Symbolic Inference for Non-Horn Knowledge Bases With Fuzzy Predicates

Alexander Sakharov

Abstract. This paper investigates non-Horn knowledge bases with fuzzy predicates. Inference from these knowledge bases excludes reasoning by contradiction, and it is characterized by means of substructural single-succedent sequent calculi with non-logical axioms expressing knowledge base rules and facts. A variety of truth functions can be used for the bodies of knowledge base rules and their contrapositives. Lower bounds of fuzzy truth values of ground literals are obtained by symbolically deriving the literals, building symbolic expressions from derivations trees, and evaluating these expressions.

In memory of Vladimir Gerdt

1. Introduction

The languages of logic programs and knowledge bases (KB) are usually based on first-order logic (FOL) [14]. In non-Horn KBs, facts are literals. Atoms are expressions $P(t_1, ..., t_k)$ where P is a predicate and $t_1, ..., t_k$ are terms. Literals are atoms or their negations. Non-Horn rules are expressions $A \Leftarrow A_1 \wedge ... \wedge A_k$, where $A, A_1, ..., A_k$ are literals. The advantages of non-Horn KB over Horn KBs and normal logic programs are discussed in [16].

KBs and logic programs may include computable (aka evaluable) functions and predicates [10]. They can be implemented as recursive functions in a functional programming language or as algorithms in a procedural programming language. The implementations of evaluable predicates serve to calculate the truth values of their atoms with constant arguments. Evaluable predicates do not have to be boolean, they may yield real numbers interpreted as fuzzy truth values. Recent advances in AI made it possible to implement some predicates as neural networks [4, 18, 17]. These networks yield the fuzzy truth values of atoms of neural predicates with constant arguments.

The principle of Reductio Ad Absurdum (RAA) states that if A is deduced from a hypothesis that is A's complement, then A is derivable. Reasoning by

contradiction, i.e. with using RAA, is not quite adequate for KBs with evaluable predicates [15]. It will be explained later that reasoning by contradiction is not appropriate for KBs with fuzzy predicates either.

We introduce a set of very simple sequent calculi that characterize inference without reasoning by contradiction for non-Horn KBs. Resolution refutations [3] and other derivations steering clear of RAA are mapped to derivations in these calculi. This paper shows how to use symbolic inference methods for the calculation of lower bounds of the truth values of ground literals, i.e. literals without variables. Symbolic derivations of literals in the sequent calculi are the input of the calculation. This calculation is done by building ground symbolic expressions and evaluating them. It is executed in a linear time of the size of the derivations trees.

2. Non-Horn Knowledge Bases With Fuzzy Predicates

A substitution is a finite set of mappings of variables to terms. The result of applying a substitution to a formula or set of formulas is called its instance. We consider inference of ground literals, which are called goals, from non-Horn KBs containing evaluable functions and fuzzy predicates. Evaluable functions and predicates may be partial. Fuzzy truth values are usually real numbers from interval $[0, 1]$. For non-Horn KBs, it is more convenient to use interval $[-1, 1]$ for the representation of truth values. One represents true, minus one represents false. Other real numbers from interval $[-1, 1]$ represent fuzzy truth values.

Terms of evaluable functions with constant arguments are evaluated as soon as they appear in KB derivations. The same applies to atoms of neural and evaluable predicates with constant arguments. The evaluation may not terminate, in which case it is assumed that the truth value is zero. Any complete search strategy for inference from KBs with evaluable and neural predicates should continue and-or search [14] simultaneously with the evaluations including neural computations. If the evaluation of ground atom $A(...)$ yields a positive value above a certain threshold $h > 0$, then $A(...)$ is considered a fact. If the evaluation of this atom yields a negative value below $-h$, then $\neg A(...)$ is considered a fact.

All other predicates will be called derivable. As explained in [16], derivable predicates should be considered partial by default. In the presence of neural predicates, the truth values of ground atoms of derivable predicates should also be real numbers from interval $[-1, 1]$, that is, derivable predicates are fuzzy. We assume that KB facts could be fuzzy. It is expected that truth values lower than one and higher than h are assigned to fuzzy KB facts. One is the default truth value for the other KB facts.

Let $|A|$ denote the truth value of ground literal A. We rely on the traditional definition of the negation truth function for fuzzy KBs: $|\neg A| = -|A|$ [2]. The use of this truth function for negation is limited to the calculation of the truth values of negatibve literals. T-norms are usually considered in the literature as conjunction

truth functions [7]. Other conjunction truth functions may be more appropriate for some KBs. We do not fix the conjunction truth function. The use of this function is limited to the calculation of the truth values of the bodies of KB rules.

Truth functions for disjunctions will not be used here, and the use of implication truth functions will be indirect. The meaning of KB rules is that the truth value of the rule body is a lower bound of the truth value of the head. Given that KB rules are implications and assuming that KB rules are not fuzzy, this semantics of KB rules is consistent with several implication truth functions for t-norms. For the Lukasiewicz, Godel, and product t-norms, $|A \Rightarrow B| = 1$ if $|A| \leq |B|$ [7].

It is explained in [16] why reasoning by contradiction is questionable for KBs containing partial functions or predicates. The same argument applies to KBs containing neural predicates. Consider two KB rules $P \Leftarrow Q$ and $P \Leftarrow \neg Q$. Here is reasoning by contradiction using these rules. Suppose P is false. The first rule implies that Q is false, and hence P is true by the second rule. Now suppose $|P| = 0$. If $|Q| = 0$ as well, then both rules are satisfied, but they do not provide any evidence that P is true or $|P| > 0$ at least.

3. Sequent Calculi

Let $-A$ denote the complement of A, i.e. it is the negation of atom A, and the atom of negative literal A. A sequent is $\Gamma \vdash \Pi$ where Γ is an antecedent and Π is a succedent [11]. Antecedents and succedents are multisets of formulas. KB inference and logic programming are concerned about the derivation of literals, i.e. sequents of the form $\vdash A$ where A is a literal. Consider single-succedent calculi in which formulas are literals. The only structural rule is *cut*.

$$\frac{\Gamma \vdash A \quad A, \Pi \vdash B}{\Gamma, \Pi \vdash B} \; cut$$

These sequent calculi do not have logical axioms. The following rule is the only logical rule. It replaces the standard negation rules and is applicable to axioms.

$$\frac{A, \Gamma \vdash B}{-B, \Gamma \vdash -A} \; swap$$

KB facts and rules can be treated as non-logical axioms [11]. Sequents of the form $\vdash A$ represent facts, and rules are represented by sequents of the form $A_1, ..., A_n \vdash A$ where $A, A_1, ..., A_n$ are literals. Variables can be replaced by any terms in instances of these axioms. The conclusions of *swap* applied to KB rules are known as contrapositives [19].

Definition 1. *L_{cs} is the set of sequent calculus instances in which formulas are literals, succedents contain one literal, the structural rule is cut, the logical rule is swap whose premises are axioms, and non-logical axioms represent KB rules and facts.*

Theorem 1. *L_{cs} is sound and complete with respect to the derivation of ground literals in FOL without RAA.*

Proof. It is proved in [15] that ground literal L is derivable from KB facts and rules in FOL without RAA if and only if $-L$ is refutable by resolution in which the factoring rule is not used and at least one premise of every resolution step is not $-L$ or its descendant. Consider such resolution refutation. The resolution steps that are not ascendants of the endclause are discarded. Let us ground this refutation and then exclude the step that resolves $-L$. There is only one such step because at least one premise of every resolution step is not $-L$ or its descendant. As a result, L is added to every descendant clause of this step including the endclause which becomes L.

Let us traverse this resolution tree bottom-up and map every resolution step to an application of *cut* in L_{cs}. Sequent $\vdash L$ is the conclusion of the last *cut* in the respective L_{cs} derivation tree. The premises of every *cut* in this tree are uniquely determined by the resolution step. The succedent of the *cut* conclusion is also the succedent of the second premise, and the succedent of the first premise is the principal formula of this *cut*. Every leaf node in the L_{cs} derivation tree is an instance of a KB fact, KB rule, or the conclusion of *swap* applied to an instance of a KB rule.

Now consider a ground L_{cs} derivation of sequent $\vdash L$. Every application of *cut* in this derivation corresponds to a resolution step but ground instances of KB rules and facts are used in this resolution derivation instead of the rules and facts. The endclause of this resolution derivation is L.

The lifting lemma [3] states that if clause A is an instance of A', B is an instance of B', and C is the resolvent of A and B, then there is such clause C' that C is its instance, and C' is the resolvent of A' and B'. It is well-known that the lifting lemma can be generalized onto arbitrary resolution derivations: If C is the endclause of a resolution derivation with input clauses $A_1, ..., A_n$ which are instances of $A'_1, ..., A'_n$, respectively, then there is such resolution derivation with input clauses $A'_1, ..., A'_n$ and endclause C' that C is an instance of C'. The proof is a straightforward induction on the depth of resolution derivations.

As a consequence of this generalization of the lifting lemma, there is a resolution tree with the input comprised of KB rules and facts treated as clauses and with such endclause L' that L is its instance. A step resolving L' and $-L$ is added to this derivation. The resolvent of this step is the empty clause, and $-L$ occurs in one premise of the last step only. □

4. Rule Truth Functions

Traditionally, the truth values for conjunction are defined in fuzzy KBs by the following equation: $|A_1 \wedge ... \wedge A_k| = \min\{|A_1|, ..., |A_k|\}$ [2]. With the Godel t-norm (min) as the truth function for KB rule bodies, a lower bound for the truth value of the head of rule $A_0 \Leftarrow A_1 \wedge ... \wedge A_k$ is given by inequality $|A_0| \geq \min\{|A_1|, ..., |A_k|\}$

according to the semantics of KB rules adopted here. Consider the case that $|A_i|$ are positive for $i = 1, ..., j-1, j+1, ..., k$, and $|A_0|$ is negative. As an implication of the semantics of KB rules, $|-A_j| \geq |-A_0|$ in this case. This inequality gives a lower bound for the truth values of the heads of KB rule contrapositives.

If we replace literal truth values with variables, then the right-hand side of the latter inequality can be viewed as the truth function for the body of contrapositive $-A_j \Leftarrow -A_0 \wedge A_1 \wedge ... \wedge A_{j-1} \wedge A_{j+1} \wedge ... \wedge A_k$, i.e. this truth function is defined as: $s'(x_0, x_1, ..., x_{j-1}, x_{j+1}, ..., x_k) = x_0$. For brevity, truth functions for the bodies of KB rules or their contrapositives will be called rule and contrapositive functions, respectively. It was possible to obtain the contrapositive function because the rule function is defined by such symbolic expression $s(x_1, ..., x_k)$ that inequality $x_0 \geq s(x_1, ..., x_k)$ is solved for x_j under the conditions: $x_1 > 0, ..., x_{j-1} > 0, x_{j+1} > 0, ..., x_k > 0, x_0 < 0$.

For KBs conatining fuzzy predicates, another truth function for the conjunctions that are KB rule bodies could give more accurate lower bounds of the truth values of the respective rule heads. Linear functions do not seem a good choice for rule truth functions because they may yield positive values even when one argument is zero. This reason applies to the Lukasiewicz t-norm too [7].

The product t-norm [7] does not look like a good choice either. The product t-norm is defined as the product of the truth values of literals in a conjunction provided that the truth values are in interval $[0, 1]$. For example, $0.6 * 0.6 = 0.36$. Linearly projecting these values into interval $[-1, 1]$, we would get the lower bound $|A_0| \geq -0.28$ for rule $A_0 \Leftarrow A_1 \wedge A_2$ and $|A_1| = |A_2| = 0.2$. The estimate 0.36 makes sense in a probabilistic setting, but the corresponding estimate for non-Horn KB rules is useless.

In contrast to the product t-norm, the following truth function for non-Horn KB rule bodies is more reasonable:

$s(x_1, ..., x_k) = \sqrt[k]{(x_1 + 1)...(x_k + 1)} - 1$

where k is the number of literals in the rule body. Given this rule function, a lower bound for the truth values of the heads of KB rule contrapositives could be obtained by solving the inequality $|A_0| \geq \sqrt[k]{(|A_1| + 1)...(|A_k| + 1)} - 1$ for $|-A_j|$:

$|-A_j| \geq 1 - (1 - |-A_0|)^k / ((|A_1| + 1)...(|A_{j-1}| + 1)(|A_{j+1}| + 1)...(|A_k| + 1))$

Again, replacing literal truth values with variables in the right-hand side of this inequality defines the contrapositive function associated with s.

Rule functions could be parametrized. For example, they could be parametrized by weights assigned to predicates. Also, custom functions yielding lower bounds for the truth values of rule heads could be defined for particular KB rules as it is done in Sugeno KBs [2]. Let $w(A)$ be the weight assigned to the predicate of literal A. Here is an example of a parametrized truth function:

$s(x_1, ..., x_k) = (x_1 + 1)^{w(A_1)/w} ... \ (x_k + 1)^{w(A_k)/w} - 1$

where s is defined for rule $A_0 \Leftarrow A_1 \wedge ... \wedge A_k$, and $w = w(A_1) + ... + w(A_k)$.

Definition 2. *Rule or contrapositive truth function s is called proper if* $s(1,...,1) = 1$, $s(0,...,0) = 0$, *and for* $j = 1...k$, $h \leq s(x_1,...,x_j,...,x_k) \leq s(x_1,...,x'_j,...,x_k)$ *if* $x_j \leq x'_j$ *and* $x_1 \geq h,...,x_k \geq h$.

It is easy to verify that the rule and contrapositive functions specified earlier satisfy the conditions of this definition.

5. Truth Value Approximation

Let $\alpha\{b_1 \to \beta_1,...,b_j \to \beta_m\}$ denote the substitution of term β_i for all occurrences of variable b_i in term α for $i = 1,...,m$. Let us define symbolic expression (term) $n(\tau)$ recursively for all ground derivations τ. In the following definition, lower-case letters are variables. These variables correspond to the same named upper-case ground literals.

- If τ is ground instance A of a KB fact, then $n(\tau) = |A|$ ($|A|$ is a constant).
- If τ is a ground instance $A_0 \Leftarrow A_1,...,A_k$ of KB rule and s is the truth function for this KB rule, then $n(\tau) = s(a_1,...,a_k)$.
- If the last rule of τ is *swap* with the conclusion $A_0, A_1,...,A_{j-1},A_{j+1},...,A_k \vdash A_j$ and s' is the truth function for the respective contrapositive, then $n(\tau) = s'(a_0,a_1,...,a_{j-1},a_{j+1},...,a_k)$.
- If the last rule of τ is *cut* with premises $A_1,...,A_k \vdash E$ and $E,C_1,...,C_m \vdash D$, then $n(\tau) = n(\nu)\{e \to n(\mu)\}$. Here, μ and ν are the parts of τ whose endsequents are the first and second premise of this *cut*, respectively.

Theorem 2. *If τ is a ground L_{cs} derivation of literal G and all rule and contrapositive functions are proper, then* $|G| \geq n(\tau) \geq h$.

Proof. By a straightforward induction of the depth of derivations, the only variables occurring in $n(\tau)$ are the variables corresponding to literals in the antecedent of the endsequent of τ. Consequently, $n(\tau)$ does not contain variables for any derivation τ with the endsequent $\vdash G$. All functions occurring in any $n(\tau)$ are increasing with respect to every argument. If $n(\tau)$ is treated as a function of the variables occurring in it, $n(\tau)$ is increasing with respect to every argument. It is proved by a straightworward induction on the depth of $n(\tau)$.

The value of any function from $n(\tau)$ is greater or equal to h if all arguments of this function are greater or equal to h. By induction on the depth of $n(\tau)$, the value of $n(\tau)$ is greater or equal to h if the value of every variable and every constant in it is greater or equal to h. Since the only constants in any $n(\tau)$ are the truth values of ground instances of KB facts, $n(\tau) \geq h$ for any τ whose endsequent is $\vdash G$.

Now we will prove by induction on the depth of derivations that if $A_1,...,A_k \vdash D$ is the endsequent of derivation μ, then $|D| \geq n(\mu)\{a_1 \to |A_1|,...,a_k \to |A_k|\}$. As a corollary, $|G| \geq n(\tau)$.

Base: The depth of derivation μ is zero. If the endsequent of μ is $\vdash D$, then D is an instance of a KB fact, and the above inequality holds. If the endsequent

of μ is $A_1, ..., A_k \vdash D$, then this sequent is a KB rule instance, it does not contain constants, and the above inequality holds due to the definition of proper functions.

Induction step. Suppose the inequality under consideration is satisfied for all derivations whose depth is less or equal n. Suppose the depth of μ is $n+1$. If the last rule in μ is *swap*, then its premise is a ground instance of a KB rule, μ does not contain constants, and again, the inequality holds due to the definition of proper functions.

Now let the last rule in μ be *cut*, the first premise of this *cut* be $B_1, ..., B_k \vdash C_1$, and the second premise be $C_1, ..., C_m \vdash D$. If γ is the derivation ending in $B_1, ..., B_k \vdash C_1$ and δ is the derivation ending in $C_1, ..., C_m \vdash D$, then $|C_1| \geq n(\gamma)\{b_1 \to |B_1|, ..., b_k \to |B_k|\}$ and $|D| \geq n(\delta)\{c_1 \to |C_1|, ..., c_m \to |C_m|\}$ by the induction assumption. Due to the monotonicity of n with respect to every variable, $|D| \geq n(\delta)\{c_1 \to n(\gamma)\{b_1 \to |B_1|, ..., b_k \to |B_k|\}, ..., c_m \to |C_m|\}$. By the definition of n, $n(\mu) = n(\delta)\{c_1 \to n(\gamma)\}$. Hence, $n(\delta)\{c_1 \to n(\gamma)\{b_1 \to |B_1|, ..., b_k \to |B_k|\}, ..., c_m \to |C_m|\} = n(\mu)\{b_1 \to |B_1|, ..., b_k \to |B_k|, ..., c_m \to |C_m|\}$. □

This theorem establishes that $n(\tau)$ is a conservative approximation of the truth value of G. The proof of Theorem 1 shows that resolution refutations without factoring can be transformed to L_{cs} derivations in a single preorder traversal of the resolution derivations. Therefore, the time complexity of this transformation is linear in the size of the derivations. We focus on resolution methods because they are known to be more efficient.

It is clear from the proof of Theorem 2 that the calculation of a lower bound of $|G|$ can be done in a single postorder traversal of the derivation tree. We assume that the time complexity of algorithms implementing rule and contrapositive functions is linear in the number of function arguments. It is usually possible to implement such algorithms approximating these functions. Consequently, the calculation of a lower bound of $|G|$ takes a linear time of the size of G's derivation in L_{cs}.

Note that lower bounds of the truth values of derived ground literals could not be expressed via terms like n in the presence of RAA. RAA steps eliminate literal $-A$ from sequents $... - A ... \vdash A$. The inequality from Theorem 2 for this sequent has the form $|A| \geq n(...)$ where $n(...)$ contains $|-A|$. This inequality does not give a lower bound for $|A|$.

It is feasible to get multiple derivations of the same goal. These derivations of one literal may give various approximations of the truth value of this literal. It may be beneficial to skip some fact instances with truth values close to h during the derivation process. The design of efficient inference methods capturing higher truth values is beyond the scope of this paper. Investigation of the applicability of non-proper truth functions is a topic for future research.

6. Related Work and Discussion

An overview of KB inference methods including resolution-based methods can be found in [14]. Resolution methods [3] are well suited for inference from non-Horn KBs. Ordered resolution is recognized as one of the most efficient inference methods [1]. It is used in modern theorem provers [8]. Ordered resolution has been adapted to inference from non-Horn KBs without RAA [15].

Like L_{cs}, LK_{-c} calculi from [16] contain non-logical axioms representing KB rules and facts. LK_{-c} calculi characterize inference of literals from non-Horn KBs without using RAA. Those calculi have the same inference power as L_{cs} but they employ standard negation rules as opposed to the swap rule, they allow multiple literals in succedents. LK_{-c} derivations cannot be directly used for the approximation of fuzzy truth values.

Our method is quite different from fuzzy KB systems [2], it does not involve fuzzification or defuzzification. Forward chaining normally serves as the inference mechanism for fuzzy KBs [2]. KB inference without RAA is more powerful than the forward application of Modus Ponens in chaining. For non-Horn KBs with neural and evaluable predicates, symbolic inference is done first. After that, a symbolic expression denoting a truth value is built from the derivation tree. Finally, this expression is evaluated.

Non-Horn KBs with fuzzy predicates are similar to possibilistic logic [5] in the sense that in both of them real numbers are associated with derived ground literals. A survey of fuzzy proof theories in which numbers indicating truthness are attached to FOL formulas is presented in [6]. The major difference of our approach is that literals are the only FOL formulas involved in the KB formalism considered here. Instead of applying fuzzy truth functions to FOL formulas [7], we propagate constraints on the truth values of literals.

The neural-symbolic method from [13] utilizes weighted real-valued functions for calculating lower and upper bounds of the truth values of FOL formulas. Inference is implemented as alternating upward and downward passes over the structure of the formulas. Truth value bounds are adjusted during these passes. Modus Ponens and Modus Tollens are used to update truth value bounds. In our work, sequents play the role of premises of Modus Ponens, and the swap rule can be viewed as a form of Modus Tollens.

ProbLog [12] extends Prolog by associating probabilities with facts. It is assumed that all ground instances of a non-ground fact are mutually independent and share the same probability. ProbLog engines calculate approximate probabilities for inference goals. Non-Horn KBs with neural and evaluable predicates are not probabilistic, they are based on fuzzy logic [7]. DeepProbLog [9] extends ProbLog by allowing neural networks to be associated with facts instead of probabilities. The probabilities of ground instances of a fact are calculated by the neural network associated with the respective predicate. In contrast, we interpret the output of neural networks as fuzzy truth values of ground facts.

References

[1] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In *Handbook of automated reasoning*, pages 19–99. Elsevier, 2001.

[2] Laécio Carvalho de Barros, Rodney Carlos Bassanezi, and Weldon Alexander Lodwick. *A first course in fuzzy logic, fuzzy dynamical systems, and biomathematics: theory and applications*. Springer, 2017.

[3] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic logic and mechanical theorem proving*. Academic press, 1973.

[4] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *International Conference on Learning Representations*, 2019.

[5] Didier Dubois and Henri Prade. Possibilistic logic-an overview. In *Computational logic*, pages 197–255, 2014.

[6] Siegfried Gottwald. *A treatise on many-valued logics*. Research Studies Press, 2001.

[7] Petr Hájek. *Metamathematics of fuzzy logic*. Springer Science & Business Media, 2013.

[8] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer, 2013.

[9] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, 298:103504, 2021.

[10] William McCune. Otter 3.3 reference manual and guide. Technical report, Argonne National Lab., 2003.

[11] Sara Negri and Jan Von Plato. *Structural proof theory*. Cambridge University Press, 2001.

[12] Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015.

[13] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

[14] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd edition, 2009.

[15] Alexander Sakharov. Inference methods for evaluable knowledge bases. In *Software Engineering Application in Informatics*, Lecture Notes in Networks and Systems, pages 499–510. Springer, 2021.

[16] Alexander Sakharov. A logical characterization of evaluable knowledge bases. In *14th International Conference on Agents and Artificial Intelligence*, pages 681–688, 2022.

[17] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

[18] Luciano Serafini and Artur S. d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In *Neural-Symbolic Learning and Reasoning*. CEUR-WS.org, 2016.

[19] Mark E Stickel. A Prolog technology theorem prover: a new exposition and implementation in Prolog. *Theoretical Computer Science*, 104(1):109–128, 1992.

Alexander Sakharov
Synstretch
Framingham, Massachusetts, USA
e-mail: mail@sakharov.net