Towards Effective and Efficient Approximate Query Answering in Probabilistic DeLP*

Mario A. Leiva, Alejandro J. García, and Gerardo I. Simari

Depto. de Cs. e Ing. de la Comp., Universidad Nacional del Sur (UNS)
Inst. de Cs. e Ing. de la Comp. (ICIC UNS-CONICET), Argentina
{mario.leiva,ajg,gis}@cs.uns.edu.ar

1 Introduction

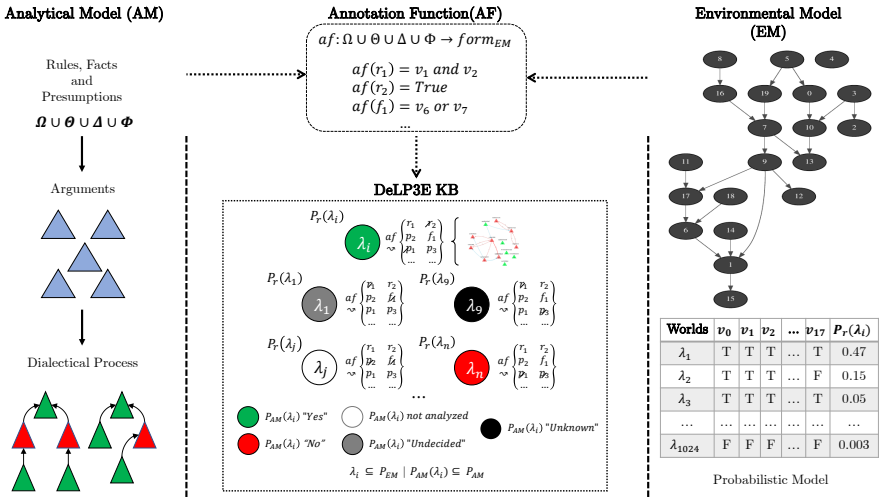
This work presents an overview of a research project focused on R&D for approximate query answering in probabilistic structured argumentation based on DeLP [2]. The ultimate goal is to develop a suite of algorithms for tackling the computational cost of this task and selection criteria for choosing the best one based on the analysis of available information. In the rest of this section, we briefly present the basics of DeLP3E knowledge bases.

The DeLP3E framework

A DeLP3E model [9] consists of two parts: an *environmental model* (EM) and an *analytical model* (AM), which represent different aspects of a domain, plus an additional component—called an *annotation function*—linking the two. Figure 1 provides an overview of these components.

The analytical model contains all the background information that is available for the analysis of a domain: it contains rules, facts, or presumptions to represent available knowledge. Since such knowledge is in general complex and prone to inconsistency and incompleteness, *Defeasible Logic Programming* (DeLP) [2] is a good choice for this component given that it is a formal model that can cope with such features. The environmental model is used to describe the background knowledge and is probabilistic in nature; the choice of such a model will depend on the specific application, but it *must be consistent*. Examples of probabilistic models that can be used are Bayesian Networks (BNs) [7], Markov Logic Networks (MLNs) [8], extensions of first order logic such as Nilsson's probabilistic logic [6], or even ad-hoc specifications of a joint probability distribution.

The main DeLP3E KB components are denoted with P_{EM}, P_{AM}, and af (*annotation function*). Analysts (or automated systems) assign a probability to statements in the EM, whereas statements in the AM can be true or false depending on a certain combination (or several possible combinations) of statements from the EM. There are thus *two kinds of uncertainty* that need to be modeled: probabilistic uncertainty and uncertainty arising from defeasible knowledge. The



Worlds	v_0	v_1	v_2	...	v_{17}	$P_r(\lambda_i)$
λ_1	T	T	T	...	T	0.47
λ_2	T	T	T	...	F	0.15
λ_3	T	T	T	...	T	0.05
...	...	...	...	...	...	...
λ_{1024}	F	F	F	...	F	0.003

Fig. 1. Overview of the DeLP3E Framework

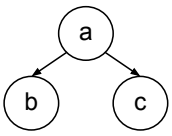
subsets of elements from P_{EM} are called *worlds*; atoms that belong to the set are said to be *true* in that world, while those that do not are *false*. This set is denoted with $\mathcal{W}_{EM}$. In the same way, the set of all subsets of elements of P_{AM} (subprograms) is called $\mathcal{W}_{AM}$. Intuitively, given P_{AM}, every element of $\Omega \cup \Theta \cup \Delta \cup \Phi$ (rules, facts or presumptions) only holds in certain worlds in the set $\mathcal{W}_{\mathcal{EM}}$, i.e., these elements are subject to probabilistic events. Each element of $\Omega \cup \Theta \cup \Delta \cup \Phi$ is thus associated with a formula over the set of all ground elements of P_{EM} (using conjunction, disjunction, and negation, as usual). The notion of *annotation function* implements this association.

In order to answer a query for a literal of interest, 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 that generate subprograms where the queried literal is warranted (*warranting scenarios*, for the lower bound) and the mass worlds whose generated subprogram warrants 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 $\mathcal{W}_{EM}$ or all the programs in $\mathcal{W}_{AM}$. In Section 2, we will present two approaches that we analyze to tackle this intractability.

Next, we present a simple example to illustrate the DeLP3E KB components and show how the probability interval is calculated for a given query.

A Simple Example. Consider a small KB, with components P_{EM}, P_{AM}, and *af* as shown in Figure 2.

For the environmental model P_{EM}, we use a very simple Bayesian network that describes relationship between the nodes *a*, *b* and *c* (EM elements); we omit



Worlds	a	b	c	$Pr(\lambda_i)$
λ_1	T	T	T	0.05
λ_2	T	T	F	0.15
λ_3	T	F	T	0.01
λ_4	T	F	F	0.09
λ_5	F	T	T	0.30
λ_6	F	T	F	0.25
λ_7	F	F	T	0.05
λ_8	F	F	F	0.10

$\Omega:\ \emptyset$
$\Theta:\ \theta_1 = l_1$
$\Delta:\ \delta_1 = l_5 -\!\prec l_3$
$\quad\ \delta_2 = \sim l_2 -\!\prec l_4$
$\quad\ \delta_3 = l_2 -\!\prec l_3$
$\Phi:\ \phi_1 = l_3 -\!\prec$
$\quad\ \phi_2 = l_4 -\!\prec$

$af(\theta_1) = true$
$af(\delta_1) = b \vee c$
$af(\delta_2) = true$
$af(\delta_3) = true$
$af(\phi_1) = a$
$af(\phi_2) = true$

Fig. 2. *Top:* (EM) Bayesian Network and Probability Distribution over $\mathcal{W}_{EM}$. *Bottom left*: (AM) DeLP program that comprises the AM. *Bottom right*: (af) Annotation function.

the detailed specification of the CPTs and instead provide directly the complete probability distribution Pr over all possible worlds $\mathcal{W}_{EM}$. Thus, for example, the probability associated with the world λ_5 where a is false and b and c are true, is 0.30.

The domain knowledge in the analytical model P_{AM} is represented by the DeLP program $P = \Omega \cup \Theta \cup \Delta \cup \Phi$ (see Figure 2 bottom left). In that program, there is an empty set of strict rules $\Omega = \emptyset$, only one fact $\Theta = \{\theta_1\}$, two presumptions $\Phi = \{\phi_1, \phi_2\}$ and three defeasible rules $\Delta = \{\delta_1, \delta_2, \delta_3\}$. The annotation function af (Figure 2 bottom right) associates elements of P with probabilistic events through formulas. As shown, in this example, almost every element hold in every world (annotated as *true*), whereas for rule ϕ_1 the annotation means that this presumption only holds in worlds which the formula a (probabilistic event a) is satisfied, and the annotation for rule δ_1 means that this rule only holds in worlds in which the formula $b \vee c$, that relates the probabilistic events b and c, is satisfied.

To answer a query, consider for example the literal $\sim l_2$. To compute the associated probability interval, we need to obtain the *warranting scenarios* for both $\sim l_2$ (for the lower bound) and its complement l_2 (for the upper bound). It is easy to check that the condition for the former is that a does not hold, which yields the set of worlds $\{\lambda_5, \lambda_6, \lambda_7, \lambda_8\}$; it is also easy to check that there are no warranting scenarios for L_2. Therefore, the lower bound is calculated by summing $\sum_{i=5}^{8} \Pr(\lambda_i) = 0.7$, while the upper bound is simply 1, so the answer to the query is $Pr(\sim l_2) \in [0.70, 1]$.

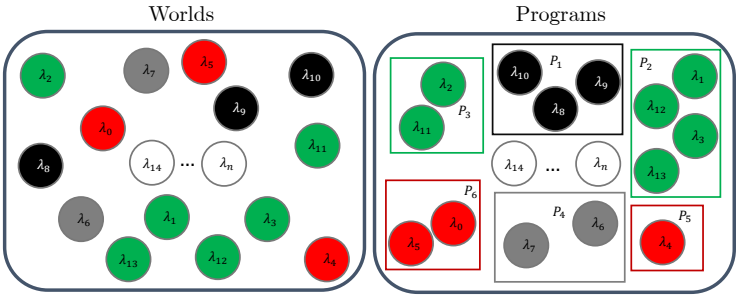


Fig. 3. Worlds and Programs sampling

2 Approximate Query Answering

A well-known technique to address intractability in this kind of situation is to sample a subset of the solution space in order to arrive at an approximate answer. In this case, we can consider two types of sampling: *world-based* and *subprogram-based*, as illustrated in Figure 3. 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 (as shown in Figure 1). 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 green and red worlds (circles in the figure); that is, those that are warranting scenarios for the query and its complement, respectively. Given the incomplete nature of the process, some probability mass will remain unexplored (the white elements in Figure 3).

A Family of KB Metrics. In order to define a suite of sampling algorithms to approximate the answer to a probabilistic query in the most efficient and effective way possible, we must consider what metrics we have available for the input KB. Figure 4 shows a set of such possible metrics, organized with respect to the component they apply to, and what attribute they are designed to measure. For each one, we also analyze whether or not they can be tractably computed (*observable* column) or approximated.

For each component (or submodel of the KB), we can focus on two main attributes: *size* and *complexity*. For P_{AM}, *#Rules* and *#Facts* refers to the total number of different rules and facts in the DeLP program, respectively. Regarding the complexity component, the *MDDL* metric refers to the *maximum defeasible derivation length*, *i.e.*, the maximum number of defeasible rules present in any argument constructed from the program. In a similar vein, h refers to

Component	Attribute	Example Metric	Observable?	Approx.?
P_{AM}	Size	$\#Rules$	Yes	–
		$\#Facts$	Yes	–
	Complexity	$MDDL$	–	Yes
		h	–	Yes
		τ	–	Yes
P_{EM}	Size	$\#RandomVars$	Yes	–
		$\#PGM\_Arcs$	Yes	–
	Complexity	PGM_TW	–	Yes
		$Entropy$	–	Sometimes
AF	Size	$\%AF\_ann$	Yes	–
	Complexity	AF_Comp	Yes	–

Fig. 4. Example metrics computable (exactly or approximately) for a DeLP3E KB.

the maximum length of an argumentation line (a sequence of arguments where each element of the sequence defeats its predecessor), and τ is the number of dialectical trees (a tree structure formed by all argumentation lines arising from one argument) for each literal in the program [2].

On the other hand, for P_{EM}, $\#RandomVars$ refers to the number of random variables in the model (for instance, nodes in a BN), while $\#PGM\_Arcs$ refers to the number of arcs in a probabilistic graphical model (PGM) [4]. The metric PGM_TW refers to the *treewidth* of a PGM (such as a BN), which intuitively measures how "close to a tree" a graph is; approximations for this metric can be computed [3], but the exact value cannot be tractably calculated for very large models. The *entropy* metric is inherent to the joint probability distribution function represented by P_{EM}, and it is possible to approximate it in some particular scenarios [1].

Finally, for annotation functions (AF) an example metric for the size attribute is $\%AF\_ann$, which refers to the percentage of formulas of the program that are annotated. For the complexity attribute, metric AF_Comp refers to the complexity of each annotation itself (that is, which operators can be used, if nesting is allowed, etc.).

Towards a Suite of Approximation Algorithms. Based on the information obtained using these metrics, we can begin to explore the possible alternatives for deriving sampling-based approximation algorithms. For example, consider a scenario in which we have 10 elements in the P_{AM} ($\#Rules + \#Facts$) and 50 in the P_{EM} ($\#RandomVars$), and AF_Comp is low (only one variable from P_{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 P_{EM} than rules in P_{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 P_{AM} and the P_{EM}—even though given a world it is only possible to generate a single program, a program can be generated by a *set of worlds* (since the formulas in the annotations can have more than one model). This can be observed also in Figure 3. Consider a case in which the P_{EM} is a BN with high values of *treewidth* and *entropy*—for example, 5 (complex structure) and 14, respectively—meaning that the network is far from a tree structure and that the underlying probability distribution has a high degree of uncertainty. This leads to a slower, more complex, and less guided sampling process; therefore, 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.

These examples show the variety of possible approaches to approximate query answering in probabilistic DeLP. Our ultimate goal is to develop a set of decision criteria that allow to select the best algorithm for the job, contemplating the inherent tradeoffs between running time (including the cost of calculating any necessary approximate metrics) and precision of the result obtained.

3 Conclusions and Future Work

In this work we presented a preliminary study of the aspects that must be considered when developing a family of approximation algorithms for query answering in the DeLP3E framework. Ongoing and future work consists of developing approximation algorithms that can handle large instances based on the criteria initially explored here, and implement them in the DAQAP platform [5]. In order to do this effectively, we are developing procedures for automatically generating DeLP3E KBs in order to simulate different scenarios, running a suite of approximation algorithms over them, and evaluating their performance. Other approaches currently being evaluated are the application of machine learning techniques to sample in a guided way during the approximation process.

Acknowledgments. This work was supported by funds provided by Universidad Nacional del Sur (UNS) under grants PGI 24/N046 and PGI 24/ZN34), Agencia Nacional de Promoción Científica y Tecnológica under grant PICT-2018-0475 (PRH-2014-0007), and CONICET under grant PIP 11220170100871CO.

References

1. Batu, T., Dasgupta, S., Kumar, R., Rubinfeld, R.: The complexity of approximating the entropy. SIAM J. Comput. **35**(1), 132–150 (2005)

2. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. TPLP **4**(1-2), 95–138 (2004)
3. Kloks, T.: Treewidth: computations and approximations, vol. 842. Springer Science & Business Media (1994)
4. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
5. Leiva, M.A., Simari, G.I., Gottifredi, S., García, A.J., Simari, G.R.: DAQAP: defeasible argumentation query answering platform. In: Proc. FQAS 2019. LNCS, vol. 11529, pp. 126–138. Springer (2019)
6. Nilsson, N.J.: Probabilistic logic. Artificial intelligence **28**(1), 71–87 (1986)
7. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Elsevier (2014)
8. Richardson, M., Domingos, P.: Markov logic networks. Machine learning **62**(1-2), 107–136 (2006)
9. Shakarian, P., Simari, G.I., Moores, G., Paulo, D., Parsons, S., Falappa, M.A., Aleali, A.: Belief revision in structured probabilistic argumentation. Annals of Mathematics and Artificial Intelligence **78**(3-4), 259–301 (2016)