

Resolution Strategies for Fuzzy Description Logic

Hashim Habiballa

Institute for Research and Applications of Fuzzy Modeling
University of Ostrava, 30. dubna 22, 701 03 Ostrava 1, Czech Republic
hashim.habiballa@osu.cz

Abstract

The paper presents experimental comparison of several resolution strategies for reasoning in Fuzzy Description Logic based on Fuzzy Predicate Logic with evaluated syntax. Resolution-based reasoning is established on previous works concerning non-clausal resolution principle both theoretical and application-oriented (FPLGERDS inference engine).

Keywords: Fuzzy inference, Fuzzy Description Logic, Resolution Strategies.

1 Introduction

Description logic (DL) has been intensively studied and it forms an interesting formalism for knowledge representation [3]. Its expressive power is sufficient for applications like semantic web. On the other hand DL is relatively simple in contrast to the full First-order Logic (FOL) bringing some significant properties like decidability. Fuzzy Predicate Logic with Evaluated Syntax (FPL) [12] is a well-studied and wide-used logic capable of expressing vagueness. It has a lot of applications based on robust theoretical background. The knowledge representation itself doesn't lead to full applicable deductive system. It also requires an efficient formal proof theory. Since DL is semantically a subclass of FOL it may use various proved techniques like tableaux algorithm. However the most widely applied resolution principle [4] brings syntactically several obstacles mainly arising from normal form transformation. FPL is associating with even harder problems when trying to use the resolution principle. The solutions to these obstacles based on the non-clausal resolution [2] were already proposed in [8] and [7].

In this article we would like to present a natural integration of these two formal logical systems into Fuzzy Description Logic (FDL). It leads to the refutational resolution theorem prover for FDL ($R RTP_{FDL}$). FDL has been

presented in a simple form in [13] and in more general form in [11] with strong computability and time complexity results. The article refers to some definitions from cited refutational resolution theorem provers for DL ($R RTP_{DL}$) and FPL ($R RTP_{FPL}$).

Another issue addressed in the paper concerns to the efficiency of presented inference strategies developed originally for the proving system. We show 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.

2 General resolution and unification extensions for existentiality

For the purposes of ($R RTP_{FPL}$) we will use generalized principle of resolution, which is defined in the research report [1]. There is a propositional form of the rule defined at first and further it is lifted into first-order logic. We will introduce the propositional form of the general resolution.

General resolution - propositional version

$$\frac{F[G] \quad F'[G]}{F[G/\perp] \vee F'[G/\top]} \quad (1)$$

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 $F[G/\perp] \vee F'[G/\top]$ 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 $F'[G/\top]$ ($F[G/\perp]$).

Revised version of the paper which forms the core of the handbook [2] is closely related with notion of selection functions and ordering constraints. By a selection functions we mean 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.) As an usual ordering can be used lexicographic path 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 \perp are smaller than the other logical symbols. It means the ordering is following $A \succ \exists \succ \top \succ \neg \succ \vee \succ \wedge \succ \top \succ \perp$. 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 a general clauses 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, \dots, A_k in $S(C)$, all atoms of which are true in I . A sequence A_1, \dots, 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 [2]. It is based on the following assumption that a subformula F' in $E[F']$ is *positive* (resp. *negative*), if $E[F'/\top]$ (resp. $E[F'/\perp]$) 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).

When trying to refine the general resolution rule for fuzzy predicate logic and description 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 doesn't 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, \dots) , functions (with n arguments) (f, g, h, \dots) , variables (X, Y, Z, \dots) , predicates (with n arguments) (p, q, r, \dots) , logical connectives $(\wedge, \vee, \rightarrow, \neg)$, quantifiers (\exists, \forall) and logical constants (\perp, \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 $F = G \vee H$	$Sub(F) = \{G, H\}, Sup(G) = Sup(H) = F$ $Pol(G) = Pol(F) = Pol(H)$
$F = G \rightarrow H$	$Sub(F) = \{G, H\}, Sup(G) = Sup(H) = F$ $Pol(G) = -Pol(F), Pol(H) = Pol(F)$
$F = \neg G$	$Sub(F) = \{G\}, Sup(G) = F$ $Pol(G) = -Pol(F)$
$F = \exists \alpha G$ 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, $G = p(t_1, \dots, t_n) \in Sub^*(F)$ 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: $\forall x(\forall xA(x) \rightarrow B(x)) - Qnt(x) = \forall xA(x)$, for x in $A(x)$ and $Qnt(x) = \forall x(\forall xA(x) \rightarrow B(x))$, 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: $F = \forall y(\forall xA(x) \rightarrow B(y)) - x$ 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 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 $G = p(t_1, \dots, t_n)$ and $G' = r(u_1, \dots, u_n)$ be atoms. Most general unifier (substitution mapping) $MGU(G, G') = \sigma$ 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 $(t_1, u_1), \dots, (t_n, u_n)$; If for every pair unifier exists then $MGU(G, G') = \sigma$ obtained during term unification (success state).
3. In any other case unifier doesn't 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 $t' = f(t'_1, \dots, t'_m)$, $u' = g(u'_1, \dots, u'_n)$ are function symbols with arguments and $f = g$ then unifier for (t', u') exists iff unifier exists for every pair $(t'_1, u'_1), \dots, (t'_n, u'_n)$ (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 doesn't exist (fail-state).

$MGU(A) = \sigma$ for a set of atoms $A = \{G_1, \dots, G_k\}$ 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 $(G_1, G_i), \sigma$ 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 [1].

Definition 5 General ordered resolution with selection for first-order logic (GR_{FOL})

$$\frac{F[G_1, \dots, G_k] \quad F'[G'_1, \dots, G'_n]}{F\sigma[G/\perp] \vee F'\sigma[G/\top]} \quad (2)$$

where $\sigma = MGU(A)$ is the most general unifier (MGU) of the set of the atoms $A = \{G_1, \dots, G_k, G'_1, \dots, G'_n\}$, $G = G_1\sigma$. 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 $F\sigma[G/\perp] \vee F'\sigma[G/\top]$. F is called positive and F' is called negative premise, G represents an occurrence of an atom. The expression $F\sigma[G/\perp] \vee F'\sigma[G/\top]$ is the resolvent of the premises on G .
and

(i) either G is selected by S in F' , or else $S(F')$ is empty, G is maximal in F' , (ii) atom G is maximal in F , and (iii) F does not contain a selected atom.

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 1. We present a very simple example of existential variable unification before we introduce the refutational theorem prover for FOL.

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 [12]. 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). Propositional version of the fuzzy resolution principle has been already presented in [6]. 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. It is important that we assume that for every subformula *Sub, Sup, Pol, Lev, Qnt, Sbt, Sig* and other derived properties defined in section 2 hold (where the classical FOL connective is presented the Łukasiewicz one has the same mapping value).

Definition 6 Evaluated proof, refutational proof and refutation degree

An evaluated formal proof of a formula A from the fuzzy set $X \simeq F_J$ is a finite sequence of evaluated formulas $w := a_0/A_0, a_1/A_1, \dots, a_n/A_n$ such that $A_n := A$ and for each $i \leq n$, either there exists an m -ary inference rule r such that

$$a_i/A_i := r^{evl}(a_{i_1}, \dots, a_{i_m})/r^{syn}(A_{i_1}, \dots, A_{i_m}),$$

$$i_1, \dots, i_m < n \text{ or } a_i/A_i := X(A_i)/A_i.$$

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 $A_n := \perp$. $Val(w) = a_n$ is called refutation degree of A .

Definition 7 General ordered resolution with selection for fuzzy predicate logic (GR_{FPL})

$$r_{GR} : \frac{a/F[G_1, \dots, G_k], b/F'[G'_1, \dots, G'_n]}{a \otimes b / F\sigma[G/\perp] \nabla F'\sigma[G/\top]} \quad (3)$$

where $\sigma = MGU(A)$ is the most general unifier (MGU) of the set of the atoms $A = \{G_1, \dots, G_k, G'_1, \dots, G'_n\}$, $G = G_1\sigma$. 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 $F\sigma[G/\perp] \nabla F'\sigma[G/\top]$. F is called positive and F' is called negative premise, G represents an occurrence of an atom. The expression $F\sigma[G/\perp] \nabla F'\sigma[G/\top]$ is the resolvent of the premises on G .

and

(i) either G is selected by S in F' , or else $S(F')$ is empty, G is maximal in F' , (ii) atom G is maximal in F , and (iii) F does not contain a selected atom.

Lemma 1 Soundness of r_{GR}

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

$$\mathcal{D}(r^{syn}(A_1, \dots, A_n)) \geq r^{evl}(\mathcal{D}(A_1), \dots, \mathcal{D}(A_n)) \quad (4)$$

holds true.

Definition 8 Refutational resolution theorem prover

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 \perp, \top (standard equivalencies for logical constants). A refutational proof by definition 6 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 9 Simplification rules for ∇, \Rightarrow

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

Lemma 2 Provability and refutation degree for GR_{FPL}

$T \vdash_a A$
iff $a = \bigvee \{Val(w) \mid w \text{ is a refutational proof of } A \text{ from } LAX \cup SAX\}$

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 $T \models_a A$.

4 Resolution principle for Fuzzy Description Logic

In this section we will naturally build up the Refutational Resolution Theorem Prover for Fuzzy Description Logic $RRTP_{FDL}$ based on resolution rule for FPL. Whole the theory is based on mapping FDL formulas into FPL formulas and hence the properties of interpretation and resolution are preserved. The paper works with standard interpretation structures and rules of FPL [12] and also for ALC-like FDL [11] based on Łukasiewicz algebra. We use atomic concepts to be bound with unary predicates A_1, \dots, A_n and atomic roles are bound with binary predicates R_1, \dots, R_m . Terms are constants a_1, \dots, a_l or variables x_1, \dots, x_k . There is also top and bottom concepts (\top, \perp). Concepts may be constructed using negation \neg , conjunction $\&$, disjunction ∇ , implication \rightarrow , restricted quantification $\forall R.C$ bounded with $\forall y(R(x, y) \rightarrow C(y))$, $\exists R.C$ bounded with $\exists y(R(x, y) \& C(y))$. Instances of concepts could be written in the form $C(t)$, where t is a term. When speaking about instances of quantified concepts its meaning in FPL is equivalent to replacing the occurrence of x variable by t . For the purposes of the prover we extend the notion of roles. At first we enable to use the atomic role negation $\neg R$, which is equivalent to its FPL representation and in semantic meaning it relates to complement operation on the extent of R . We also enable to use negation, conjunction and disjunction of roles and we introduce notion of full role (\top_R) and empty role (\perp_R). We call formulas of FDL - FDL formulas.

Definition 10 Embedding of structural notions into FOL
Let D be a DL formula. We call a formula F of FOL equivalent embedding formula by the following mapping $Emb(D) = F$ (embedding). All the above defined notions for FOL holds also for $Emb(D)$. Additionally we define a mapping Ext (extension) for D , which returns an explicit form of DL-formula with terms assigned to atomic concepts and roles.

Let C, D be concepts, R, S be roles, t, t_1, t_2 be terms, x, y be variables:

Recursive definiton of Ext :

$$Ext(C) = C(x), \quad Ext(R) = R(x, y), \quad Ext(C(t)) = C(t), \\ Ext(R(t_1, t_2)) = R(t_1, t_2) \quad (\text{for atomic concepts and roles})$$

For every $\bullet \in \{\sqcap, \sqcup, \rightarrow\}$ it holds:

$$Ext(C \bullet D) = (C \bullet D), \quad Ext(R \bullet S) = (R \bullet S)$$

$$Ext(\neg C) = \neg C, \quad Ext(\neg R) = \neg R$$

For every $Q \in \{\forall, \exists\}$ it holds:

$$Ext(QR.C) = QR.C,$$

Recursive definiton of Emb :

Let C, D be concepts and R, S be roles:

$$Emb(C(t)) = C(t), \quad Emb(R(t_1, t_2)) = R(t_1, t_2), \quad Emb(A) = A(x), \\ Emb(R) = R(x, y) \quad (\text{atomic concept and role})$$

$$Emb(C \sqcap D) = C \wedge D, \quad Emb(C \sqcup D) = C \vee D, \quad Emb(C \rightarrow D) = C \rightarrow D, \\ Emb(\neg C) = \neg C, \quad Emb(R \sqcap S) = R \wedge S, \quad Emb(R \sqcup S) = R \vee S, \\ Emb(R \rightarrow S) = R \rightarrow S, \quad Emb(\neg R) = \neg R,$$

$$Emb(\forall R.C) = \forall y(X), \quad \text{where } Emb(R.C) = X = R(x, y) \rightarrow C(y),$$

$$Emb(\exists R.C) = \exists y(X), \quad \text{where } Emb(R.C) = X = R(x, y) \wedge C(y),$$

$$Emb(\top) = \top, \quad Emb(\perp) = \perp, \quad Emb(\top_R) = \top, \quad Emb(\perp_R) = \perp.$$

For every DL formula D or DL formula part $R.C = D$, a variable y occurring in D or $R.C$, where $Emb(D) = F$, and formula mappings

$Map \in \{Sub, Sup, Sub^*, Sup^*, Pol, Lev\}$, it holds $Map(D) = Map(F)$. For a variable y in an atomic concept or role (C or R) and F it holds $Qnt(y) = Qnt(\alpha)$, where α is a variable occurring in $Emb(C)$ or $Emb(R)$. Sig and Sbt mappings could be defined subsequently. If not stated otherwise $Sig(\alpha) = 0$. No free variables should occur in $Emb(D)$.

Example 1 Examples of structural mappings

Let $C \rightarrow \exists R.C'$ be a DL formula D . Then there are some notions (not all possible) holding for D :

$$Ext(D) = C(x) \rightarrow \exists R(x, y).C'(y) \quad \text{and}$$

$$Pol(D) = 0, \quad Pol(C) = -1, \quad Pol(\exists R.C') = 1, \quad Pol(C') = 1, \\ Lev(D) = 0, \quad Lev(\exists R.C') = 1, \quad Lev(R.C) = 2, \quad Lev(R) = 3, \\ Sub(\exists R.C') = R(x, y) \rightarrow C'(y), \quad Sub^*(\exists R(x, y).C'(y)) = \\ \{R(x, y) \rightarrow C'(y), R(x, y), C'(y)\},$$

$$Sub(D) = \{C(x), \exists y(R(x, y) \rightarrow C'(y))\}, \quad Sup(C'(y)) = R(x, y) \rightarrow C'(y)$$

$$\text{For } y \text{ in } C'(y) \quad Qnt(y) = \exists y(R(x, y) \rightarrow C'(y))$$

Definition 11 General ordered resolution with selection for Fuzzy Description Logic (GR_{FDL})

Let D and D' be DL formulas projected by Ext_{Max} ,

G_i, G'_i be an atomic concept or role (a subformula of D, D').

$$r_{GR} : \frac{a/D[G_1, \dots, G_k], b/D'[G'_1, \dots, G'_n]}{a \otimes b/D\sigma[G/\circ]\nabla D'\sigma[G/\bullet]} \quad (5)$$

where $\sigma = MGU(A)$ is the most general unifier (MGU) of the set of the atoms, $A = \{G_1, \dots, G_k, G'_1, \dots, G'_n\}$, $G = G_1\sigma$. For G being an atomic concept $\circ = \perp, \bullet = \top$, for G being an atomic role $\circ = \perp_R, \bullet = \top_R$. For every variable α in D or D' , $(Sbt(\gamma) = \alpha) \cap \sigma = \emptyset \Rightarrow (Sig(\alpha) = 1$ in D or D' iff $Sig(\alpha) = 1$ in $D\sigma[G/\circ]\nabla D'\sigma[G/\bullet]$.

and

(i) either G is selected by S in F' , or else $S(F')$ is empty, G is maximal in F' , (ii) atom G is maximal in F , and (iii) F does not contain a selected atom.

Definition 12 Refutational resolution theorem prover for FDL

Refutational non-clausal resolution theorem prover for FDL ($RRTP_{FDL}$) is the inference system with the inference rule GR_{FDL} and sound auxiliary simplification rules. A refutational proof of the goal G from the set of axioms $N = \{A_1, \dots, A_m\}$ is a sequence of DL formulas with full terms on atoms (max. atomic extension) $w := a_0/A_0, a_1/A_1, \dots, a_n/A_n, a/\perp$, where D_i is an axiom from N , $\neg G$ or a resolvent from premises D_k and D_l ($k, l < i$); simplification rules apply to the resolvent. It is assumed that $Sig(\alpha) = 1 \forall \alpha$ variable in $N \cup \neg G$.

Example 2 Proof of child's happiness by r_{GR}

Consider the following knowledge (significantly simplified in contrast to the reality) about child's happiness. We suppose that a child is happy in the degree 0.8 if it has mother and father. It can be expressed in compact FDL-formula form:

$$0.8/[\exists child.female \wedge \exists child.male] \Rightarrow happy$$

Further we suppose that a child is happy in the degree 0.5 if it has a lot of toys (we suppose parents are a bit more important for children). We will present proofs and then we mark the best provability degree from the following axioms. According to the definitions FDL-formulas are projected by Ext_{Max} . It was used the automated theorem prover of the author for classical logic [5]. *Xa.* steps represent application of simplification rules.

Common proof members (axioms):

1. $0.8/[\exists child(X, Y).female(Y) \wedge \exists child(X, Y).male(Y)] \Rightarrow happy(X)$ (happy with parents - 0.8)
2. $0.5/toys(X) \Rightarrow happy(X)$ (happy with toys - 0.5)
3. $1/child(johana, hashim)$ (clear crisp fact)
4. $1/child(johana, lucie)$ (clear crisp fact)
5. $1/male(hashim)$ (clear crisp fact)
6. $1/female(lucie)$ (clear crisp fact)
7. $0.9/toys(johana)$ (johana has a lot of toys - 0.9)
8. $1/\neg happy(johana)$ (negated goal - is johana happy?)

Proof 1:

9. $0.9 \otimes 0.5 / \perp \vee [\top \Rightarrow happy(johana)]$
 - 9a. $0.4 / happy(johana)$ (r_{GR} on 7.2, $X = johana$)
 10. $1 \otimes 0.4 / \perp \vee \neg \top$
 - 10a. $0.4 / \perp$ (r_{GR} on 9.8.)
- (happy(johana) is provable in 0.4)

Proof 2:

9. $0.8 \otimes 1 / [[\exists child(johana, Y).female(Y) \wedge \exists child(johana, Y).male(Y)] \Rightarrow \perp] \nabla \neg \top$
 - 9a. $0.8 / \neg [\exists child(johana, Y).female(Y) \wedge \exists child(johana, Y).male(Y)]$ (r_{GR} on 1.8, $X = johana$)
 10. $0.8 \otimes 1 / \neg [\exists child(johana, lucie). \top \wedge \exists child(johana, Y).male(Y)] \nabla \perp$
 - 10a. $0.8 / \neg [\exists child(johana, lucie). \top \wedge \exists child(johana, Y).male(Y)]$ (r_{GR} on 6.9, $Y = lucie$)
 11. $0.8 \otimes 1 / \neg [\exists child(johana, lucie). \top \wedge \exists child(johana, hashim). \top] \nabla \perp$
 - 11a. $0.8 / \neg [\exists child(johana, lucie). \top \wedge \neg child(johana, hashim). \top]$ (r_{GR} on 5.10, $Y = hashim$)
 12. $0.8 \otimes 1 / \neg [\exists \top_R. \top \wedge \exists child(johana, hashim). \top] \nabla \perp$
 - 12a. $0.8 / \neg [\exists child(johana, hashim). \top]$ (r_{GR} on 4.11.)
 13. $0.8 \otimes 1 / \neg \exists \top_R. \top \nabla \perp$
 - 13a. $0.8 / \perp$ (r_{GR} on 3.12.)
- (happy(johana) is provable in 0.8)

We have stated two different proofs and it is clear that several other proofs could be constructed. Let us note that these proofs either consist of redundant steps or they are variants of Proof 1 and Proof 2, where only the order of resolutions is different. So we can conclude that it is effectively provable that Johana is a happy child in the degree 0.8.

The proof of soundness and completeness of the $RRTP_{FDL}$ is straightforward. Since the FPL embedding is semantically equivalent to the interpretation of FDL formulas, all the properties of $RRTP_{FPL}$ holding for FPL should also hold for $RRTP_{FDL}$.

5 Implementation and efficiency

The author also currently implements the non-clausal theorem prover into fuzzy logic and description logic as an extension of previous prover for FOL (GENERALIZED Resolution Deductive System - GERDS) [5]. Experiments concerning prospective inference strategies can be performed with this extension. The prover called Fuzzy Predicate Logic GENERALIZED Resolution Deductive System - FPLGERDS provides standard interface for input (knowledge base and goals) and output (proof sequence and results of fuzzy inference, statistics).

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"

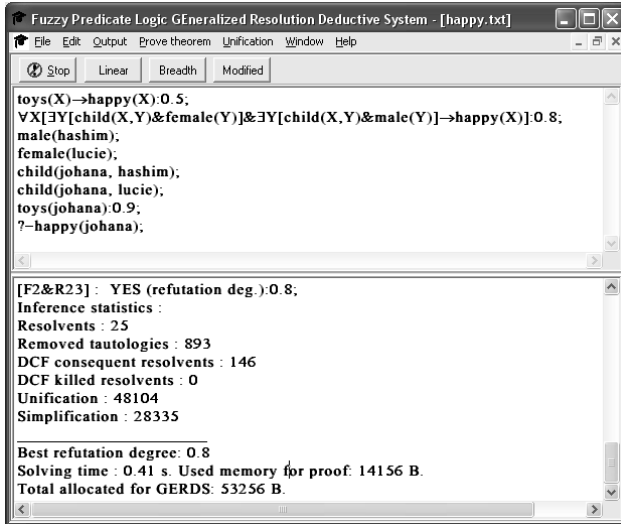


Figure 1: Fuzzy Predicate Logic Generalized Resolution Deductive System

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 empirically very effective. Completeness of such a strategy is also straight-forward in FOL:

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

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:

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

Even the usage of this technique 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: $Sbt(x) = a$, $Res = \neg(\perp \rightarrow \perp) \vee \neg(\top \rightarrow \top) \Rightarrow \perp$

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 and Fuzzy Description 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. This measurements were done using the FPLGERDS application [9]. 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 Description 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" than 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 3 Redundant knowledge base

Consider the following knowledge base (fragment):

-
- $0.51/a \wedge b_1 \Rightarrow z, 0.61/a \wedge b_1 \wedge b_1 \Rightarrow z,$
- $0.71/a \wedge b_1 \wedge b_1 \wedge b_1 \Rightarrow z, 0.81/a \wedge b_1 \wedge b_1 \wedge b_1 \wedge b_1 \Rightarrow z,$
- $0.91/a \wedge b_1 \wedge b_1 \wedge b_1 \wedge b_1 \wedge b_1 \Rightarrow z, 1/b_1,$
-
- $0.52/a \wedge b_2 \Rightarrow z, 0.62/a \wedge b_2 \wedge b_2 \Rightarrow z,$
- $0.72/a \wedge b_2 \wedge b_2 \wedge b_2 \Rightarrow z, 0.82/a \wedge b_2 \wedge b_2 \wedge b_2 \wedge b_2 \Rightarrow z,$
- $0.92/a \wedge b_2 \wedge b_2 \wedge b_2 \wedge b_2 \wedge b_2 \Rightarrow z, 1/b_2,$
-

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 [9]. 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.

Table 1: Proof search algorithms

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

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.

Table 2: DCF heuristics

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

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, 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.

Table 3: Inference strategies

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

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 have tested presented inference strategies on the sample knowledge bases with redundancy level 5 with 20, 40, 60, 80 and 100 groups of mutually redundant formulas (the 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.

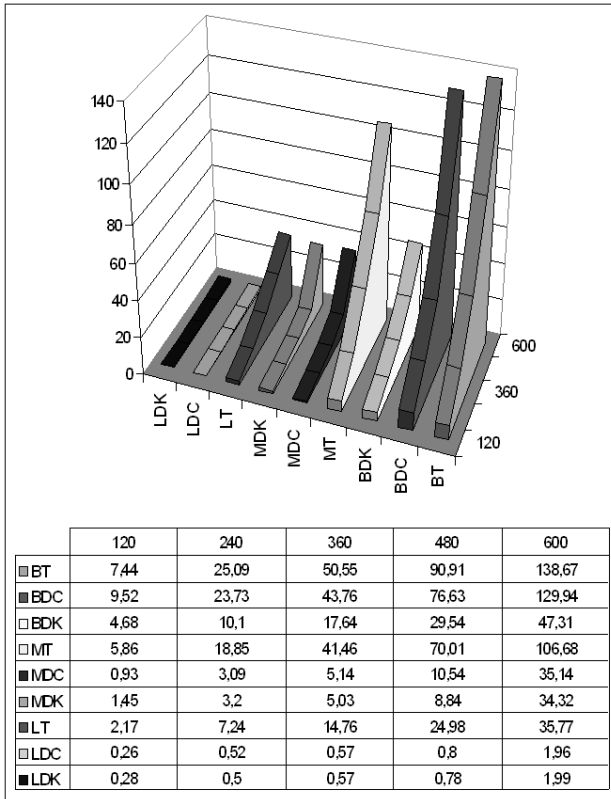


Figure 2: Time complexity for redundancy level 5 (seconds)

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.

6 Conclusion, further research and applications

The *Non-clausal Refutational Resolution Theorem Prover* forms a powerful inference system for automated theorem proving in fuzzy description 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 recent idea of fuzzy description logic is suitable for further extensions with the presented inference rules [11] and also reflects real situations as it could be observed from the example.

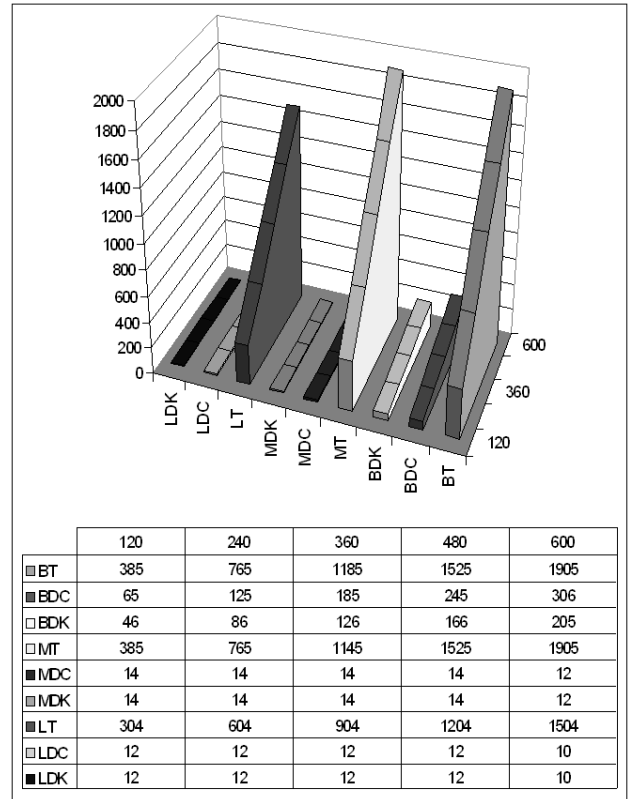


Figure 3: Space complexity for redundancy level 5 (resolvents)

The Detection of Consequent Formulas algorithms family brings significant improvements in time and space efficiency for the best proof search. We have 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 we also use efficient incomplete strategies for fragments of fully general logics.

We have briefly shown 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. There is also prepared the experimental application FPLGERDS, which is obtainable from URL: <http://www1.osu.cz/home/habibal/files/gerds.zip>. The package contains current version of the application, source

codes, examples and documentation.

Acknowledgement

This work was supported by research plan of MSMT CR - MSM 6198898701.

References

- [1] Bachmair, L., Ganzinger, H. A theory of resolution. Technical report: Max-Planck-Institut, 1997.
- [2] Bachmair, L., Ganzinger, H. Resolution theorem proving. In Handbook of Automated Reasoning, MIT Press, 2001.
- [3] Baader et col. (eds.). The Description Logic Handbook - Theory, Interpretation and Applications. Cambridge Univ. Press, 2003.
- [4] Dukić, N., Avdagić, Z. Fuzzy Functional Dependency and the Resolution Principle. In Informatica, Vilnius: Lith. Acad. Sci. (IOSPRESS), 2005, Vol.16, No. 1, pp. 45 - 60, 2005.
- [5] Habiballa, H. Non-clausal resolution - theory and practice. Research report: University of Ostrava, 2000, <http://www.volny.cz/habiballa/files/gerds.pdf>
- [6] Habiballa, H., Novák, V. 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>
- [7] Habiballa, H. Non-clausal Resolution in Fuzzy Predicate Logic with Evaluated Syntax (background and implementation). In Proc. of Intl. Conf. The Logic of Soft Computing IV, Ostrava, pp. 51 - 54, 2005, also available as research rep. at <http://ac030.osu.cz/irafm/ps/rep70.ps.gz>
- [8] Habiballa, H. 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>.
- [9] Habiballa, H. Fuzzy Predicate Logic Generalized Resolution Deductive System. Technical Report, Institute for Research and Application of Fuzzy Modeling, University of Ostrava, 2006.
- [10] Hájek, P. Metamathematics of fuzzy logic. Kluwer Academic Publishers - Dordrecht, 2000.
- [11] Hájek, P. Making fuzzy description logic more general. Fuzzy Sets and Systems 154(2005),pp. 1-15.
- [12] Novák, V., Perfilieva, I., Močkoř, J. Mathematical principles of fuzzy logic. Kluwer, 1999.
- [13] Straccia, U. Reasoning within Fuzzy Description Logics. J. of AI Research 14 (2001), pp. 137-166.