A Probabilistic Logic for Multi-source Heterogeneous Information Fusion

T.C. Henderson, R. Simmons, and D. Sacharny
School of Computing
University of Utah
Salt Lake City, UT, USA
Email: tch@cs.utah.edu

A. Mitiche
INRS, Montreal, Canada
Email: mitiche@emt.inrs.ca

X. Fan
Nanyang Technological University
Singapore
Email: xyfan@ntu.edu.sg

***Abstract*— We investigate methods to define a probabilistic logic and their application to multi-source fusion problems in geospatial decision support systems[1]. We begin with a discussion of augmenting propositional calculus with probabilities. Given a set of sentences, S, each with a known probability, the problem is to determine the probability of a query sentence that is a disjunction of literals appearing in S. First, we examine Nilsson's [19] solution based on the semantic models of the sentences; we develop two different approaches to solving the problem as posed: (1) using a linear solver, and (2) geometrically finding the intersection of a line with the probability convex hull. Nilsson's approach provides lower and upper bounds on the solution. We then propose a new approach which finds probabilities for the atoms found in the sentences, and then uses these probabilities to compute the probability of the query sentence. Finally, we describe how this probability representation method can form the basis for a probabilistic logic system to support a multi-source knowledge base for decision support.**

I. Introduction and Background

Given the uncertainty associated with statements about the world, or with sensor data, or hypothesis formation in general, it is important to be able to represent such uncertainty appropriately for each source of information, and to be able to combine those disparate types of uncertainty in a consistent manner. For example, sensor data may have associated Gaussian or other types of noise models, while computational processes may have algorithmic uncertainty due to truncation errors, roundoff errors, etc. Logical sentences have commonly been used to represent knowledge, and it is useful to associate an uncertainty with such sentences.

Here we address the problem of finding a suitable representation for uncertainty associated with logical sentences. Although several approaches have been proposed in the past (see [2], [10], [16], [18], [20], [23]), they generally have some significant drawbacks. Usually, these have to do with the computational complexity of the semantics of the sentences (i.e., finding the set of consistent truth assignments is exponential in the number of sentences). There have been some attempts to address these issues, but even those generally have algorithmic issues. For example, Markov Logic Networks solve the problem by creating a Markov network; however, this method is exponential in the number of maximal cliques in the graph [7], and usually MCMC sampling is used to estimate a solution. More recently, Gogate and Domingos [14] have proposed probabilistic theorem proving (PTP):

> by reducing it to lifted weight model counting. *Model counting* is the problem of determining the number of worlds that satisfy a KB (knowledge base of sentences). *Weighted* model counting can be defined as follows. Assign a weight to each literal, and let the weight of a world be the product of the weights of the literals that are true in it. Then weighted model counting is the problem of determining the sum of the weights of the worlds that satisfy a KB.

This approach is based on Nilsson's early work on probabilistic logic [19] (for more details on Nilsson's approach, see Section 2). In particular, they use Nilsson's framework in which the probability of a query formula is equal to the sum of the probabilities of the worlds that satisfy it, and give a formula:

$$P(T \mid K) = \frac{\sum_x 1_T(x) \prod_i \Phi_i(x)}{Z(K)},$$

where $P(T \mid K)$ is the probability of the query, T, given the knowledge base of sentences, $1_T(x)$ is the characteristic function for when the query is true in a possible world, and $\Phi_i(x)$ results from using a set of potential functions (see [14] for details on this) to estimate the probability of the possible world (i.e., $P(x)$), and $Z(K)$ is a normalizing factor found by a weighted model counting method. This method also requires exploring the semantic models for the $KB \cup \{T\}$, or using Monte Carlo to sample that space.

We describe here an alternative approach which avoids the computation of the semantic models, and provides a solution for $P(T \mid K)$. Basically, this is done by exploiting the probability of a disjunctive clause, and developing a set of equations from the sentences and their probabilities, and then solving those equations (where the number of equations is equal to the number of sentences).

[1]This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0077.

II. Nilsson's Proposed Method for Probabilistic Logic

Our approach to probabilistic logic starts with an analysis of Nilsson's method [19] (note that Hailperin [15] gives an in-depth description of this method which was first proposed in the 1800's by George Boole [8]). Given a set of n sentences, $S = \{S_1, S_2, \ldots, S_n\}$, in the propositional calculus, where $\{S_1, \ldots, S_{n-1}\}$ is the KB and S_n is the query, he first finds the set of models of the sentences (i.e., the set of truth value assignments to the sentences that are consistent). Figure 1 shows the general semantic tree [17] for a set of sentences. A simple example given by Nilsson

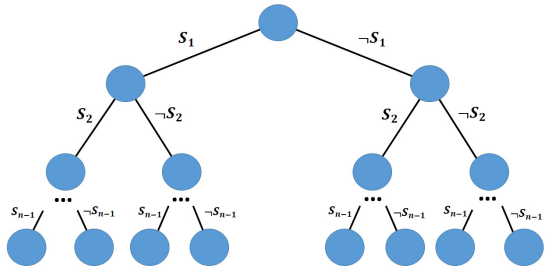

Fig. 1. The General Semantic Tree for a Set of Sentences.

is shown in Figure 2 which is the semantic tree for the sentences $S = \{S_1, S_2, S_3\} = \{P, \neg P \vee Q, Q\}$; we will call this the **Modus Ponens Problem**. That is, show that $KB = \{S_1, S_2\} \models S_3$. This set of models is formed (as columns) into a matrix, V:

$$V = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix},$$

where V is determined by expanding the semantic tree of all possible combinations of truth assignments to the sentences; note that the determination of the models of a set of sentences has computational cost exponential in the number of sentences or $O(2^{|S|})$, where $\mid S \mid = n$. Each row, i, of V gives the truth assignment of sentence S_i in the models.

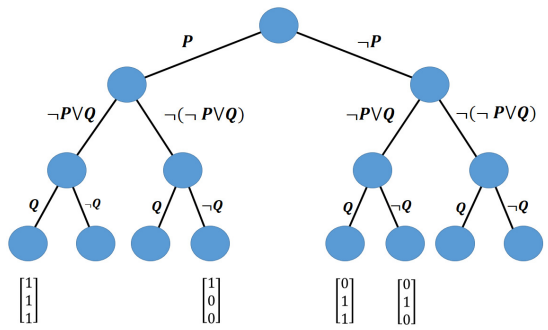

Fig. 2. The Semantic Tree for $S = \{P, \neg P \vee Q, Q\}$.

Nilsson exploits the relation between the probability of the sentences, Π, and the probabilities of the models of the sentences, P, by means of the truth value models themselves, V. Π is an $n \times 1$ vector, P is an $m \times 1$ vector, and V is an $n \times m$ array such that:

$$\Pi = VP.$$

Note that each column of V is a distinct possible world (model) for the sentences.

In order to determine $\Pi(n)$, the probability of the last sentence. Nilsson proposes to solve:

$$\Pi' = V'P,$$

where Π' is the first $n-1$ elements of Π, and V' is the first $n-1$ rows of V. In addition, to impose the constraint that $\sum_{i=1}^{m} P(i) = 1$, a new first element, 1, is added to Π', and an all 1 first row is added to V' (this specifies that *True* is true in all models). We can now solve for P. However, this system is generally under-determined and can be severely so for large n. Nilsson provides some ways to overcome this, but the computational complexity of the approach makes solving for $n > 10$ difficult.

A. Linear Solvers for $\Pi' = V'P$

A method is now given which can find the lower and upper bounds for the probability of the query sentence using standard linear solvers (in this case *lsqlin* in Matlab). Given the matrix V, and a set of probabilities, Π, for the n sentences (i.e., $n-1$ from the knowledge base and the n^{th} sentence which is the query), do the following:

1) Add a row of 1's as the first row of V; call this new matrix V_{aug}.
2) Add a 1 as the first element of Π; call this vector Π_{aug}. This means that $\Pi_{aug}(k)$ is $\Pi(k-1)$ for $k = 2 \ldots n+1$.
3) Set $\Pi_{aug}(n+1)$ to 0.
4) For $y = 0$ to 1 in steps of 0.01, set $\Pi_{aug}(n+1) \leftarrow y$, and solve:

$$\Pi_{aug} = V_{aug}P.$$

The first two steps ensure that the probabilities of the possible worlds sum to 1. The last step produces a solution for each guess for the value of the probability of the query sentence. Here we use *lsqlin* which returns a zero probability for one of the world models when the guess value, y, is not a valid solution. For example, in the **Modus Ponens Problem**, we find $[0.4, 0.69]$ as the bounds for the probability of the query sentence, $\Pi(S_{n+1})$. The upper bound should be 0.7, and the difference is due to the step size for y. Alternatively, constraints can also be added to the Matlab *lsqlin* call in order to find the bounds; this computation and returns $[0.4, 0.7]$ as the upper and lower bounds.

Summary: The basic method proposed by Nilsson does not provide a unique solution for the probability of the query sentence since the system is under-determined. Moreover, the computation of the semantic tree is exponential in the number of sentences.

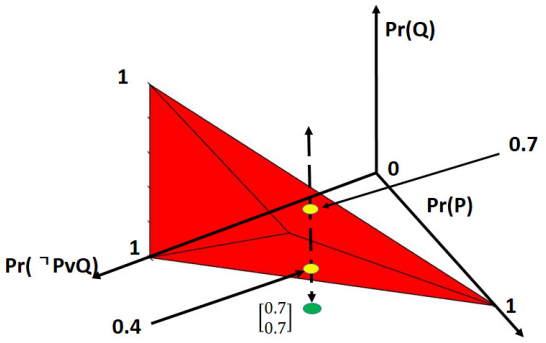

Fig. 3. The Convex Hull for the Modus Ponens Semantic Tree.

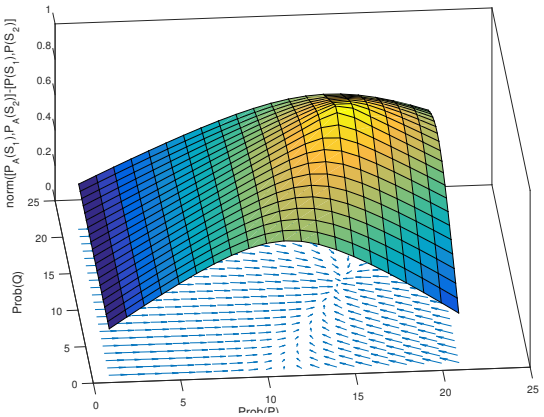

Fig. 4. Error in Known Sentence Probabilities and Sentence Probabilities based on Atom Probabilities.

B. A Geometric Approach to Find Lower and Upper Bounds

Nilsson also points out that consistent probability assignments lie within the convex hull of the semantic vectors (columns of V). Figure 3 shows the convex hull for the set of vectors in the **Modus Ponens Problem**. The axes are the probabilities of the sentences; the figure shows that the bounds on the probability of the query (i.e., $\Pi(S_n)$) can be found by intersecting the line parallel to the probability of the $P(Q)$ axis (and in general, the axis for the last sentence or query) that goes through the lower dimensional point in the space of the probabilities of the sentences whose probabilities are known, and the convex hull of the V column points. The figure indicates that these bounds are 0.4 for the lower probability and 0.7 for the upper bound. The results found here were produced using Matlab.

III. A New Method for Probabilistic Logic

We propose an alternative approach that overcomes the complexity issue. First, we assume that the sentences are given in conjunctive normal form. This means that each sentence is a disjunct of literals (an atom or its negation). Our second assumption is that P and Q are independent random variables. In this case, $P(P \wedge Q) = P(P \mid Q)P(Q) = P(P)P(Q)$; note that if this assumption is violated, our methods also allow the bounds on the probability to be determined. Next, we determine the set of logical atoms (i.e., variables) in S; let $V = \{V_1, V_2, \ldots, V_k\}$ be this set. In this case the probability of a sentence can be computed from the probability of the literals as follows:

$$P(L_1 \vee L_2 \vee \ldots \vee L_p) =$$
$$P(L_1) + P(L_2 \vee \ldots \vee L_p)$$
$$-P(L_1)P(L_2 \vee \ldots \vee L_p),$$

where the probabilities of clauses on the right hand side are computed recursively.

Assuming that the logical (random) variables are independent, each sentence gives rise to a (usually) nonlinear equation defined by the recursive probability of the disjunctive clause as defined above. The resulting set of equations can be solved using standard nonlinear solvers (e.g., *fsolve* in Matlab), and a set of consistent values for the probabilities of the atoms determined.

Consider Nilsson's example: $S_1 = P$, $S_2 = \neg P \vee Q$, and $S_3 = Q$ with $\Pi(S_1) = \Pi(S_2) = 0.7$. The resulting equations for *fsolve* are:

$$F(1) = -0.7 + x_1$$
$$F(2) = -0.7 + (1 - x_1) + x_2 - (1 - x_1)x_2$$

where $x_1 = P(P)$ and $x_2 = P(Q)$. These are derived as follows:

$$\Pi(S_1) = P(P)$$
$$\Rightarrow 0.7 = P(P)$$
$$\Rightarrow 0 = x_1 - 0.7$$
$$\Rightarrow F(1) = x_1 - 0.7$$

$$\Pi(S_2) = P(\neg P \vee Q)$$
$$\Rightarrow 0.7 = (1 - P(P)) + P(Q) - (1 - P(P))P(Q)$$
$$\Rightarrow 0 = (1 - x_1) + x_2 - (1 - x_1)x_2 - 0.7$$
$$\Rightarrow F(2) = (1 - x_1) + x_2 - (1 - x_1)x_2 - 0.7$$

A solution found by the nonlinear solver (*fsolve* in Matlab) is $x_1 = 0.7$ and $x_2 = 0.571$. The equations for F are generated automatically from the sentences and then solved (For **F**=0. Figure 4 shows the norm of **F** (i.e., distance function between the known sentence probabilities and the atom determined sentence probabilities) over all possible atom probability combinations. The gradient vectors are shown in the x-y plane, and as can be seen, they all lead to the optimal assignment of atom probabilities. (Note that although the surface is shown as a max, *fsolve* finds the minimum of 1 minus this distance.)

Consider a second example with 16 sentences and 9 atoms:

```
1. A
2. ~A v C v D
```

```
 3. ~B v C v D
 4. ~A v E v F
 5. ~B v E v F
 6. ~C v G v H
 7. ~D v G v H
 8. ~E v G v H
 9. ~F v G v H
10. ~A v G v H
11. ~C v ~G
12. ~C v ~H
13. ~E v ~G
14. ~E v ~H
15. ~D v I
16. ~F v I
17. I          -- query
```

The solution to this has $P(A) = 1$, $P(B) \in \{0, 1\}$, $P(C) = 0$, $P(D) = 1$, $P(E) = 0$, $P(F) = 1$, $(P(G), P(H)) \in \{(0, 1), (1, 0), 1, 1)\}$, $P(I) = 1$. Assigning sentence probabilities all 1 (i.e., $P(S_i) = 1, i = 1 \ldots 16$), the nonlinear solver finds the following 3 solutions from starting at initial estimate all 0's, all 0.5's and all 1's, respectively:

```
1 0 -0 1 0 1 0.9998 0.9998 1
1 0.0644 -0 1 -0 1 0.9998 0.9998 1
1 1 -0 1 -0 1 0.9997 0.9997 1
```

A. The Nonlinear Probabilistic Logic (NLPL) Algorithm

Of course, one problem with the nonlinear solver approach is that it may not find a solution, even when one or more exist. For this reason and the fact that high-dimensional spaces (in the number of atoms or sentences) may also pose problems for such solvers, we have developed a robust non-exponential method. This method (NLPPL) avoids the calculation of the semantic tree and provides a set of probabilities, A, for the atoms where these probabilities produce the known sentence probabilities. A can then be used to compute the probability of the clauses in the KB (call this vector $\bar{S}_A$), as well as the probability of the query since it is a disjunction in the literals of the atoms. The NLPL algorithm works well on knowledge bases whose disjunctions have only a few literals.

Summary: This method avoids the exponential cost of the semantic tree expansion, and finds a unique (deterministic) solution for the probabilities of the atoms in the KB, and minimizes $\|\bar{p} - \bar{S}_A\|$.

IV. Experiments

In order to demonstrate the performance of NLPL, a method is required which allows knowledge of the exact probabilities of any logical (disjunctive) clause formed from the literals in the KB. Adams [1] describes a method which allows for this using what he calls *basic states*. Note that these are called *complete conjunctions* by Thimm [22], and he uses them to determine the consistency of an assignment of probabilities to a set of logical sentences. Let $V = \{V_1, V_2, \ldots, V_k\}$ be a set of Boolean random variables. Let's represent the probability of each variable as a subset of the unit square. Then the power set of V provides a set of distinct regions described by the k-bit binary numbers from 0 to 2^k; if bit b is 0, it means that region b is excluded, while a 1 means it is included (note that the low order bit is considered to be bit 1). Thus, for a 3 variable case, 000 means the region which is not in any of the regions of the 3 variables (i.e., the probability of $\neg V_1 \wedge \neg V_2 \wedge \neg V_3$).

Algorithm: Nonlinear Probabilistic Logic (NLPL);
Data: KB - a CNF knowledge base with $C = \{C_i, i = 1 \ldots n\}$, a set of disjuncts; $\bar{p}$, an n-vector of probabilities for the clauses
Result: $A = \{a_i, i = 1 \ldots k\}$, a set of probabilities for the logical atoms in the KB
create a set of formulas, F, from C;
while $\exists f \in F$ *with only one unknown atom probability* **do**
  solve for the atom probability in f
end
while $error > tolerance$ *and* $iteration < maxIter$ **do**
  pick an atom whose probability is unknown;
  change its value a small amount to reduce the distance between the $\bar{p}$ and the $\bar{S}_A$.
end

Algorithm 1: The Nonlinear Probabilistic Logic (NLPL) Algorithm.

For example, Figure 5 shows a 3-variable set of basic regions. The left side shows the regions (each basic state is a different color), and the right side shows the probabilities of each basic state. This set was produced by randomly picking 3 circle centers and radii, and then computing the basic state regions from their power set (each region corresponds to a unique 3-bit binary number). Given the probabilities of the basic states, the probability of any disjunctive clause can be computed by converting the clause to conjunctive form (i.e., negate it), matching the resulting asserted set relations to

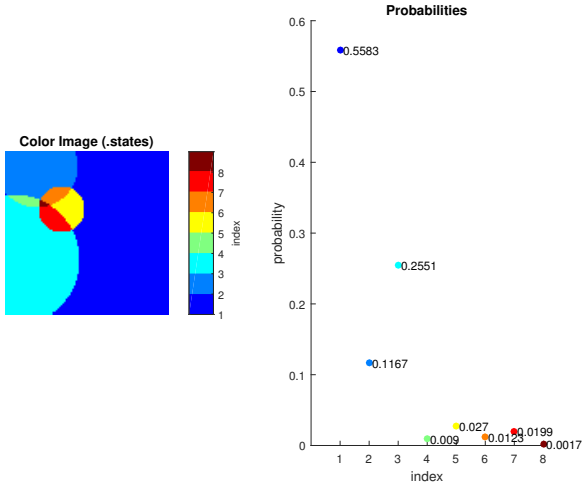

Fig. 5. The Basic States for a 3-Variable Set.

the basic states, and summing the probabilities of matching states. E.g., the probability of V_1 is:

$$P(V_1) = P(001) + P(011) + P(101) + P(111)$$

Given this tool, the experiment is performed as follows. A set of 200 sample knowledge bases is generated, where each randomly generated sample has between 2 to 10 literals, and between 2 to 20 disjuncts. The ground truth probability for the logical atoms is determined using the basic states, and then the NLPL algorithm is given the KB and clause probabilities, and produces its values for the atom probabilities. The performance is measured as μ, the mean error in the individual atom probabilities, and was found to be:

$$\mu = \frac{1}{200} \sum_{i=1}^{200} \mid p(i) - S_A(i) \mid = 0.0424$$

with variance $\sigma^2 = 0.0039$. Thus, we conclude that a mean error of about 4% is to be expected in the atom probabilities found by NLPL.

The above described experiment is only limited in the number of variables due to the fact that the ground truth requires the computation of the power set of the variable regions. An alternative problem is now described which highlights the computational effectiveness of the NLPL algorithm. We define the k-fold **Modus Ponens Problem** as follows: Given a knowledge base, KB with k variables and k clauses, C, of the following form:

$$C_1 \equiv V_1$$

$$C_i \equiv \neg V_{i-1} \vee V_i, i = 2 \ldots k$$

with the appropriate clause probabilities, find:

$$P(V_k)$$

The NLPL algorithm computes this directly from the equations, and is approximately linear in k (i.e., $O(k)$). Neither Nilsson's method nor Markov Logic Networks can solve this for very large k. Figure 6 shows timing results for NLPL on this problem for $k = 2 \ldots 150$.

V. Conclusions and Future Work

The NLPL algorithm produces very usable results probability (likelihood) assessment of statements in logical knowledge bases with reasonably low complexity. However, there are some possible improvements. A major one is to more accurately determine $P(A \wedge B)$; whereas we assume random variable independence, it is possible use $P(A \wedge B) = P(A \mid B)P(B)$ and if information about $P(A \mid B)$ becomes available, use those. For example, if it is known that $A \vdash \neg B$, then $P(A \mid B) = 0$; also, if $B \subseteq A$ (or A is always entailed by B), then $P(A \mid B) = 1$. Note that these are the extreme values of $P(A \mid B)$, and that the value we use, $P(A)$, is intermediate between these extremes, and serves as a weighted estimate.

We also plan to explore the use of logical argumentation theory to reduce the complexity of the query probability calculation; for more on argumentation, see [3], [4], [5], [6], [9], [11], [12], [13], [21], [24]. Given a query, Q, an argument for the query is a minimal, consistent set of clauses, α, from the knowledge base, such that $\alpha \vdash Q$. A smaller set of clauses will most likely reduce the execution cost of NLPL. We also intend to explore how this approach can be used in a non-monotonic knowledge base setting (i.e., where the information updates may produce contradictions – this is where argumentation will be most useful).

We have examined Nilsson's probabilistic logic and discussed its shortcomings; moreover, we have provided new ways to solve this problem and obtain the unique solution (assuming the atoms are independent random variables). We believe that this new approach offers an effective and efficient approach to providing a probabilistic logic for argumentation. We are currently extending this approach to First Order Logic, as well as testing it on a set of geospatial intelligence applications.

References

[1] E.W. Adams. *A Primer of Probability Logic*. CSLI Publicationss, Stanford, CA, 1998.

[2] T. Alsinet, C.I. Chesnevar, L. Godo, and G.R. Simari. A Logic Programming Framework for Possibilistic Argumentation. *Fuzzy Sets and Systems*, 159(10):1208–1228, 2008.

[3] T. Bench-Capon and P.E. Dunne. Argumentation in Artificial Intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007.

[4] T. Bench-Capon, H. Prakken, and G. Sartor. Argumentation in Legal Reasoning. In I. Rahwan and G.R. Simari, editors, *Argumentation in Artificial Intelligence*, pages 363–382, New York, NY, 2009. Springer Verlag.

[5] P. Besnard, A.J. Garcia, A. Hunter, S. Modgil, H.Prakken, and G.R. Simari. Introduction to Structured Argumentation. *Argument and Computation*, 5(1):1–4, 2014.

[6] P. Besnard and A. Hunter. *Elements of Argumentation*. MIT Press, Cambridge, MA, 2008.

[7] M. Biba. *Integrating Logic and Probability: Algorithmic Improvements in Markov Logic Networks*. PhD thesis, University of Bari, Bari, Italy, 2009.

[8] G. Boole. *An Investigation of the Laws of Thought, on which are Founded the mathematical Theorems of Logic and Probability*. Walton and Maberly, London, UK, 1854.

[9] M. Caminada and D. Gabbay. A Logical Account of Formal Argumentation. *Studia Logica*, 93(2):109–145, 2009.

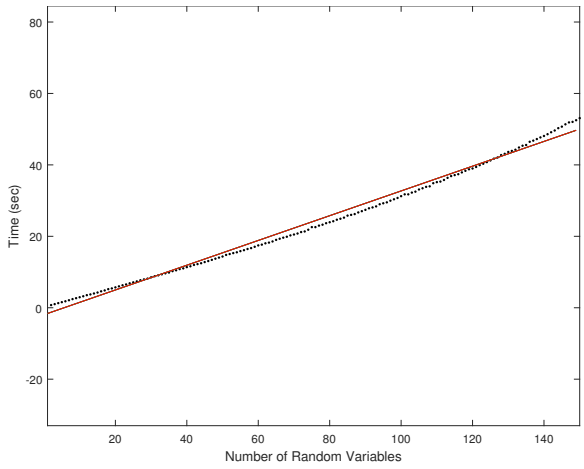

Fig. 6. Timing Performance of NLPL on the k-fold Modus Ponens Problem (red line is linear regression fit).

[10] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool, San Rafael, CA, 2009.

[11] P.M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programmin and n-Person Games. *Artificial Intelligence*, 77(2):321–357, 1995.

[12] P.M. Dung, R.A. Kowalski, and F. Toni. Assumption-Based Argumentation. In I. Rahwan and G.R. Simari, editors, *Argumentation in Artificial Intelligence*, pages 25–44, New York, NY, 2009. Springer Verlag.

[13] X. Fan and F. Toni. On Computing Argumentative Explanation for Abstract Argumentation. In *Proceedings of 21st European Conference on Artificial Intelligence*, Prague, Czech Republic, august 2014.

[14] V. Gogate and P. Domingos. Probabilistic Theorem Proving. *Communications of the ACM*, 59(7):107–115, 2016.

[15] T. Hailperin. *Sentential Probability Logic*. Lehigh University Press, Cranbury, NJ, 1996.

[16] A. Hunter. A Probabilistic Approach to Modelling Uncertan Logical Arguments. *International Journal of Approximate Reasoning*, 54(1):47–81, January 2013.

[17] R. Kowalsi and P.J. Hayes. Semantic Tress in Automatic Theorem Proving. In J.J. Siekmann and G. Wrightson, editors, *Automation of Reasoning*, pages 217–232, Berlin, 1983.

[18] H. Li, N. Oren, and T. Norman. Probabilistic Argumentation Frameworks. In *Proc. 1st International Workshop on the Theory and Applications of Formal Argumentation*, Beijing, China, August 2011.

[19] N. Nilsson. Probabilistic Logic. *Artificial Intelligence Journal*, 28:71–87, 1986.

[20] L. De Raedt, A. Kimmig, and H. Toivonen. HProbLog: A Probabilistic Prolog and its Application in Link Discovery. pages 2462–2467, 2007.

[21] I. Rahwan and G.R. Simari. *Argumentation in Artificial Intelligence*. Springer Verlag, New York, NY, 2009.

[22] M. Thimm. Measuring Inconsistency in Probabilistic Knowledge Bases. In *Proc. 25th Conference on Uncertainty in Artificial Intelligence*, pages 530–537, Montreal, Quebec, June 2009.

[23] M. Thimm. A Probabilistic Semantics for Abstract Argumentation. In *Proc. 20th European Conference on Artificial Intelligence*, Montpellier, France, August 2012.

[24] F. Toni. A Generalized Framework for Dispute Derivations in Assumption-based Argumentation. *Artificial Intelligence*, 195:1–43, 2013.