Fuzzy Predicate Logic
Generalized Resolution Deductive System

Hashim Habiballa *

Institute for Research and Applications of Fuzzy Modeling
and Department of Computer Science
University of Ostrava
30. dubna 22
Ostrava, Czech Republic
Hashim.Habiballa@osu.cz
http://www.volny.cz/habiballa/index.htm

Abstract. The article presents the implementation of the refutational resolution theorem proving system for Fuzzy Predicate Logic based on the general (non-clausal) resolution rule. The implementation called Fuzzy Predicate Logic Generalized Resolution Deductive System is intended for Fuzzy Logic with Evaluated Syntax and utilizes also originally developed proof-search heuristics called Detection of Consequent Formulas.

keywords: automated theorem proving, non-clausal resolution, general resolution, automated theorem proving, fuzzy logic.

1 Introduction

The Fuzzy Predicate Logic of First-Order (FPL) forms a powerful generalization of the classical two-valued logic [11]. This generalization brings several hard problems with automated theorem proving especially when utilizing the widely used resolution principle. The resolution-based reasoning in its usually preferred way of application uses the clausal form formulas. In the FPL the standard properties related to the clausal form transformation do not hold. Although there are some attempts to apply the resolution principle in the fuzzy propositonal calculus [10] we will present more general and more straightforward way. We will present the refutational resolution theorem proving system for FPL ($RRTP_{FPL}$) based on general (non-clausal) resolution principle in first-order logic (FOL) [1]. It requires more complex unification algorithm based on the polarity criteria and the quantifier mapping. The below presented idea has its origin in implementation of non-clausal resolution theorem prover [5].

2 General resolution and unification extensions for existentiality

For the purposes of ($RRTP_{FPL}$) we will use generalized principle of resolution, which is defined in the handbook [2].

General resolution

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

where F and F' are formulas - premises of first-order logic and G represents an occurrence of a subformula of F and F'.

When trying to refine the general resolution rule for first-order logic and description logic, it is important to devise a sound and complete unification algorithm. Standard unification

* This work was supported by research project of MSMT - MSM 6198898701

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 the following 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. $Sub^*(F) \supseteq \{H | G \in Sub^*(F) \wedge H \in Sub(G)\}$, $Sup^*(F) \supseteq \{H | G \in Sup^*(F) \wedge H \in Sup(G)\}$

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.

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.

It is clear with respect to the 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. It means the substitution of an existential variable into universal one produces a logically consequent formula. 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 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 $t' = f(t'_1, ..., t'_m)$, $u' = g(u'_1, ..., u'_n)$ 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 doesn't 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 [1].

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

$$\frac{F[G_1, , ..., G_k] \quad F'[G'_1, ..., G'_n]}{F\sigma[G/\bot] \vee F'\sigma[G/\top]} \tag{2}$$

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

Example 1. 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) doesn't 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 [11]. 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). For the purposes of the fuzzy resolution principle extension the Modus ponens rule was considered as an inspiration [4]. 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

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

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 $\mathcal{L}_\mathrm{L}$ 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), $\wedge$ (conjunction), **∇** (Łukasiewicz disjunction), $\vee$ (disjunction), **⇒** (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 standard semantic interpretations
– for every subformula defined above $Sub, Sup, Pol, Lev, Qnt, Sbt, Sig$ and other derived properties defined in section 2 hold (Łukasiewicz connetives 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 [11] namely: inference rule, formal fuzzy theory with set of logical and special axioms, evaluated formal proof.

Definition 6. ***Evaluated proof, refutational proof and refutation degree***
An evaluated formal proof of a formula A from the fuzzy set $X \subsetneq F_J$ 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 7. ***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.

$$T \vdash_a A \text{ iff } a = \bigvee\{Val(w) |\ w \text{ is a proof of } A \text{ from } \mathrm{LAx} \cup \mathrm{SAx}\} \tag{3}$$

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

$$T \models_a A \text{ iff } a = \bigwedge\{\mathcal{D}(A) \,|\, \mathcal{D} \models T\}, \text{ where the condition } \mathcal{D} \models T \text{ holds}$$
$$\text{if for every } A \in \mathrm{LAx} : \mathrm{LAx}(A) \leq \mathcal{D}(A),\ A \in \mathrm{SAx} : \mathrm{SAx}(A) \leq \mathcal{D}(A) \tag{4}$$

Definition 8. ***General resolution for fuzzy predicate logic*** (GR_{FPL})

$$r_{GR} : \frac{a/F[G_1, , ..., G_k],\ b/F'[G'_1, ..., G'_n]}{a \otimes b/F\sigma[G/\bot]\boldsymbol{\nabla} F'\sigma[G/\top]} \tag{5}$$

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.

Definition 9. ***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 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 10. ***Simplification rules for $\boldsymbol{\nabla}, \Rightarrow$***

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

Lemma 1. ***Provability and refutation degree for*** GR_{FPL}
$T \vdash_a A$ *iff* $a = \bigvee\{Val(w)|$ *w is a refutational proof of A from* $\mathrm{LAx} \cup \mathrm{SAx}\}$

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* $T \models_a A$.

Proofs could be found in [6]. Proof of the lemma and completeness theorem are constructive and the soundness reduces to the proof of soundness of r_{GR} by induction on standard inference rule equations.

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. 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 several proofs and then we mark the best provability degree from the following axioms. It was used the automated theorem prover of the author for classical logic [5]. Xa. steps represent application of simplification rules for $\bot$ and $\top$.

Common proof members (axioms):

1. $^{0.8}/\forall X[\exists Y[child(X,Y)\,\&\,female(Y)]$ $\&\,\exists Y[child(X,Y)\,\&\,male(Y)] \Rightarrow happy(X)]$	(happy with parents - 0.8)
2. $^{0.5}/\forall X[toys(X) \Rightarrow happy(X)]$	(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. $^{0.9 \otimes 0.5}/\bot\nabla[\top \Rightarrow happy(johana)]$	
9a. $^{0.4}/happy(johana)$	(r_{GR} on 7.,2., $Sbt(X) = johana$)
10. $^{1 \otimes 0.4}/\bot\nabla\neg\top$	
10a. $^{0.4}/\bot$	(r_{GR} on 9.,8.)

(happy(johana) is provable in 0.4)

Proof 2:

9. $0.8 \otimes 1/[\exists Y[child(johana, Y) \,\&\, female(Y)]$
$\,\&\, \exists Y[child(johana, Y) \,\&\, male(Y)] \Rightarrow \bot]\nabla\neg\top$

9a. $0.8/\neg[\exists Y[child(johana, Y) \,\&\, female(Y)]$
$\,\&\, \exists Y[child(johana, Y) \,\&\, male(Y)]]$ (r_{GR} on 1.,8., $Sbt(X) = johana$)

10. $0.8 \otimes 1/\neg[[child(johana, lucie) \,\&\, \top]$
$\,\&\, \exists Y[child(johana, Y) \,\&\, male(Y)]]\nabla\bot$

10a. $0.8/\neg[child(johana, lucie)$
$\,\&\, \exists Y[child(johana, Y) \,\&\, male(Y)]]$ (r_{GR} on 6.,9., $Sbt(Y) = lucie$)

11. $0.8 \otimes 1/\neg[child(johana, lucie)$
$\,\&[child(johana, hashim) \,\&\, \top]]\nabla\bot$

11a. $0.8/\neg[child(johana, lucie)$
$\,\&\, child(johana, hashim)]$ (r_{GR} on 5.,10., $Sbt(Y) = hashim$)

12. $0.8 \otimes 1/\neg[\top \,\&\, child(johana, hashim)]\nabla\bot$

12a. $0.8/\neg[child(johana, hashim)]$ (r_{GR} on 4.,11.)

13. $0.8 \otimes 1/\neg\top\nabla\bot$

13a. $0.8/\bot$ (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.

4 Implementation

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).

Above mentioned theoretical framework is already implemented in the of a computer application called Fuzzy Predicate Logic Generalized Resolution Deductive System (FPLGERDS). It enables to edit knowledge bases of FPL with evaluated syntax and performing deduction on required goals. The fig. 1 shows GERDS's GUI.

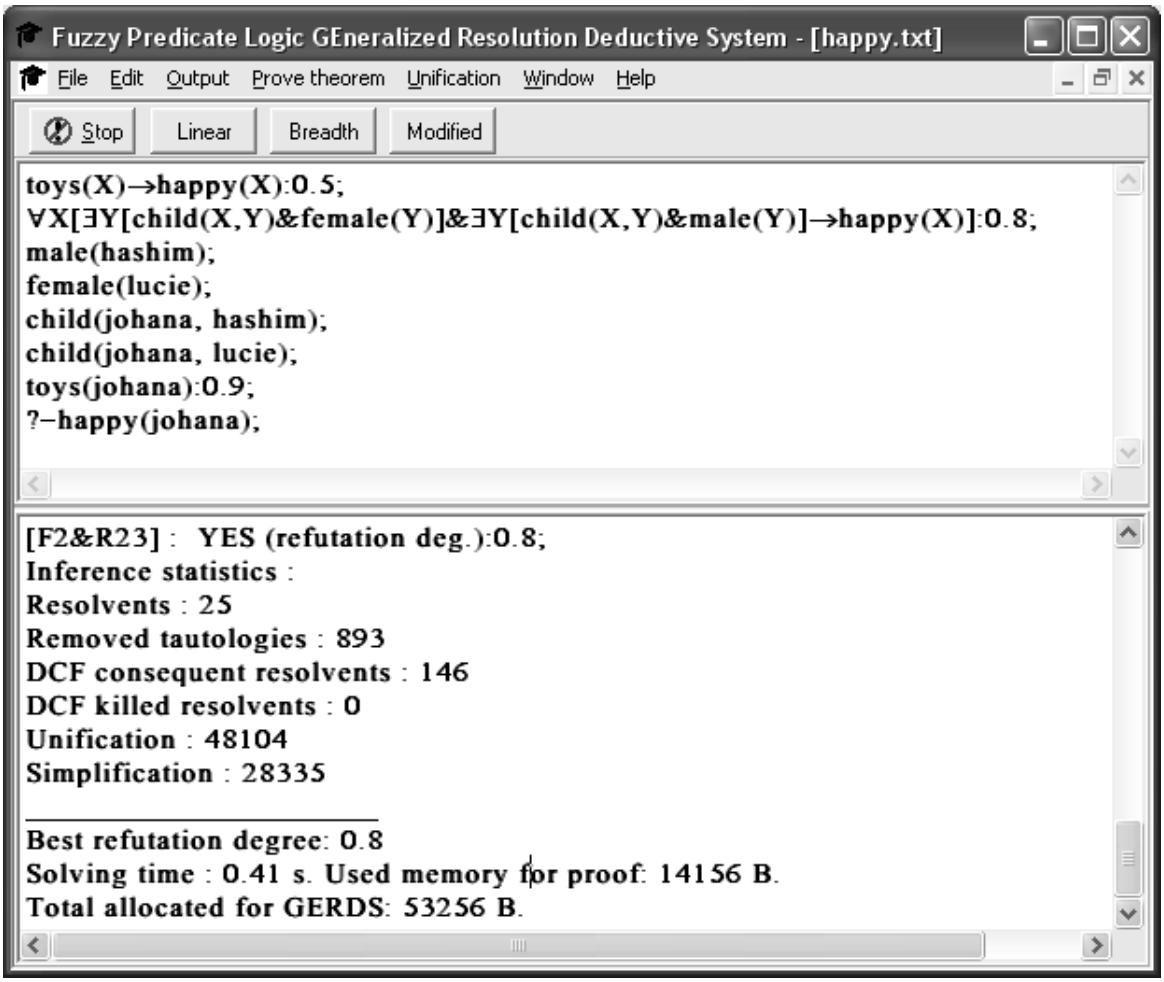

Fig. 1. Frame of GERDS

Axioms are written in common mathematical notion and results of inference provide proof sequence with marked premises, which particular resolvent was derived from. It can present the terms used for unification in goal's variables (PROLOG like). It also offer several types of resolution strategies as visible on the upper panel and several unification and resolution restrictions as well as statistics of inference.

In contrast with clausal resolution prover the implementation of non-clausal prover requires more **complex pointer-based data structures** for internal formula representation. At the first sight it may be observed as a significant disadvantage, since simple data structures of CNF representation are easy to create, store and handle and therefore their algorithms should have reasonable time and space complexity.

It could be observed from the fig. 2, how the formula data structure is constructed. After compilation the syntactical tree is constructed without variable occurrences links to specific

quantifier. The hierarchy of syntactical elements could be specified in Backus-Naur form (for details see [3]). Abstract class TSub represents general subformula of FOL, which has its specifications according to specific type of logical connective (e.g. TCon - conjunction, TImp - implication etc.).

```
TSub = class
    Neg : boolean;  { Flag of logical negation. }
    Ev : shortint;  { Indicator of the subformula logical value. }
    Pol : shortint; { Stores the polarity of the node. }
    L : TSub;
    R : TSub;         { Left and right subtree. }
    Ac : TObject;    { Parent object (logical connective). }
    Q : TQuant;     { Quantifier containing variables at this level. }
    ...
end;
```

Constructed tree is built and linked in three basic levels:

- Object hierarchy - provides standard linkage of parent-child relations.
- Parent connective - enables to quickly evaluate and simplify the tree.
- Atomic level - provides access to atoms (literals) in linear time for simpler general resolution rule application.

Parser is based on LL(1) grammar for FOL and it is constructed using recursive descent parsing technique (RDP). It produces a procedure for every non-terminal symbol according to the string that is contained in production for particular syntactical structure. Advantage of RDP lies mainly in the possibility to perform compilation phase simultaneously with parsing. The algorithm in each non-terminal related procedure performs not only check of syntactical correctness but also stack based creation of the tree (both on logical and term level). In [3] it can be seen which procedures create the tree.

Once the original tree is built the postprocessing phase may continue. Its main aim is to establish links (pointers) to appropriate quantifiers, to evaluate polarities of subformulas and to evaluate infix operators in the formula. This implementation method provides as with **structure containing all necessary information built simultaneously in these two phases**.

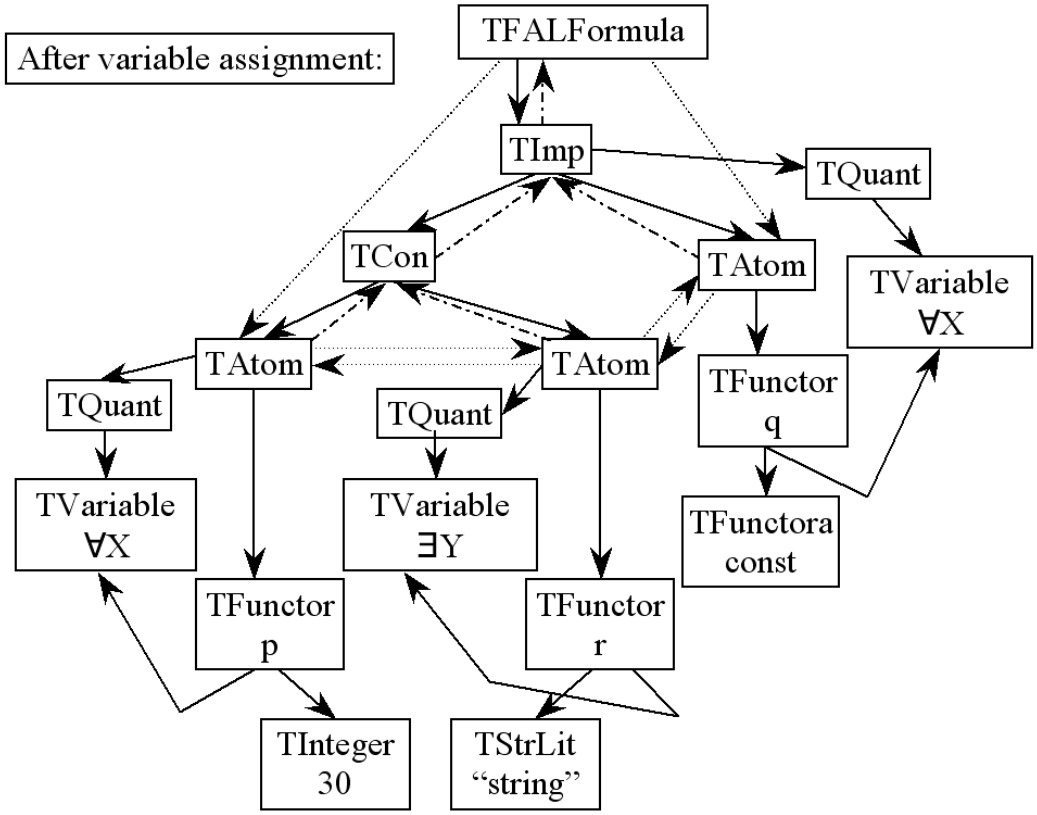

Fig. 2. Example of formula data structure

Inference engine is based on general resolution rule procedure for production of resolvent on two formulas of FOL. The base procedure controlling the process tries to resolve upon all possible premises (it contains also filtrating mechanism of resolution strategies). It checks the consistency of the axiom set with negated query formula. The core procedure performs the general resolution rule. Creation of a resolvent is relatively simple. It requires only creation of premise copies and unification procedure. When then unification exists it is possible to **link copies of premises by disjunction** (TDis object) and to **substitute terms** generated by unification (there is **no** need to **replace all occurrences** but only quantifier object in one memory cell). DCF theorem may use existing general resolution procedure since self-resolution is its special case.

User environment enables to work with two essential panels: Editor and Output. Editor is intended for editing knowledge bases and goals while Output shows results of inference.

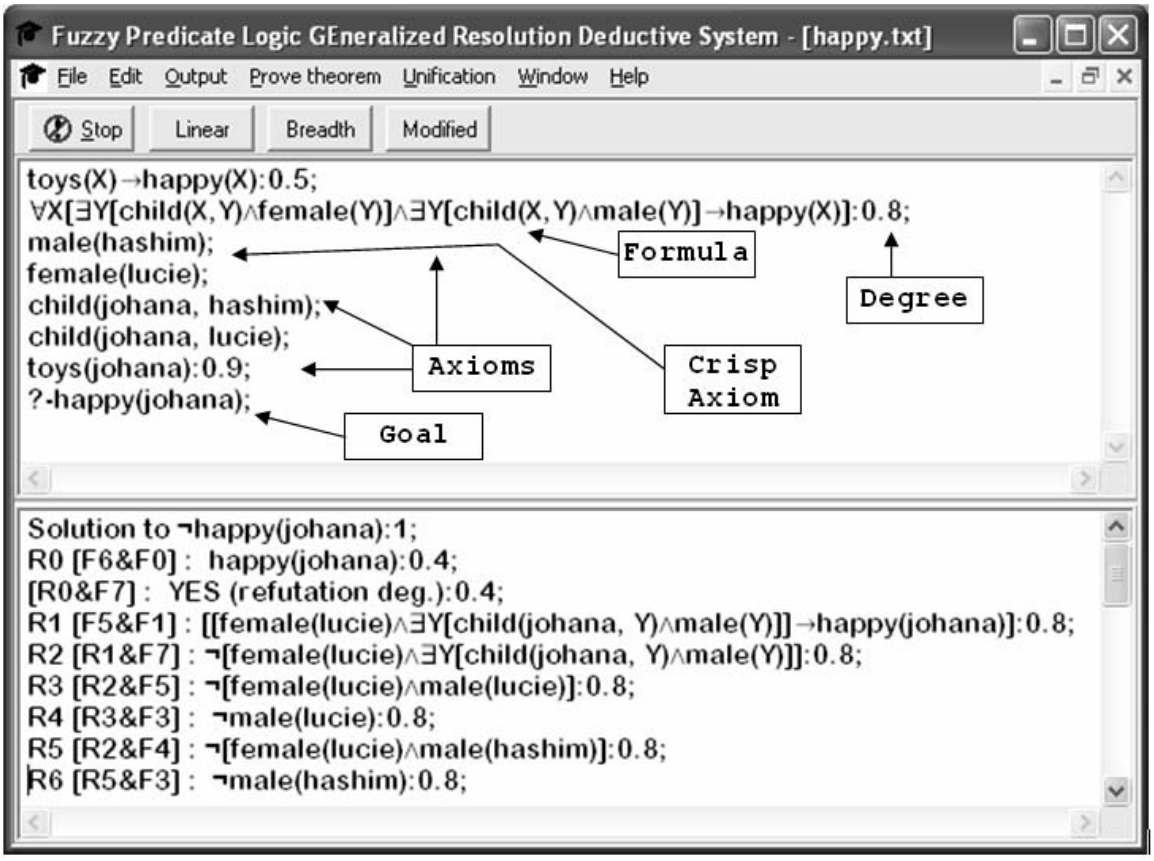

Fig. 3. Structure of Editor Panel

Special axioms and goals have the format described in fig. 3. Every special axiom or goal is followed by semicolon. Special axioms consist of the formula and optional syntactic degree as described in section 3. Syntactic degree ranges from 0 to 1. If no syntactic degree is given then implicit degree of 1 is assigned. A goal cannot have syntactic degree. If all axioms have no degree or their degree is equal to 1, then the inference process degrades to classical two-valued logic.

Output serves for observation of inference results. User can choose from various types of output information. The essential information consists of the axiom and resolvent list together with identification of every formula and identification of source formulas for resolvents. If a refutation is found the word "YES" is printed with appropriate refutation degree.

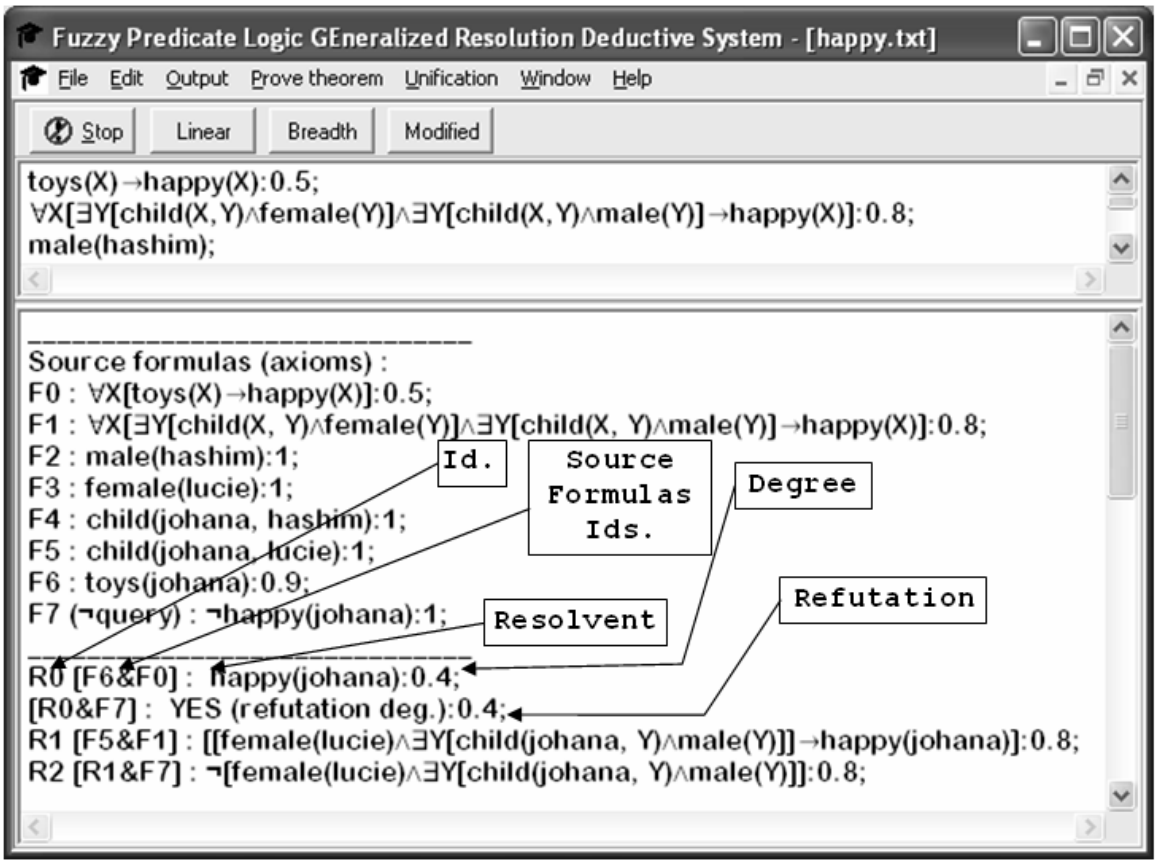

Fig. 4. Structure of Output Panel

Knowledge base file format is plain text where the following table describes ASCII codes of the logical characters. The user font "Frame Logic" (MyFont.ttf) is prepared for screen output based on the code mapping.

Character	ASCII code
$\forall$	247
$\exists$	246
$\rightarrow$	244
$\leftrightarrow$	243
$\neq$	225
$\leq$	223
$\geq$	221
$\neg$	172
$\wedge$	253
$\vee$	252

Tab. 1. Special characters ASCII mapping

5 User settings

FPLGERDS allows user to edit knowledge bases and perform inference upon them. It enables to set up custom preferences which depends mainly on crisp/fuzzy type of the formulas. If completely crisp knowledge base is supported then it is possible to use standard inference strategies like Set of Support strategy while on fuzzy knowledge bases these setting have not sense. Standard strategies are not complete under fuzzy logic.

User menu of the application includes standard items.

- File menu - creating, loading and saving of knowledge bases
- Edit menu - copying, cutting, inserting the text in Editor and Output panel
- Window menu - arranging of the windows (the application allows to work with several knowledge bases simultaneously)

Further user can use special menus for controlling the inference process.
Output menu (output format specifications)

- Axioms - checking on will produce the list of axioms and goals (inputs of inference) at the beginning of output
- Progress - checking on adds every produced resolvent into output
- Sources - includes identification marks for premises of resolution
- Resolvents - produces additional summarization of all resolvents produced during the inference
- Time, Memory - incorporates time and space requirements for inference
- Unsimplified - every resolvent is additionally printed in raw form without any simplification (no application of rewrite rules for $\bot, \top$)
- Statistics - prints at the end of output number of produced resolvents, removed tautologies, DCF consequent resolvents, DCF killed resolvents, unifications and simplifications.
- DCF - adds every successful application of DCF algorithm (with identification of resolvent which caused DCF)
- Interactive - when checked off the output information is printed only when inference finishes and results are stored also to the file named with current knowledge base name and extension + .out; when checked on all results are printed immediately (be aware of outputs longer than 64 kBytes - they will be truncated and it should be used non-interactive setting); when measuring time complexity of the inference use non-interactive setting since printing significantly affects the time (especially for detailed outputs)

Prove theorem (controlling the inference)

- Stop - stops the inference before completion
- DCF limit - sets number of steps before the DCF algorithm will be stopped
- Linear search - starts the inference searching proofs by linear search strategy (not complete for general knowledge bases!)
- Breath-first search - starts the inference searching completely for every proof (complete upon unification with every possible atom)
- Modified linear search - linear strategy starting from not only goal, but every axiom and goal (not complete, but better than linear search)
- Trivial check only - filtrates resolvents by checking their exact symbolic representation
- DCF - performs DCF algorithm (checks if the resolvent is a logical consequence of any previously generated one)
- DCF + Kill - performs DCF and also DCF Kill technique (any added resolvent may "kill" other resolvents - logical consequences)
- Without restr. strategy - no additional inference strategy is applied (use for fuzzy knowledge bases)
- Filtration strategy - Filtration inference strategy is applied (use only for crisp knowledge bases)
- Support set strategy - Support Set strategy is applied (use only for crisp knowledge bases)

Unification (controlling the unification)

- Quantification on - checking off will cause treating of every variable to be universally quantified (ignoring quantifier); use this option for more efficient inference if no existentiality is required
- General cut - every unifiable atom in premises is removed
- Exit on first unused - finishes generation of resolvents on two premises on first atomic formula (use only for crisp knowledge bases - for fuzzy logic such inference is not complete!)
- Exit on first match - finishes generation of resolvents from two premises on first unifiable atomic formula (use only for crisp knowledge bases - for fuzzy logic such inference is not complete!)
- Exit on last match - performs generation of all possible resolvents on all atoms in two premises
- One refutation - searches only for first refutation (use only for crisp knowledge bases - for for fuzzy logic such inference is not complete!)
- All refutations - searches for all possible refutations upon selected strategy

6 FPLGERDS package

The FPLGERDS package is obtainable from the http location:

http://www1.osu.cz/home/habibal/files/gerds.zip

The package could be directly decompressed to any folder and used on Windows 32 platform (Windows 98 and higher, compactibility for Windows 95 is not tested). It produces two folders - Program and Docs. Docs contains the file oldGerds.pdf file with documentation for older version of GERDS intended for two-valued logic and also it contains documentation for current FPLGERDS - FPLGERDS.pdf. Program directory contains executables and sources for FPLGERDS and directory Data with sample knowledge bases. Sources are written in Borland Delphi 3 Object Pascal. The list of files in Program directory:

- About.dfm, About.dcu, About.pas - Information window for the application
- FPLGERDS.dof, FPLGERDS.dpr, FPLGERDS.dsk, FPLGERDS.res - Project files
- FPLGERDS.exe - project executable
- Inference.pas, Inference.dcu - Core unit (no GUI sources)
- MdiEdit.pas, MdiEdit.dcu, MdiEdit.dfm - GUI of knowledge base window
- MdiFrame.pas, MdiFrame.dcu, MdiFrame.dfm - GUI of the basic application interface
- MyFont.ttf - true type font for logical symbols

References

1. Bachmair, L., Ganzinger, H. A theory of resolution. Technical report: Max-Planck-Institut für Informatik, 1997
2. Bachmair, L., Ganzinger, H. Resolution theorem proving. In Handbook of Automated Reasoning, MIT Press, 2001
3. Habiballa, H. Non-clausal resolution - theory and practice. Research report: University of Ostrava, 2000,
   http://www.volny.cz/habiballa/files/gerds.pdf
4. Habiballa, H., Novák, V. Fuzzy general resolution. Research report: Institute for research and applications of fuzzy modeling, University of Ostrava, 2002,
   http://ac030.osu.cz/irafm/ps/rep47.ps

5. Habiballa, H. Non-clausal Resolution Theorem Prover. Research report, No.64: University of Ostrava, 2005,
http://ac030.osu.cz/irafm/ps/rep64.ps.gz
6. Habiballa, H. Non-clausal Resolution Theorem Prover for Fuzzy Predicate Logic. Research report, No.70: University of Ostrava, 2005,
http://ac030.osu.cz/irafm/ps/rep70.ps.gz
7. Habiballa, H. Non-clausal Resolution Theorem Proving for Description Logic. Research report, No.66: University of Ostrava, 2005,
http://ac030.osu.cz/irafm/ps/rep66.ps.gz
8. Hájek, P. Metamathematics of fuzzy logic. Kluwer Academic Publishers - Dordrecht, 2000
9. Hájek, P. Making fuzzy description logic more general. Research report: Institute of Computer Science, Czech Academy of Sciences, 2005
10. Lehmke, S. On resolution-based theorem proving in propositional fuzzy logic with bold connectives. University of Dortmund, 1995
11. Novák, V., Perfilieva, I., Močkoř, J. Mathematical principles of fuzzy logic. Kluwer Academic Publishers, 1999