

Efficient and Effective Approximate Query Answering in Probabilistic DeLP (Thesis Summary)

Mario A. Leiva

Departamento de Cs. e Ing. de la Computacion, Universidad Nacional del Sur (UNS) & Inst. de Cs. e Ing. de la Computacion (UNS-CONICET), San Andres 800, (8000) Bahia Blanca, Argentina

1. Introduction

Within the domain of Artificial Intelligence, continuous developments and research across diverse domains are a daily occurrence, with machine learning taking the forefront in recent years. An ongoing challenge in this field is the need for explainability and interpretability. While knowledge-based systems offer promise in this regard, many systems in various domains grapple with information from multiple heterogeneous sources characterized by different levels of uncertainty stemming from gaps in knowledge, over-specification, or inherent unpredictability.

Given the intricate nature of these domains, an ideal solution involves automated reasoning systems with human-in-the-loop capabilities. Such systems should possess several key features [1], including the ability to: (i) reason about evidence rigorously; (ii) account for evidence with probabilistic uncertainty; (iii) incorporate logical rules to derive conclusions from evidence; (iv) handle conflicting pieces of information, determining their relevance and; (v) provide an understandable and transparent status report, explaining the rationale behind conclusions (i.e., ensuring explainability and interpretability).

The DeLP3E framework [1], tailored to meet these specific needs, is adopted in this work. A DeLP3E knowledge base is comprised of two components: an *analytical model* (AM) and an *environmental model* (EM), representing distinct aspects of the domain (cf. Figure 1). The AM models background knowledge, including rules, facts, or presumptions that inform domain knowledge. Conclusions are drawn from this model using *defeasible logic programming* (DeLP) [2]. In contrast, the EM focuses on probabilistic background knowledge, ensuring consistency by adhering to probabilistic distributions over possible domain states while satisfying constraints and the axioms of probability theory. The EM typically contains data such as evidence, intelligence reporting, or information about actions, software, and systems. DeLP3E effectively accommodates both probabilistic and defeasible uncertainty, allowing for a seamless combination of the two, including annotations of defeasible rules and presumptions with probabilistic events through *annotation functions* (AF).

7th Workshop on Advances in Argumentation in Artificial Intelligence (AI³), November 06–09, 2023, Rome, Italy


✉ mario.leiva@cs.uns.edu.ar (M. A. Leiva)

🌐 <http://marioaleiva.com/> (M. A. Leiva)

🆔 0000-0003-4812-8268 (M. A. Leiva)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

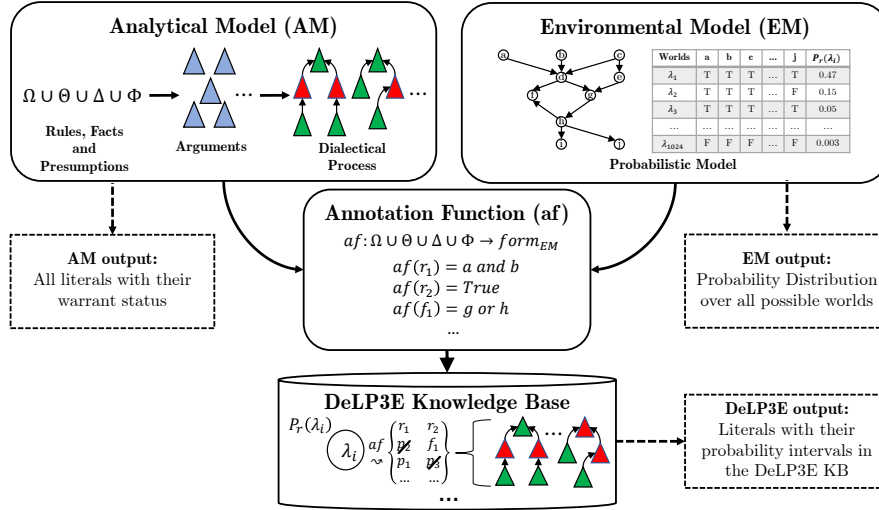


Figure 1: Overview of the DeLP3E framework (reproduced from [3]).

In this model, to answer a query for a literal we need to compute the probability interval with which it is warranted in the DeLP3E KB. For this, we must sum the probability mass of the worlds where the queried literal is warranted (*warranting scenarios*, for the lower bound) and the mass of worlds that warrant the *complement* of the query (for the upper bound). The lower and upper bounds obtained in this manner comprise the probability interval for the query.

This is one of the main sources of computational intractability, since we must answer the query either for all the worlds in the EM model or all the programs in the AM model. Addressing such intractability is the main problem that we address in this paper, which is focused on approximate query answering in probabilistic structured argumentation based on DeLP3E. The ultimate goal is to develop a suite of algorithms for tackling the intractability of this task and selection criteria for choosing the best one based on the analysis of available information.

2. Approximation Methods based on information from the KB

One established approach for addressing the issue of intractability in such situations is to sample a subset of the solution space, aiming to obtain an approximate solution. Two sampling techniques were proposed to approximate the exact value. To determine the most suitable technique for a given scenario, we conducted an analysis of the information within each model, aiming to establish a range of metrics for assessing the configuration's characteristics. Once these metrics were defined, we devised a collection of algorithms capable of leveraging them to approximate the exact interval.

Regarding sampling techniques, we can consider two types of sampling: *world-based* and *subprogram-based*. In world-based sampling, a subset of the possible EM worlds are chosen and, based on the annotation function, different subprograms of P_{AM} are obtained. So, each $P_{AM}(\lambda)$ is a classical DeLP program [2] in which we can query for the status of some literal. On the other

hand, subprogram-based sampling divides the universe of all possible subprograms into regions that represent programs that warrant the query or its complement. Each of these programs can be generated by multiple worlds through their annotations. In both cases, we also have those worlds or programs in which the status of the query can be “undecided” or “unknown”. The objective is to sum the probability value corresponding to the worlds that are warranting scenarios for the query or its complement. Given the incomplete nature of the process, some probability mass will remain unexplored (though a sound approximation is guaranteed).

According to the information obtained using the metrics that determine the structure of each model, we can explore the possible alternatives to derive sampling-based approximation algorithms and run different experiments in different configurations. Our ultimate goal is to develop a set of decision criteria to select the best algorithm for the job, taking into account the inherent tradeoffs between execution time (including the cost of computing any necessary approximate metrics) and the accuracy of the result obtained.

3. KB Metrics, Master Algorithm, and Benchmark Generators

To base technique selection on the best available data, we analyzed the information available in each model in order to define a series of metrics that characterize the current configuration.

KB Metrics. For the definition of metrics of the input KB, two main attributes were established: *size* and *complexity*. The first defines a quantitative value in terms of the number of elements of the model, while the second is related to the difficulty of working and generating structures associated with the model. The defined metrics are shown below, along with an annotation of the attribute they measure – (*s*) for size and (*c*) for complexity, and whether it is possible to compute it tractably (*) or approximate it (~):

- **Analytical Model (AM):**
 - *#Rules* and *#Facts*: Number of rules and facts in the program, resp. (*s*, *)
 - *ALE*: Average argument length (defeasible rules present in any argument) (*c*, ~)
 - *h*: Average height of *dialectic trees* (*c*, ~)
 - τ : Number of *dialectical trees* (*c*, ~)
- **Environmental Model (EM):**
 - *#RandomVars* and *#PGM_Arcs*: Number of random variables and number of arcs in a probabilistic graphical model (PGM), resp. (*s*, *)
 - *PGM_TW*: *Treewidth* of a PGM (such as a BN); approximations for this metric can be computed (*c*, ~)
 - *ent*: *Entropy* of the probability distribution represented by the model (*c*, ~)
- **Annotation Function (AF):** *%AF_ann* and *AF_Comp* – Percentage of formulas of the AM that are annotated and the complexity of each annotation itself, resp. (*s*, *)

Master Query Answering Algorithm. We defined a series of sampling algorithms according to the values of the presented metrics: For example, consider a scenario in which we have 10 elements in the AM (*#Rules*+*#Facts*) and 50 in the EM (*#RandomVars*), and *AF_Comp* is low (only

one variable from EM is used in the annotations, without operators). Here, we can approximate the probability of a query by sampling *subprograms*, since the number of subprograms will be smaller in comparison with the number of worlds ($2^{10} < 2^{50}$), and the annotations associated with subprograms are simple to evaluate (conjunctions of variables or their negation). On the other hand, if the annotations are complex (they use several variables, connectors and negations, for example $af(\omega_1) = a \vee b \rightarrow \neg c$) then sampling by worlds is simpler (even if there are more elements in the EM than rules in the AM). This is because the task of obtaining the subprogram is simple (substitute the random variable values according to the sampled world and evaluating the annotation formula, then query for the literal in the generated program) compared to computing the probability of a complex expression in the EM.

Note that the examples mentioned above expose a kind of asymmetry between the AM and the EM – though given a world it is only possible to generate a single program, a program can be generated by a *set of worlds* (since formulas in annotations can have many models). Consider a case in which the EM is a Bayesian Network with high values of *treewidth* and *entropy*. This leads to slower, more complex, and less guided sampling ; thus, an algorithm to sample worlds randomly is not recommended, and in this case it is better to decide in a previous step which worlds to sample (such as weighted sampling given the BN), in order to optimize resources.

Based on the characterization of the size and complexity of DeLP3E models, we can define a set of criteria to determine different levels of complexity, which can be represented as branches of a *decision tree*, where the leaf nodes will determine the sampling algorithm to choose. The purpose of this tree will be to serve as a guide for the selection of the approximation algorithm based on the value of the metrics and the cost of computing and/or approximating them. This *master query answering algorithm* is presented in detail in the thesis [4].

Automatic Generation of Benchmarks. After developing approximation methods, the need arises to evaluate them under different conditions of size and complexity to analyze their behavior and performance. Towards this end, we developed generators designed to yield synthetic instances of each of the DeLP3E components, and thus be able to create our own datasets to act as benchmarks in our experiments. In particular, the objective was to generate DeLP programs, Bayesian Networks (to be used as a probabilistic model in the environment model) and Annotation Functions that relate the elements of the other two models. Each component generator was designed with parameters that allowed flexibility and range in terms of their capacity to yield instances of varied size and complexity, in order to map instances to each branch of the decision tree.

Here, we only briefly describe the main aspects that guide the creation of models in each generator. In the case of analytical model generation, we build DeLP programs in three stages: (Stage 1) Begins by generating the basic components on which the most complex structures will be created, here the *facts* and *presumptions* are created; (Stage 2) The arguments are created following an organization by levels, where each level groups arguments of a certain length (number of rules that make up the argument) and is constructed using arguments from the lower levels; and (Stage 3) The defeaters and dialectical trees are generated; for this, the structure of an argument is taken and its defeater is constructed using a greater number of rules and supporting the complement of the conclusion of the initial argument. This entire process is guided by a set of parameters that allow the generation of structures that will later impact the

value of the metrics of the model.

As for the Bayesian network generator, it begins by creating a directed acyclic graph with as many variables and arcs as required. The conditional probability tables of each variable are then modified to reduce or increase the entropy of the network. Finally, the annotation function generator consists of taking a subset of AM elements and, for each of them, a logical formula is created and associated using the EM variables and logical connectors. The detail of all the parameters that guide these generators, as well as their design and implementation, are presented in Chapter 4 of the thesis [4].

Based on all these developments, several benchmark scenarios were generated and each approximation method was tested. The results of these tests informed the process of developing the master approximate query answering algorithm described above.

4. Conclusions and Future Work

This work primarily focuses on efficiently computing query answers in probabilistic structured argumentation formalisms, specifically focusing on the DeLP3E model. This formalism is instrumental in handling inconsistent, incomplete, and probabilistic information. To address this challenge, our work introduces mechanisms that approximate answers using algorithms that make the most of available information from each component of the model.

Future research will involve a broader empirical evaluation exploring various sampling methods. The aim is to optimize sampling processes to avoid redundancy and wasted effort. Additionally, we will investigate the use of different probabilistic models, ranging from complex ones like Markov Logic Networks to simpler ones like sets of pairwise independent random variables, each offering unique opportunities for approximations. Finally, we will explore the application of machine learning techniques to guide sampling during the approximation process.

Acknowledgments. This Ph.D. thesis was carried out at Universidad Nacional del Sur (UNS) under the supervision of Gerardo I. Simari and Marcelo A. Falappa.

References

- [1] P. Shakarian, G. I. Simari, G. Moores, D. Paulo, S. Parsons, M. A. Falappa, A. Aleali, Belief revision in structured probabilistic argumentation, *Annals of Mathematics and Artificial Intelligence* 78 (2015) 259–301. URL: <https://doi.org/10.1007/s10472-015-9483-5>. doi:10.1007/s10472-015-9483-5.
- [2] A. J. García, G. R. Simari, Defeasible logic programming: an argumentative approach, *Theory and Practice of Logic Programming* 4 (2004) 95–138. URL: <https://doi.org/10.1017/S1471068403001674>. doi:10.1017/S1471068403001674.
- [3] M. A. Leiva, A. J. García, P. Shakarian, G. I. Simari, Argumentation-based query answering under uncertainty with application to cybersecurity, *Big Data Cogn. Comput.* 6 (2022) 91. URL: <https://doi.org/10.3390/bdcc6030091>. doi:10.3390/bdcc6030091.
- [4] M. A. Leiva, *Herramientas Prácticas para Argumentación Estructurada Probabilística con Aplicación a Ciberseguridad*, Ph.D. thesis, DCIC, Universidad Nacional del Sur (UNS), 2022. URL: <https://repositoriodigital.uns.edu.ar/handle/123456789/6314>.