and the evaluation operation* r^{evl} *is an n-ary lower semicontinous operation on L (i.e. it preserves arbitrary suprema in all variables).*

Definition 7. *Formal fuzzy theory*
A formal fuzzy theory T in the language J is a triple

$$T = \langle \mathrm{LAx}, \mathrm{SAx}, R \rangle$$

where $\mathrm{LAx} \subsetneq F_J$ *is a fuzzy set of logical axioms,* $\mathrm{SAx} \subsetneq F_J$ *is a fuzzy set of special axioms, and R is a set of sound inference rules.*

Definition 8. ***Evaluated proof, refutational proof and refutation degree***
An evaluated formal proof of a formula A from the fuzzy set $X \underset{\sim}{\subseteq} F_J$ *is a finite sequence of evaluated formulas* $w := a_0/A_0,\ a_1/A_1, ...,\ a_n/A_n$ *such that* $A_n := A$ *and for each* $i \leq n$*, either there exists an m-ary inference rule r such that*
$a_i/A_i := r^{evl}(a_{i_1}, ..., a_{i_m})/r^{syn}(A_{i_1}, ..., A_{i_m}), \quad i_1, ..., i_m < n$ *or* $a_i/A_i := X(A_i)/A_i$.
We will denote the value of the evaluated proof by $Val(w) = a_n$.
An evaluated refutational formal proof of a formula A from X is w, where additionally $a_0/A_0 := 1/\neg A$ *and* $A_n := \bot$. $Val(w) = a_n$ *is called refutation degree of A.*

Definition 9. ***Provability and truth***
Let T be a fuzzy theory and $A \in F_J$ *a formula. We write* $T \vdash_a A$ *and say that the formula A is a theorem in the degree a, or provable in the degree a in the fuzzy theory T.*

$$T \vdash_a A \text{ iff} a = \bigvee\{Val(w) | \text{ w is a proof of A from LAx} \cup \text{SAx}\} \tag{5}$$

We write $T \models_a A$ *and say that the formula A is true in the degree a in the fuzzy theory T.*

$$\mathcal{D} \models T \text{ if } \forall A \in \text{LAx} : \text{LAx}(A) \leq \mathcal{D}(A),\ A \in \text{SAx} : \text{SAx}(A) \leq \mathcal{D}(A) \tag{6}$$

$$T \models_a A \text{ iff } a = \bigwedge\{\mathcal{D}(A) \,|\, \mathcal{D} \models T\} \tag{7}$$

The fuzzy modus ponens rule could be formulated:

Definition 10. *Fuzzy modus ponens*

$$r_{MP} : \frac{a/A,\ b/A \Rightarrow B}{a \otimes b/B} \tag{8}$$

where from premise A holding in the degree a and premise $A \Rightarrow B$ *holding in the degree b we infer B holding in the degree* $a \otimes b$.

In classical logic r_{MP} could be viewed as a special case of the resolution. The fuzzy resolution rule presented below is also able to simulate fuzzy r_{MP}. From this fact the completeness of a system based on resolution can be deduced. It will only remain to prove the soundness. It is possible to introduce following notion of resolution w.r.t. the modus ponens.

Definition 11. ***General resolution for fuzzy predicate logic*** (GR_{FPL})

$$r_{GR} : \frac{a/F[G_1, , ..., G_k],\ b/F'[G'_1, ..., G'_n]}{a \otimes b/F\sigma[G/\bot] \nabla F'\sigma[G/\top]} \tag{9}$$

where $\sigma = MGU(A)$ *is the most general unifier (MGU) of the set of the atoms*
$A = \{G_1, \ldots, G_k, G'_1, \ldots, G'_n\}$, $G = G_1\sigma$. *For every variable* α *in F or F',* $(Sbt(\gamma) = \alpha) \cap \sigma = \emptyset \Rightarrow Sig(\alpha) = 1$ *in*

F or F' iff $Sig(\alpha) = 1$ *in* $F\sigma[G/\bot] \vee F'\sigma[G/\top]$. *F is called positive and F' is called negative premise, G represents an occurrence of an atom. The expression* $F\sigma[G/\bot] \vee F'\sigma[G/\top]$ *is the resolvent of the premises on G.*

Lemma 2. ***Soundness of*** r_{GR}
The inference rule r_{GR} *for FPL based on* $\mathcal{L}_{\L}$ *is sound i.e. for every truth valuation* $\mathcal{D}$*,*

$$\mathcal{D}(r^{syn}(A_1,...,A_n)) \geq r^{evl}(\mathcal{D}(A_1),...,\mathcal{D}(A_n)) \tag{10}$$

holds true.

Proof. Before we solve the core of GR_{FPL} we should prove that the unification algorithm preserves soundness. But it could be simply proved since in the classical FPL with the rule of Modus-Ponens (Novák, V.) from the axiom $\vdash (\forall x)A \Rightarrow A[x/t]$ and $\vdash (\forall x)A$ we can prove $A[x/t]$. For r_{GR} we may rewrite the values of the left and right parts of equation (10):

$$\mathcal{D}(r^{syn}(A_1,...,A_n)) = \mathcal{D}[\mathcal{D}(F_1[G/\bot])\nabla\mathcal{D}(F_2[G/\top])]$$

$$r^{evl}(\mathcal{D}(A_1),...,\mathcal{D}(A_n)) = \mathcal{D}(F_1[G]) \otimes \mathcal{D}(F_2[G])$$

It is sufficient to prove the equality for $\Rightarrow$ since all other connectives could be defined by it. By induction on the complexity of formula $|A|$, defined as the number of occurrences of connectives, we can prove:

Let premises F_1 and F_2 be atomic formulas. Since they must contain the same subformula then $F_1 = F_2 = G$ and it holds

$$\mathcal{D}[\mathcal{D}(F_1[G/\bot])\nabla\mathcal{D}(F_2[G/\top])] = D(\bot\nabla\top) = 0 \oplus 1 = 1 \geq \mathcal{D}(F_1[G]) \otimes \mathcal{D}(F_2[G])$$

Induction step: Let premises F_1 and F_2 be complex formulas and let A and B are subformulas of F_1, C and D are subformulas of F_2 and G is an atom where generally $F_1 = (A \Rightarrow B)$ and $F_2 = (C \Rightarrow D)$. The complexity of $|F_1| = |A| + 1$ or $|F_1| = |B| + 1$ and $|F_2| = |C| + 1$ or $|F_2| = |D| + 1$. Since they must contain the same subformula and for A, B, C, D the induction presupposition hold it remain to analyze the following cases:

1. $F_1 = A \Rightarrow G \quad F_2 = G \Rightarrow D : \mathcal{D}[\mathcal{D}(F_1[G/\bot])\nabla\mathcal{D}(F_2[G/\top])] = \mathcal{D}([A \Rightarrow \bot]\nabla[\top \Rightarrow D]) = \mathcal{D}(\neg A \nabla D) = 1 \wedge (1 - a + d)$
   We have rewritten the expression into Łukasiewicz interpretation. Now we will try to rewrite the right side of the inequality, which has to be proven.
   $\mathcal{D}(F_1[G]) \otimes \mathcal{D}(F_2[G]) = \mathcal{D}(A \Rightarrow G) \otimes \mathcal{D}(G \Rightarrow D) = 0 \vee ((1 \wedge (1 - a + g)) + (1 \wedge (1 - g + d)) - 1) = 1 \wedge (1 - a + d)$ The left and right side of the equation (10) are equal and therefore

   $$\mathcal{D}[\mathcal{D}(F_1[G/\bot])\nabla\mathcal{D}(F_2[G/\top])] \geq \mathcal{D}(F_1[G]) \otimes \mathcal{D}(F_2[G])$$

   for this case holds.
2. $F_1 = A \Rightarrow G \quad F_2 = C \Rightarrow G : \mathcal{D}[\mathcal{D}(F_1[G/\bot])\nabla\mathcal{D}(F_2[G/\top])] = \mathcal{D}([A \Rightarrow \bot]\nabla[C \Rightarrow \top]) = 1 \geq \mathcal{D}(F_1[G]) \otimes \mathcal{D}(F_2[G])$
3. $F_1 = G \Rightarrow B \quad F_2 = G \Rightarrow D : \mathcal{D}[\mathcal{D}(F_1[G/\bot])\nabla\mathcal{D}(F_2[G/\top])] = \mathcal{D}([\bot \Rightarrow B]\nabla[\top \Rightarrow D]) = 1 \geq \mathcal{D}(F_1[G]) \otimes \mathcal{D}(F_2[G])$
4. $F_1 = G \Rightarrow B \quad F_2 = C \Rightarrow G : \mathcal{D}[\mathcal{D}(F_1[G/\bot])\nabla\mathcal{D}(F_2[G/\top])] = \mathcal{D}([\bot \Rightarrow B]\nabla[C \Rightarrow \top]) = 1 \geq \mathcal{D}(F_1[G]) \otimes \mathcal{D}(F_2[G])$

By induction we have proven that the inequality holds and the r_R is sound. The induction of the case where only one of the premises has greater complexity is included in the above solved induction step. □

Definition 12. ***Refutational resolution theorem prover for FPL***
Refutational non-clausal resolution theorem prover for FPL ($RRTP_{FPL}$) is the inference system with the inference rule GR_{FPL} and sound simplification rules for $\bot$, $\top$ (standard equivalencies for logical constants). A refutational proof by definition 8 represents a proof of a formula G (goal) from the set of special axioms N. It is assumed that $Sig(\alpha) = 1$ for $\forall\alpha$ in $F \in N \cup \neg G$ formula, every formula in a proof has no free variable and has no quantifier for a variable not occurring in the formula.

Definition 13. ***Simplification rules for*** $\nabla, \Rightarrow$

$$r_{s\nabla} : \frac{a/\bot\nabla A}{a/A} \quad and \quad r_{s\Rightarrow} : \frac{a/\top\Rightarrow A}{a/A}$$

Lemma 3. ***Provability and refutation degree for*** GR_{FPL}
$T \vdash_a A$ iff $a = \bigvee\{Val(w)|$ w is a refutational proof of A from $\mathrm{LAx} \cup \mathrm{SAx}\}$

Proof. If $T \vdash_a A$ then $a = \bigvee\{Val(w)|$ w is a proof of A from $\mathrm{LAx} \cup \mathrm{SAx}\}$ and for every such a proof of we can construct refutational proof as follows ($Val(w) \leq a$):
$w := {}^a/_A$ {proof A}, ${}^1/_{\neg A}$ {member of refutational proof}, $a \otimes 1/\bot$ $\{r_{GR}\}$
If $a = \bigvee\{Val(w)|$ w is refutational proof of A from $\mathrm{LAx} \cup \mathrm{SAx}\}(Val(w) \leq a)$:
$w := {}^{a_0}/_{A_0}, ..., {}^{a_i}/_{A_i}, {}^1/_{\neg A}, ..., {}^a/_{\bot}$, where $A_0, ..., A_i$ are axioms.
There is a proof:
$w' := {}^{a_0}/_{A_0}, ..., {}^{a_i}/_{A_i}, {}^1/_{\neg A\nabla A}, {}^{a_{i+2}}/_{A_{i+2}\nabla A}, ..., {}^a/_{\bot\nabla A}$.
All the schemes of the type $A_j\nabla A$, $j > i$ could be simplified by sound simplification rules and the formula $\neg A\nabla A$ may be removed.
The proof $w'' := {}^{a_0}/_{A_0}, ..., {}^{a_i}/_{A_i}, {}^{a_{i+2}}/_{A_{i+2}\nabla A}, ..., {}^a/_A$ is a correct proof of A in the degree a since the formulas are either axioms or results of application of resolution. □

Theorem 1. ***Completeness for fuzzy logic with*** r_{GR}, $r_{s\nabla}$, $r_{s\Rightarrow}$ ***instead of*** r_{MP}
Formal fuzzy theory, where r_{MP} is replaced with r_{GR}, $r_{s\nabla}$, $r_{s\Rightarrow}$, is complete i.e. for every A from the set of formulas $T \vdash_a A$ iff $T \models_a A$.

Proof. The left to right implication (soundness of such formal theory) could be easily done from the soundness of the resolution rule. Conversely it is sufficient to prove that the rule r_{MP} can be replaced by r_{GR}, $r_{s\nabla}$, $r_{s\Rightarrow}$. Indeed, let w be a proof:
$w := {}^a/_A$ {proof w_a}, ${}^b/_{A\Rightarrow B}$ {proof $w_{A\Rightarrow B}$}, $a \otimes b/B$ $\{r_{MP}\}$. Then we can replace it by the proof:
$w := {}^a/_A\{proof\, w_a\}, {}^b/_{A\Rightarrow B}\{proof\, w_{A\Rightarrow B}\}, a \otimes b/\bot\nabla[\top\Rightarrow B]\{r_{GR}\}$,
$a \otimes b/\top\Rightarrow B\{r_{s\nabla}, a \otimes b/B\{r_{s\Rightarrow}\}$

Using the last sequence we can easily make a proof with r_{MP} also with the proposed r_R and simplification rules. Since usual formal theory with r_{MP} is complete as it is proved in (Novák, V.), every fuzzy formal theory with these rules is also complete. Note that the non-ground case (requiring unification) could be simulated in the same way like in the proof of soundness. □

4. Implementation and efficiency

The author also currently implements the non-clausal theorem prover into fuzzy logic as an extension of previous prover for FOL (GEneralized Resolution Deductive System - GERDS) (Habiballa, H.). Experiments concerning prospective inference strategies can be performed with this extension. The prover called Fuzzy Predicate Logic GEneralized Resolution

Deductive System (Fig. 1) - FPLGERDS provides standard interface for input (knowledge base and goals) and output (proof sequence and results of fuzzy inference, statistics).

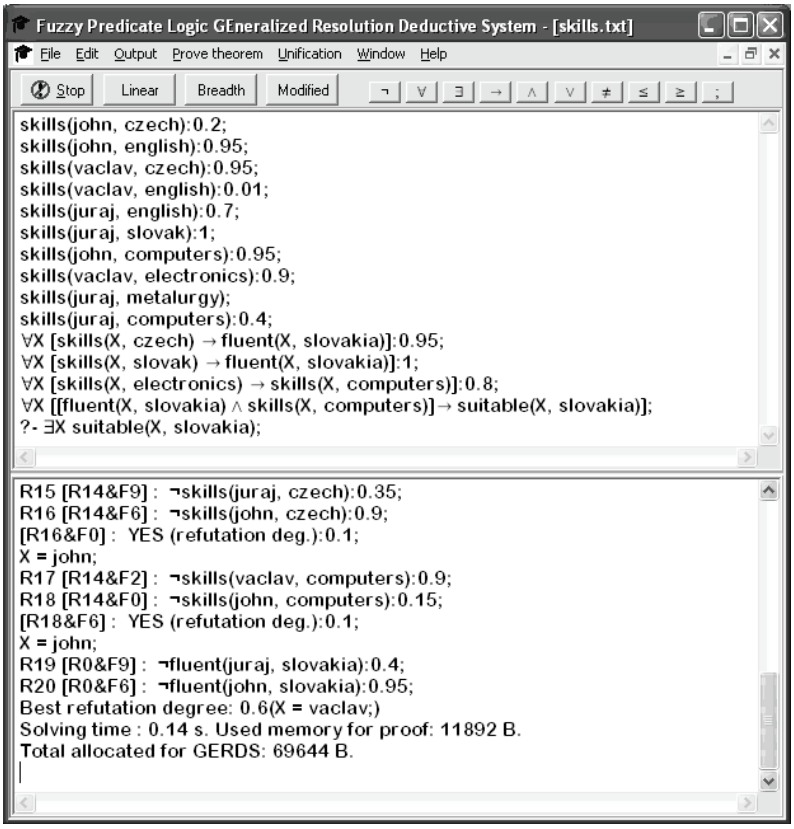


Fig. 1. Fuzzy Predicate Logic GEneralized Resolution Deductive System

There are already several efficient strategies proposed by author (mainly Detection of Consequent Formulas (DCF) adopted for the usage also in FPL). With these strategies the proving engine can be implemented in real-life applications since the complexity of theorem proving in FPL is dimensionally harder than in FOL (the need to search for all possible proofs - we try to find the best refutation degree). The DCF idea is to forbid the addition of a resolvent which is a logical consequence of any previously added resolvent. For refutational theorem proving it is a sound and complete strategy and it is emiprically very effective. Completeness of such a strategy is also straight-forward in FOL:

$$(R_{old} \vdash R_{new}) \wedge (U, R_{new} \vdash \bot) \Rightarrow (U, R_{old} \vdash \bot)$$

Example: $R_{new} = p(a)$, $R_{old} = \forall x(p(x))$, $R_{old} \vdash R_{new}$.

DCF could be implemented by the same procedures like General Resolution (we may utilize self-resolution). Self-resolution has the same positive and negative premise and needs to

resolve all possible combinations of an atom. It uses the following scheme:

$$R_{old} \vdash R_{new} \Leftrightarrow \neg(R_{old} \rightarrow R_{new}) \vdash \bot$$

Even the usage of this teachnique is a semidecidable problem, we can use time or step limitation of the algorithm and it will not affect the completeness of the $RRTP_{FOL}$.
Example: $R_{new} = p(a)$, $R_{old} = \forall x(p(x))$, $\neg(\forall x(p(x)) \rightarrow p(a))$
MGU: $Sbt(x) = a$, $Res = \neg(\bot \rightarrow \bot) \vee \neg(\top \rightarrow \top) \Rightarrow \bot$
We have proved that R_{new} is a logical consequence of R_{old}.

In FPL we have to enrich the DCF procedure by the limitation on the provability degree. if $U \vdash_a R_{old} \wedge U \vdash_b R_{new} \wedge b \leq a$ then we can apply DCF. DCF Trivial check performs a symbolic comparison of R_{old} and R_{new} we use the same provability degree condition. In other cases we have to add R_{new} into the set of resolvents and we can apply DCF Kill procedure. DCF Kill searches for every R_{old} being a logical consequence of R_{new} and if $U \vdash_a R_{old} \wedge U \vdash_b R_{new} \wedge b \geq a$ then Kill R_{old} (resolvent is removed).

We will now show some efficiency results concerning many-valued logic both for Fuzzy Predicate Logic. We have used the above mentioned application FPLGERDS and originally developed DCF strategy for FPL. It is clear that inference in $RRTP_{FPL}$ and $RRTP_{FDL}$ on general knowledge bases is a problem solved in exponential time. Nevertheless as we would like to demonstrate the need to search for every possible proof (in contrast to the two-valued logic) will not necessarily in particular cases lead to the inefficient theory. We have devised knowledge bases (KB) on the following typical problems related to the use of fuzzy logic.

We have performed experimental measurements concerning efficiency of the presented non-clausal resolution principle and also DCF technique. These measurements were done using the FPLGERDS application (Habiballa, H.). Special testing knowledge bases were prepared and several types of inference were tested on a PC with standard Intel Pentium 4 processor as described below.

Fuzzy predicate Logic redundancy-based inefficient knowledge bases

As it was shown above in the theorem proving example the problem of proof search is quite different in FPL and FDL in comparison with the two-valued logic. We have to search for the best refutation degree using refutational theorem proving in order to make sensible conclusions from the inference process. It means we cannot accept the **first successful** proof, but we have to check **"all possible proofs"** or we have to be sure that every omitted proof is **worse** that some another one. The presented DCF and DCF Kill technique belong to the third sort of proof search strategies, i.e. they omit proofs that are really worse than some another (see the explication above). Proofs and formulas causing this could be called redundant proofs and redundant formulas. Fuzzy logic makes this redundancy dimensionally harder since we could produce not only equivalent formulas but also equivalent formulas of different evaluation degree.

Example 4. ***Redundant knowledge base***
Consider the following knowledge base (fragment):

...,
$0.51/a \wedge b_1 \Rightarrow z$,
$0.61/a \wedge b_1 \wedge b_1 \Rightarrow z$,
$0.71/a \wedge b_1 \wedge b_1 \wedge b_1 \Rightarrow z$,
$0.81/a \wedge b_1 \wedge b_1 \wedge b_1 \wedge b_1 \Rightarrow z$,

Search method		Description
Breadth	B	Level order generation, start - special axioms + goal
Linear	L	Resolvent ⇒ premise, start - goal
Modified-Linear	M	Resolvent ⇒ premise, start - goal + special axioms

Table 2. Proof search algorithms

DCF Method		Description
Trivial	T	Exact symbolic comparison
DCF	DC	Potential resolvent is consequent (no addition)
DCF Kill	DK	DCF + remove all consequent resolvents

Table 3. DCF heuristics

$0.91/a \wedge b_1 \wedge b_1 \wedge b_1 \wedge b_1 \wedge b_1 \Rightarrow z, 1/b_1,$
...,
$0.52/a \wedge b_2 \Rightarrow z,$
$0.62/a \wedge b_2 \wedge b_2 \Rightarrow z,$
$0.72/a \wedge b_2 \wedge b_2 \wedge b_2 \Rightarrow z,$
$0.82/a \wedge b_2 \wedge b_2 \wedge b_2 \wedge b_2 \Rightarrow z,$
$0.92/a \wedge b_2 \wedge b_2 \wedge b_2 \wedge b_2 \wedge b_2 \Rightarrow z, 1/b_2,$
...,
Goal: $? - a \Rightarrow z$

Searching for the best proof of a goal will produce a lot of logically equivalent formulas with different degrees. These resolvents make the inference process inefficient and one of the essential demands to the presented refutational theorem prover is a reasonable inference strategy with acceptable time complexity.

We have compared efficiency of the standard **breadth-first search**, **linear search** and **modified linear search** (starting from every formula in knowledge base) and also combinations with DCF and DCF-kill technique (Habiballa, H.). We have prepared knowledge bases of the size 120, 240, 360, 480 and 600 formulas. It has been compared the time and space efficiency on the criterion of 2 redundancy levels. This level represents the number of redundant formulas to which the formula is equivalent (including the original formula). For example the level 5 means the knowledge base contain 5 equivalent redundant formulas for every formula (including the formula itself). The basic possible state space search techniques and DCF heuristics and their combinations are presented in the following tables.

We use standard state space search algorithms in the FPLGERDS application - Breadth-first and Linear search. Breadth-first method searches for every possible resolvent from the formulas of the level 0 (goal and special axioms). These resolvents form formulas of the level 1 and we try to combine them with all formulas of the same and lower level and continue by the same procedure until no other non-redundant resolvent could be found. Linear search performs depth-first search procedure, where every produced resolvent is used as one of the premises in succeeding step of inference. The first produced resolvents arises from the goal formula. Modified linear search method posses the same procedure as linear one, but it starts from goal and also from all the special axioms.

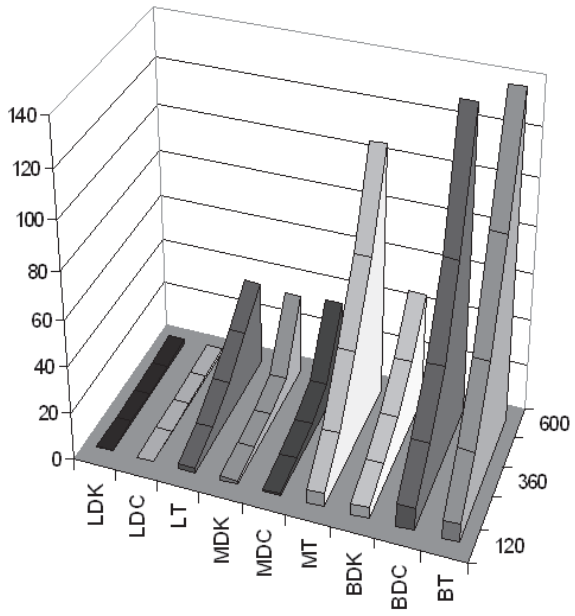
DCF methods for reduction of resolvent space are basically three. The simplest is trivial DCF method, which detects redundant resolvent only by its exact symbolic comparison, i.e. formulas are equivalent only if the are syntactically the same. Even it is a very rough method,

Search	**DCF**	Code	**Description**
Breadth	Trivial	BT	Complete
Breadth	DCF	BDC	Complete
Breadth	DCF Kill	BDK	Complete
Mod. Linear	Trivial	MT	Incomplete (+)
Mod. Linear	DCF	MDC	Incomplete (+)
Mod. Linear	DCF Kill	MDK	Incomplete (+)
Linear	Trivial	LT	Incomplete
Linear	DCF	LDC	Incomplete
Linear	DCF Kill	LDK	Incomplete

Table 4. Inference strategies

it is computationally very simple and forms necessary essential restriction for possibly infinite inference process. The next method of DCF technique enables do detect the equivalency of a formula (potential new resolvent) by the means described above. DCF Kill technique additionally tries to remove every redundant resolvent from the set of resolvents. The important aspect of the theorem DCF lies in its simple implementation into an automated theorem prover based on general resolution. The prover handles formulas in the form of syntactical tree. It is programmed a procedure performing general resolution with two formulas on an atom. This procedure is also used for the implementation of the theorem. A "virtual tree" is created from candidate and former resolvent (axiom) connected by negated implication. Then it remains to perform self-resolution on such formula until a logical value is obtained. Let us compare the efficiency of standard strategies and the above-defined one. We have built-up 9 combinations of inference strategies from the mentioned proof search and DCF heuristics. They have different computational strength, i.e. their completeness is different for various classes of formulas. Fully complete (as described above) for general formulas of FPL and FDL are only breadth-first search combinations. Linear search strategies are not complete even for two-valued logic and horn clauses. Modified linear search has generally bad completeness results when an infinite loop is present in proofs, but for guarded knowledge bases it can assure completeness preserving better space efficiency than breadth-first search. We tested presented inference strategies on sample knowledge bases with redundancy level 5 with 20, 40, 60, 80 and 100 groups of mutually redundant formulas (total number of formulas in knowledge base is 120, 240, 360, 480 and 600). At first we have tested their time efficiency for inference process. As it could be observed from figure 2, the best results have **LDK and LDC** strategies. For simple guarded knowledge bases (not leading to an infinite loop in proof search and where the goal itself assures the best refutation degree) these two methods are **very efficient**. DCF strategies significantly reduces the proof search even in comparison with LT strategy (standard), therefore the usage of any non-trivial DCF heuristics is significant. Next important result concludes from the comparison of BDK and MDK, MDC strategies. We can conclude that MDK and MDC strategies are relatively comparable to BDK and moreover BDK preserves completeness for general knowledge bases.

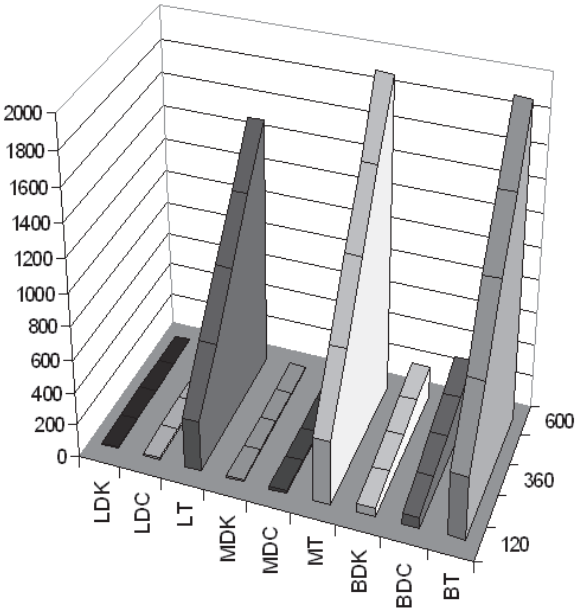
Space complexity is even more significantly affected by the DCF heuristics. There is an interesting comparison of trivial and non-trivial DCF heuristics in figure 3. Even BDK strategy brings significant reduction of resolvents amount, while LDK, LDC, MDK, MDC strategies have minimal necessary amount of kept resolvents during inference process. The second examined redundancy level 10 shows also important comparison for increasing redundancy in knowledge bases. Tested knowledge bases contained 10, 20, 30, 40 and 50 groups of 10 equivalent formulas (the total number of formulas was 110, 220, 330, 440 and 550 formulas).



	120	240	360	480	600
BT	7,44	25,09	50,55	90,91	138,67
BDC	9,52	23,73	43,76	76,63	129,94
BDK	4,68	10,1	17,64	29,54	47,31
MT	5,86	18,85	41,46	70,01	106,68
MDC	0,93	3,09	5,14	10,54	35,14
MDK	1,45	3,2	5,03	8,84	34,32
LT	2,17	7,24	14,76	24,98	35,77
LDC	0,26	0,52	0,57	0,8	1,96
LDK	0,28	0,5	0,57	0,78	1,99

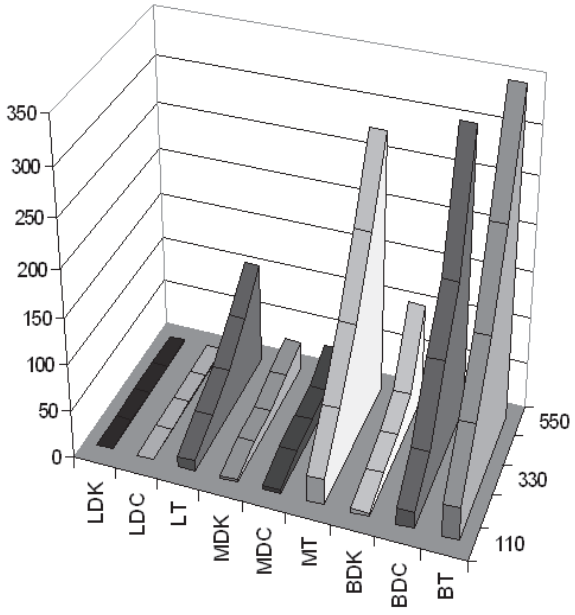
Fig. 2. Time complexity for redundancy level 5 (seconds)

Time efficiency results shows that higher redundancy level causes expected increase in the necessary time for the best proof search (figure 4). The approximate increase is double, while the proportion shows good results for MDK, MDC and LDK, LDC (linear search based) strategies. This property also holds for space complexity as shown in figure 5. Performed experiments shows the significance of originally developed DCF strategies in combination with standard breadth-first search (important for general knowledge bases - **BDK**). We also outlined high efficiency for linear search based strategies (mainly **LDK**). Even this strategy is not fully complete and could be used only for guarded fragment of FDL, this problem is already known in classical (two-valued) logic programming and automated theorem proving. We also use these highly efficient linear search strategies, even they are not complete.



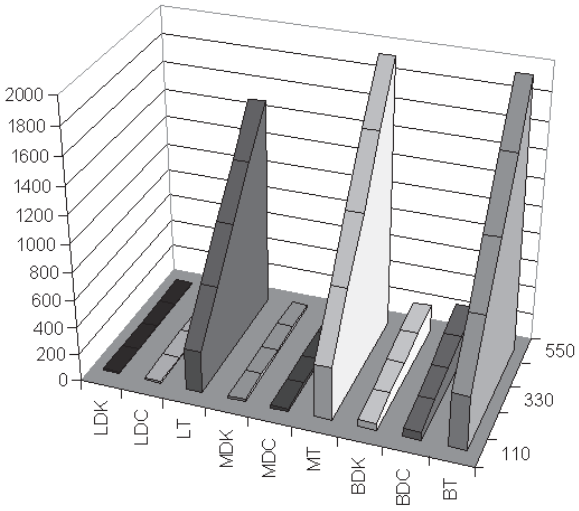
	120	240	360	480	600
BT	385	765	1185	1525	1905
BDC	65	125	185	245	306
BDK	46	86	126	166	205
MT	385	765	1145	1525	1905
MDC	14	14	14	14	12
MDK	14	14	14	14	12
LT	304	604	904	1204	1504
LDC	12	12	12	12	10
LDK	12	12	12	12	10

Fig. 3. Space complexity for redundancy level 5 (resolvents)



	110	220	330	440	550
BT	34,16	89,55	150,35	235,61	348,32
BDC	16,95	47,89	114,01	166,67	301,63
BDK	3,37	13	21,95	35,76	103,81
MT	27,34	71,17	130,21	199,73	278,27
MDC	3,76	13,85	16,05	18,16	34,87
MDK	2,85	11,92	14,35	23,45	31,4
LT	10,99	29,1	49,17	76,83	107,86
LDC	0,61	1,42	2,08	1,67	2,31
LDK	0,44	1,01	1,3	1,1	1,39

Fig. 4. Time complexity for redundancy level 10 (seconds)



	110	220	330	440	550
BT	395	785	1155	1575	1955
BDC	55	105	155	205	255
BDK	46	86	126	166	206
MT	395	785	1155	1575	1955
MDC	17	19	19	19	19
MDK	17	19	19	19	19
LT	304	604	904	1204	1504
LDC	14	16	16	16	16
LDK	14	16	16	16	16

Fig. 5. Space complexity for redundancy level 10 (resolvents)

5. Conclusions and further research

The *Non-clausal Refutational Resolution Theorem Prover* forms a powerful inference system for automated theorem proving in fuzzy predicate logic. The main advantage in contrast with other inference systems lies in the possibility to utilize various inference strategies for effective reasoning. Therefore it is essential for practically successful theorem proving.

The Detection of Consequent Formulas algorithms family brings significant improvements in time and space efficiency for the best proof search. It has been shown results indicating specific behavior of some combinations of the DCF and standard proof search (breadth-first and linear search). DCF strategies (BDC, BDK) have interesting results even for fully general fuzzy predicate logic with evaluated syntax, where the strategy makes the inference process practically manageable (in contrast to unrestricted blind proof-search). However it seems to be more promising for practical applications to utilize incomplete strategies with high time efficiency like LDK (even for large knowledge bases it has very short solving times). It conforms to another successful practical applications in two-valued logic like logic programming or deductive databases where there are also used efficient incomplete strategies for fragments of fully general logics.

It has been briefly presented some efficiency results for the presented automated theorem prover and inference strategies. They show the significant reduction of time and space complexity for the DCF technique. Experimental application FPLGERDS can be obtained from URL:// *http://www1.osu.cz/home/habibal/files/gerds.zip*. The package contains current version of the application, source codes, examples and documentation. This work was supported by project DAR (1M0572).

6. References

Bachmair, L., Ganzinger, H. (1997). A theory of resolution. Technical report: Max-Planck-Institut, 1997.

Bachmair, L., Ganzinger, H. (2001). Resolution theorem proving. In Handbook of Automated Reasoning, MIT Press, 2001.

Dukić, N., Avdagić, Z. (2005). Fuzzy Functional Dependency and the Resolution Principle. In Informatica, Vilnius: Lith. Acad. Sci. (IOSPRESS), 2005, Vol.16, No. 1, pp. 45 - 60, 2005.

Habiballa, H. (2000). Non-clausal resolution - theory and practice. Research report: University of Ostrava, 2000, http://www.volny.cz/habiballa/files/gerds.pdf

Habiballa, H., Novák, V. (2002). Fuzzy General Resolution. In Proc. of Intl. Conf. Aplimat 2002. Bratislava, Slovak Technical University, 2002. pp. 199-206, also available as research rep. at http://ac030.osu.cz/irafm/ps/rep47.ps

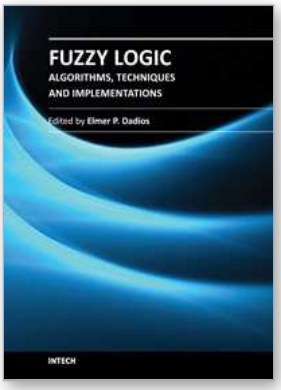
Habiballa, H. (2006). Resolution Based Reasoning in Description Logic. In Proc. of Intl. Conf. ZNALOSTI 2006, Univ. of Hradec Kralove, 2006, also available as research rep. at http://ac030.osu.cz/irafm/ps/rep66.ps.gz.

Habiballa, H.(2006a). Fuzzy Predicate Logic Generalized Resolution Deductive System. Technical Report, Institute for Research and Application of Fuzzy Modeling, University of Ostrava, 2006.

Hájek, P. (2000). Metamathematics of fuzzy logic. Kluwer Academic Publishers - Dordrecht, 2000.

Hájek, P. (2005). Making fuzzy description logic more general. Fuzzy Sets and Systems 154(2005),pp. 1-15.

Novák, V., Perfilieva, I., Močkoř, J. (1999). Mathematical principles of fuzzy logic. Kluwer, 1999.



Fuzzy Logic - Algorithms, Techniques and Implementations
Edited by Prof. Elmer Dadios

ISBN 978-953-51-0393-6
Hard cover, 294 pages
Publisher InTech
Published online 28, March, 2012
Published in print edition March, 2012

Fuzzy Logic is becoming an essential method of solving problems in all domains. It gives tremendous impact on the design of autonomous intelligent systems. The purpose of this book is to introduce Hybrid Algorithms, Techniques, and Implementations of Fuzzy Logic. The book consists of thirteen chapters highlighting models and principles of fuzzy logic and issues on its techniques and implementations. The intended readers of this book are engineers, researchers, and graduate students interested in fuzzy logic systems.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hashim Habiballa (2012). Resolution Principle and Fuzzy Logic, Fuzzy Logic - Algorithms, Techniques and Implementations, Prof. Elmer Dadios (Ed.), ISBN: 978-953-51-0393-6, InTech, Available from: http://www.intechopen.com/books/fuzzy-logic-algorithms-techniques-and-implementations/resolution-principle-in-fuzzy-logic

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821