big data and cognitive computing

Article

Argumentation-Based Query Answering under Uncertainty with Application to Cybersecurity

Mario A. Leiva [1,2], Alejandro J. García [1,2], Paulo Shakarian [3] and Gerardo I. Simari [1,2,3,*]

[1] Department of Computer Science and Engineering, Universidad Nacional del Sur (UNS), Bahia Blanca 8000, Argentina
[2] Institute for Computer Science and Engineering (UNS–CONICET), Bahia Blanca 8000, Argentina
[3] School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85281, USA
[*] Correspondence: gis@cs.uns.edu.ar

Abstract: Decision support tools are key components of intelligent sociotechnical systems, and their successful implementation faces a variety of challenges, including the multiplicity of information sources, heterogeneous format, and constant changes. Handling such challenges requires the ability to analyze and process inconsistent and incomplete information with varying degrees of associated uncertainty. Moreover, some domains require the system's outputs to be explainable and interpretable; an example of this is cyberthreat analysis (CTA) in cybersecurity domains. In this paper, we first present the P-DAQAP system, an extension of a recently developed query-answering platform based on defeasible logic programming (DeLP) that incorporates a probabilistic model and focuses on delivering these capabilities. After discussing the details of its design and implementation, and describing how it can be applied in a CTA use case, we report on the results of an empirical evaluation designed to explore the effectiveness and efficiency of a possible world sampling-based approximate query answering approach that addresses the intractability of exact computations.

Keywords: intelligent sociotechnical systems; human-in-the-loop computing; structured probabilistic argumentation; cybersecurity

Citation: Leiva, M.A.; García, A.J.; Shakarian, P.; Simari, G.I. Argumentation-Based Query Answering under Uncertainty with Application to Cybersecurity. *Big Data Cogn. Comput.* **2022**, *6*, 91. https://doi.org/10.3390/bdcc6030091

Academic Editors: Peter R.J. Trim and Yang-Im Lee

Received: 26 July 2022
Accepted: 22 August 2022
Published: 26 August 2022

1. Introduction

Sociotechnical systems [1] are an important class of applications of artificial intelligence (AI) tools, since many deployments of technology built on their foundations are at the core of decision processes at the individual and the organizational levels. An inherent problem in this area is that of explainability and interpretability, topics that were not central in earlier "AI booms" characterized by expert systems and rule-based models. The issues underlying this problem are within the domain of explainable AI (XAI) [2], which is now widely recognized as a crucial feature for the practical deployment of AI models [3]. The importance of this aspect can be appreciated by pointing to the Explainable Artificial Intelligence (XAI) program launched by the Defence Advanced Research Projects Agency (DARPA) [4], which aims to create a set of new artificial intelligence techniques that allow for end users to understand, properly trust, and effectively manage the emerging generation of artificial intelligence systems [5]. The danger is that complex black-box models (some of which can comprise hundreds of layers and millions of parameters) [6] are increasingly used for important predictions in critical contexts, and these models generate outputs that may not be justified or simply do not allow for detailed explanations of their behavior [4]. In this direction, recent work focused on addressing these problems from different points of view [7–9]. In this paper, we focus on cybersecurity as a salient example of a sociotechnical domain [10] in which the availability of explanations that support the output of a model are crucial. Transparency, together with a human-in-the-loop (HITL) scheme, leads to more robust decision-making processes whose results can be trusted by users [8]. Achieving this is challenging, since many domains involve information arriving from multiple heterogeneous sources with different levels of uncertainty due to gaps in knowledge (incompleteness), overspecification (inconsistency), or inherent uncertainty.

In cybersecurity domains, a clear example is the task of real-time security analysis, a complex process in which many uncertain factors are involved, given that analysts must deal with the behavior of different actors and entities, the dynamic nature of exploits, and the fact that the observations of potentially malicious activities are limited. Cyberthreat analysis (CTA) [11] is a highly technical intelligence problem in which (human) analysts take into consideration multiple sources of information, with possibly varying degrees of confidence or uncertainty, with the goal of gaining insight into events of interest that may represent a threat to a system. When building AI tools to assist such a process, knowledge engineers face the challenge of leveraging uncertain knowledge in the best possible way [12]. Due to the nature of these analytical processes, an automated reasoning system with human-in-the-loop capabilities would be best suited for the task. Such a system must be able to accomplish several goals, among which we distinguish the following main capabilities [13]: (i) reason about evidence in a formal, principled manner; (ii) consider evidence associated with probabilistic uncertainty; (iii) consider logical rules that allow for the system to draw conclusions on the basis of certain pieces of evidence and iteratively apply such rules; (iv) consider pieces of information that may not be compatible with each other, deciding which the most relevant are; and (v) show the actual status of the system on the basis of the above-described features, and provide the analyst with the ability to understand why an answer is correct, and how the system arrives at that conclusion (i.e., *explainability and interpretability*). In this context, there is a specific literature to the study of techniques and methodologies for providing explanations in cybersecurity domains [14–17]. The model that we develop in this work is based on *argumentation-based reasoning*, an approach that is designed to mimic the way humans with which rationally arrive at conclusions by analyzing arguments for and against them, and is especially well-suited for accommodating desirable features, such as reasoning about possibly uncertain evidence in a principled manner, handling pieces of information that may not be compatible with each other, and showing the actual status of the system to analysts along with the ability to understand why an output is produced.

Contributions. We contribute to the area of intelligent systems applied to cybersecurity in the following ways:

- A use case for the application of a structured probabilistic argumentation model (DeLP3E) [18] based on publicly available cybersecurity datasets.
- Design of the P-DAQAP framework, an extension of DAQAP [19], to work with DeLP3E, and the proposal of different classes of queries in the context of applications related to CTA.
- A preliminary empirical evaluation of an approximation algorithm for probabilistic query answering in P-DAQAP, showing the potential for the system to scale to nontrivial problem sizes, arriving at solutions efficiently and effectively.

To the best of our knowledge, this is the first system of its kind. In particular, being able to consider the internal structure of arguments allows for the platform to be extended to work with other defeasible argumentation formalisms, and offers greater transparency to adapt classical approaches that do not consider probabilistic information.

2. Preliminaries

Tools developed in the area of argumentation-based reasoning offer the possibility of analyzing complex and dynamic domains by studying the arguments for and against a conclusion. Specifically, *defeasible argumentation* leverages models that contain inconsistency, evaluating arguments that support contradictory conclusions and deciding which ones to keep [20]. An argument supports a conclusion from a set of premises [20]; a conclusion `C` constitutes a piece of tentative information that an agent is willing to accept. If the agent then acquires new information, conclusion `C`, along with the arguments that support it, could be invalidated. The validity of a conclusion `C` is guaranteed when there is an argument that provides justification for `C` that is undefeated. This process involves the construction of an argument $\mathcal{A}$ for C, and the analysis of counterarguments that are possible defeaters of $\mathcal{A}$; as these defeaters are arguments, it must be verified that they are not themselves defeated. There are several formalisms that are based on this idea, such as ABA [21], ASPIC+ [22], *defeasible logic programming* (DeLP) [23], and *deductive argumentation* [24], which consider the structure of the arguments that model a discussion. The DAQAP platform [19] on which the presented system is based uses DeLP as its central formalism. We now briefly present the necessary background, starting with DeLP and its probabilistic extension.

2.1. Defeasible Logic Programming (DeLP)

DeLP combines logic programming and defeasible argumentation. A DeLP program $\mathcal{P}$, also denoted as (Π, Δ), is a set of facts and strict rules (Π), and defeasible rules (Δ). *Facts* are ground literals representing atomic information (or its negation using strong negation "$\sim$"), *strict rules* represent nondefeasible information, and *defeasible rules* represent tentative information. Here, we consider the extension that incorporates *presumptions* to set Δ, which can be thought of as a kind of defeasible fact [25].

The dialectical process used in deciding which information prevails as *warranted* involves the construction and evaluation of arguments that either support or interfere with the query under analysis. An *argument* $\mathcal{A}$ is a minimal set of defeasible rules that, along with the set of strict rules and facts, are not contradictory and derive a certain conclusion α, denoted as $\langle \mathcal{A}, \alpha \rangle$. Arguments supporting the answer for a query can be organized using *dialectical trees*. A query is issued to a program (Π, Δ) in the form of a ground literal α.

A literal α is *warranted* if there exists a nondefeated argument $\mathcal{A}$ supporting α. To establish if $\langle \mathcal{A}, \alpha \rangle$ is a nondefeated argument, *defeaters* for $\langle \mathcal{A}, \alpha \rangle$ are considered, *i.e.*, *counterarguments* that by some criteria are preferred to $\langle \mathcal{A}, \alpha \rangle$. An argument $\mathcal{A}_1$ is a counterargument for $\mathcal{A}_2$ *iff* $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \Pi$ is contradictory. Given a preference criterion, and an argument $\mathcal{A}_1$ that is a *defeater* for $\mathcal{A}_2$, $\mathcal{A}_1$ is called a *proper defeater* if it is preferred to $\mathcal{A}_2$, or a *blocking defeater* if it is equally preferred or is incomparable with $\mathcal{A}_2$. Since there may be more than one defeater for a particular argument, many acceptable argumentation lines could arise from one argument, leading to a tree structure. This is called a *dialectical tree* because it represents an exhaustive dialectical analysis for the argument in its root; every node (except the root) represents a defeater of its parent, and leaves correspond to nondefeated arguments. Each path from the root to a leaf corresponds to a different acceptable argumentation line. A dialectical tree provides a structure for considering all possible acceptable argumentation lines that can be generated for deciding whether an argument is defeated.

Given a literal α and an argument $\langle \mathcal{A}, \alpha \rangle$ from a program $\mathcal{P}$, to decide whether α is warranted, every node in the tree is recursively marked as D (*defeated*) or U (*undefeated*), obtaining a marked dialectical tree $\mathcal{T}_{\mathcal{P}}(\mathcal{A})$: (1) all leaves in $\mathcal{T}_{\mathcal{P}}(\mathcal{A})$ are marked as "U"s; and (2) let $\mathcal{B}$ be an inner node of $\mathcal{T}_{\mathcal{P}}(\mathcal{A})$; then, $\mathcal{B}$ is marked as U *iff* every child of $\mathcal{B}$ is marked as D. Thus, node $\mathcal{B}$ is marked as D *iff* it has at least one child marked as U. Given an argument $\langle \mathcal{A}, \alpha \rangle$ obtained from $\mathcal{P}$, if the root of $\mathcal{T}_{\mathcal{P}}(\mathcal{A})$: is marked as U, then $\mathcal{T}_{\mathcal{P}}(\mathcal{A})$ *warrants* α, and α is *warranted* from $\mathcal{P}$. The DeLP interpreter takes a program $\mathcal{P}$ and a DeLP query L, and returns one of the following four possible answers: YES if L is warranted from $\mathcal{P}$, NO if the complement of L regarding strong negation is warranted from $\mathcal{P}$, UNDECIDED if neither L nor its complement are warranted from $\mathcal{P}$, or UNKNOWN if L is not in the language of the program $\mathcal{P}$.

2.2. Probabilistic DeLP: DeLP3E Framework

We now provide a brief introduction to DeLP3E; for full details, we refer the reader to [18]. A DeLP3E KB $P = (AM, EM, af)$ consists of three parts that correspond to *two separate models of the world*, and a function linking the two; these components are illustrated in Figure 1.

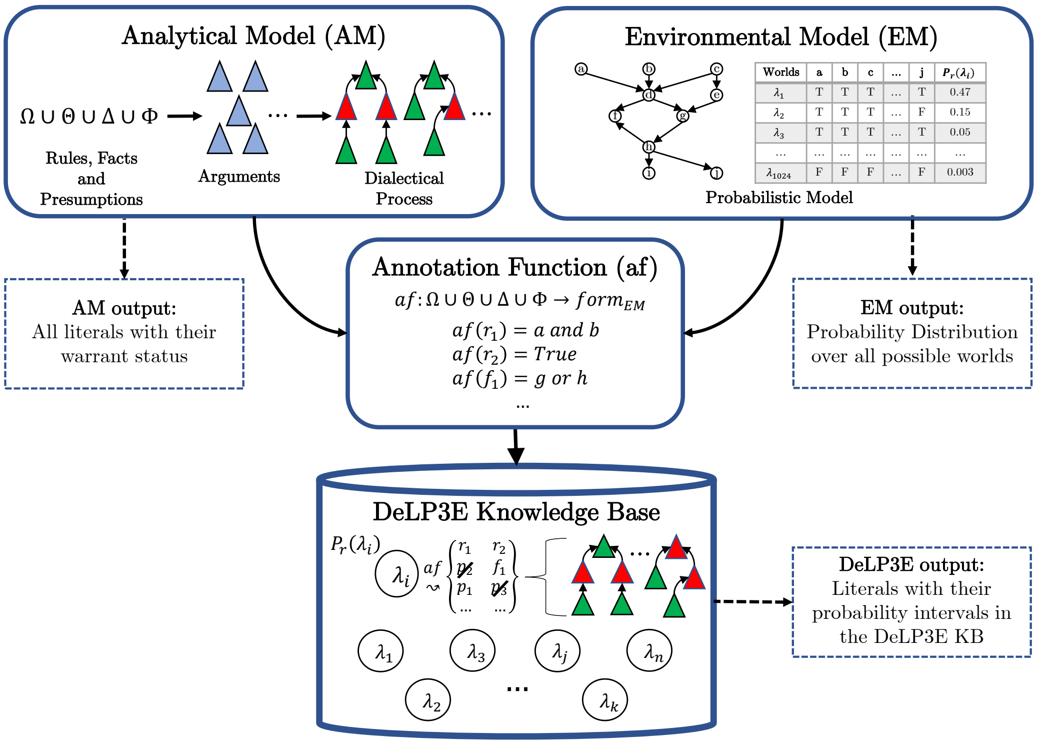

Analytical Model (AM)
Ω ∪ Θ ∪ Δ ∪ Φ
Rules, Facts and Presumptions
Arguments
Dialectical Process
Environmental Model (EM)
Worlds | a | b | c | … | j | $P_r(\lambda_i)$
λ_1 | T | T | T | … | T | 0.47
λ_2 | T | T | T | … | F | 0.15
λ_3 | T | T | T | … | T | 0.05
λ_{1024} | F | F | F | … | F | 0.003
Probabilistic Model
Annotation Function (af)
$af: \Omega \cup \Theta \cup \Delta \cup \Phi \rightarrow form_{EM}$
$af(r_1) = a \text{ and } b$
$af(r_2) = True$
$af(f_1) = g \text{ or } h$
AM output: All literals with their warrant status
EM output: Probability Distribution over all possible worlds
DeLP3E Knowledge Base
$P_r(\lambda_i)$
λ_i
af
λ_1 λ_2 λ_3 λ_j λ_k λ_n
DeLP3E output: Literals with their probability intervals in the DeLP3E KB

Figure 1. Overview of the DeLP3E framework.

The *environmental model* (EM) is used to describe background knowledge that is probabilistic in nature, while the *analytical model* (AM) is used to analyze competing hypotheses that can account for a given phenomenon. The EM *must be consistent*, while the AM allows for contradictory information as the system must have the capability to reason about competing explanations for a given event. In general, the EM contains knowledge such as evidence, intelligence reporting, or uncertain knowledge about actors, software, and systems, while the AM contains elements that the analyst can leverage on the basis of information in the EM. AMs correspond to DeLP programs, while EMs in this paper are abstracted away, assuming that the well-known Bayesian network model is used.

Finally, the third component is the *annotation function*, which links components in the AM with conditions over the EM (the conditions under which statements in the AM can potentially be true). We use G_{EM} to denote the sets of all ground atoms for the EM; here, we concentrate on subsets of ground atoms from G_{EM}, called *worlds*. Atoms that belong to the set are *true* in the world, while those that do not are *false* (Therefore, there are $2^{|G_{EM}|}$ possible worlds in the EM). This set is denoted with $\mathcal{W}_{EM}$. Logical formulas arise from the combination of atoms using the traditional connectives ($\wedge$, $\vee$, and $\neg$); we use $form_{EM}$ to denote the set of all possible (ground) formulas in the EM. Annotation functions then assign formulas in $form_{EM}$ to components in the AM to indicate the conditions (probabilistic events) under which they hold. In this way, each world $\lambda \in \mathcal{W}_{EM}$ *induces* a subset of the AM, comprised of all elements whose annotations are satisfied by λ; for DeLP3E program P, we denote the subset of the AM induced by λ with $P_{AM}(\lambda)$ (cf. Figure 1). Exact probabilistic query answering is carried out via Algorithm 1.

Algorithm 1: Exact probabilistic query answering

```
def compute_answer(query)
    Data: $P = (AM, EM, af)$
    Result: $[\ell, u]$
    begin
        Initialize $\ell = 0$ and $u = 1$          /* the limits of the interval */
        for EM worlds $\lambda_i$ do
            Compute the induced AM subprogram $P_{AM}(\lambda_i)$
            if the query is warranted in that program then
                $\ell \leftarrow \ell + \Pr(\lambda_i)$
            else if the negation of the query is warranted then
                $u \leftarrow u - \Pr(\lambda_i)$
        end
        return $[\ell, u]$
    end
end
```

Since the number of worlds in $\mathcal{W}_{EM}$ is exponential in the number of EM random variables, this procedure quickly becomes intractable. However, a *sound approximation* of the exact interval can be obtained by simply selecting a subset of $\mathcal{W}_{EM}$ and executing the same procedure. We refer to this algorithm as approximate query answering via *world sampling*. It is easy to see that this approximation scheme is sound since it always yields intervals $[\ell', u'] \subseteq [\ell, u]$. Section 5 is dedicated to studying the effectiveness and efficiency of this approach.

A Simple Illustrative Example

In order to clearly illustrate the model and query-answering procedure in DeLP3E, we present the following simple example of knowledge base $P = (AM, EM, af)$:

Analytical Model

$\theta_1 : L_1$
$\theta_2 : L_2$
$\theta_3 : \sim L_1$

Annotation Function

$af(\theta_1) : a \wedge \neg b$
$af(\theta_2) : b$
$af(\theta_3) : b$

Environmental Model

World	a	b	$P_r(\lambda_i)$
λ_1	T	T	0.25
λ_2	T	F	0.20
λ_3	F	T	0.05
λ_4	F	F	0.50

We have an AM consisting of three literals, an EM consisting of two variables, and an annotation function that relates these two models; suppose we query for the literal L_1. To compute the exact probability interval, we go world by world as described above, generating the corresponding subprogram and querying each one of them for the status of the query. Lastly, in order to arrive at the probability interval with which L_1 is warranted in P, we keep track of the probability of the worlds where the query is warranted (for the lower limit of the interval) and the probability of the worlds where the *complement* of the query is warranted (for the upper limit). In our example, the result for query L_1 is $[0.20, 0.70]$; the details of this calculation are as follows:

- **Subprograms induced in each possible world:**
  - $P_{AM}(\lambda_1) = \{L_2, {\sim}L_1\}$
  - $P_{AM}(\lambda_2) = \{L_1\}$
  - $P_{AM}(\lambda_3) = \{L_2, {\sim}L_1\}$
  - $P_{AM}(\lambda_4) = \{\emptyset\}$

  Query L_1 is, thus, clearly warranted only in world λ_2, while its complement (${\sim}L_1$) is warranted in λ_1 and λ_3.
- **Probability interval calculation:**

$$\left[\ell = \sum P_r(\lambda_2), \quad u = 1 - \sum_{i=1,3} P_r(\lambda_i)\right]$$

- **Result:** $0.20 \leq P_r(L_1) \leq 0.70$

The resulting probability interval represents *two kinds of uncertainty*: the first, called *probabilistic* uncertainty, arises from the environmental model since we have a probability distribution over possible worlds; the second, *epistemic* uncertainty, arises from the fact that we we generally have a probability interval instead of a point probability, which happens when there are worlds in which neither the query nor its complement are warranted (as is the case of world λ_4 above).

Having presented the preliminary concepts, in the next section, we illustrate the application of DeLP3E in a cybersecurity domain.

3. Cyberthreat Analysis with DeLP3E

We now present a use case leveraging several datasets developed and maintained by the MITRE Corporation (a not-for-profit organization that works with governments, industry, and academia) and National Institute of Standards and Technology (NIST) (MITRE datasets: ATT&CK (https://attack.mitre.org, accessed on 21 August 2022), CAPEC (https://capec.mitre.org, accessed on 21 August 2022), and CWE (https://cwe.mitre.org, accessed on 21 August 2022). NIST manages the National Vulnerability Database (NVD) (https://nvd.nist.gov, accessed on 21 August 2022) that includes CVE and CPE). Figure 2 shows an overview of our approach. We first describe the basic components and then show how the DeLP3E components are specified, along with two queries for addressing specific problems in the CTA domain.

The ATT&CK model is a curated knowledge base and model geared towards adversarial behavior in cybersecurity settings; it contains information on the various phases of an attack and the platforms that are most commonly targeted. The behavioral model consists of several core components:

(i) *Tactics*, denoting short-term tactical adversary goals during an attack.
(ii) *Techniques*, describing the means by which adversaries achieve tactical goals.
(iii) *Subtechniques*, describing more specific means at a lower level than that of techniques by which adversaries achieve tactical goals.
(iv) Documented *adversary usage* of techniques, their procedures, and other metadata.

The supporting datasets provide information on *attack patterns* (Common Attack Pattern Enumeration and Classification—CAPEC), software and hardware *weakness types* (Common Weakness Enumeration—CWE), and the *National Vulnerability Database* (NVD). The latter is a rich repository of data; here, we distinguish two subsets including data about *vulnerabilities* (Common Vulnerabitlities and Exposures—CVE) and *platforms* (Common Platform Enumeration—CPE).

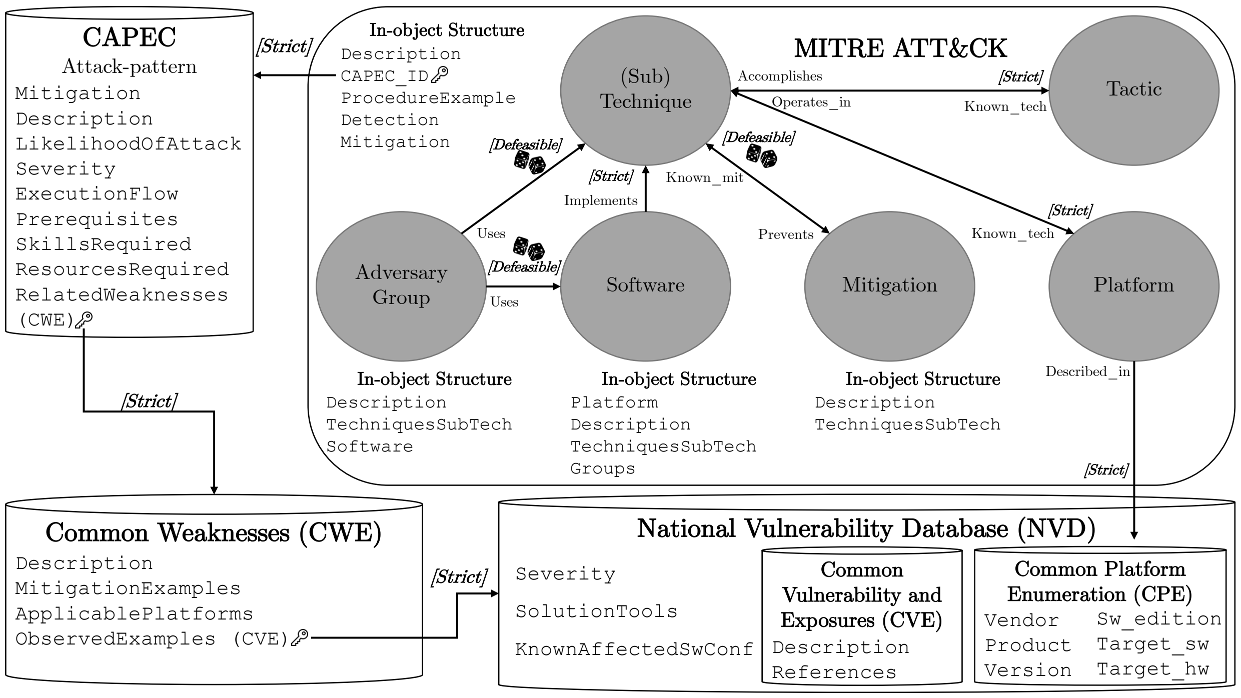


Figure 2. Designing a DeLP3E KB for cyberthreat analysis from a variety of publicly available cyber security datasets.

Figure 2 shows the information provided by each dataset, and how they are related to each other via foreign keys. For instance, attack techniques included in ATT&CK link to entries in CAPEC, which in turn link to CWE and NVD. We augmented this structure with two features towards deriving a DeLP3E KB. First, we labeled connections between datasets (and components within ATT and CK) with either "[*strict*]" or "[*defeasible*]", indicating the type of knowledge being encoded. For instance, observed examples of a weakness included in CWE are linked to CVEs included in the NVD as strict, since this is well-established knowledge. On the other hand, mitigation strategies are linked to techniques as defeasible knowledge, since the relationship between the two is tentative in nature. The second feature, which appears in the figure as a small icon depicting a pair of dice, indicates relationships that are subject to *probabilistic events*. For the purposes of this use case, we label all defeasible relations in this way.

We used all this information to create the AM, EM, and annotation function, and create a DeLP3E KB; an introductory example is shown in Listing 1. On the left-hand side, we have the elements of the AM that can be used to create arguments for and against conclusions; for instance:

$$\langle \mathcal{A}_1,\ \textit{tech\_in\_use(account\_discovery)} \rangle, \text{ with}$$
$$\mathcal{A}_1 = \{\delta_3,\ \theta_1(adv\_group(apt29))\}$$
$$\langle \mathcal{A}_2,\ \sim\!\textit{impl\_techsub(os\_credential\_dumping)} \rangle, \text{ with}$$
$$\mathcal{A}_2 = \{\delta_6,\ \delta_1(\textit{prev\_techsub(os\_credential\_dumping)}),\ \phi_1(\textit{mitigation(credential\_access\_protection)})\}.$$

Listing 1. *Left*: DeLP program that comprises the AM. *Right*: Annotation function.

Θ	$\theta_1 : adv\_group(G)$	
	$\theta_3 : platform\_available(P)$	
	$\theta_2 : software(S)$	
	$\theta_4 : tech\_subtech(T\_ST)$	
Ω	$\omega_1 : accomp\_tactic(Tactic) \leftarrow tech\_subtech(T\_ST)$	
	$\omega_2 : op\_in\_platform(Platform) \leftarrow tech\_subtech(T\_ST)$	
	$\omega_3 : impl\_techsub(T\_ST) \leftarrow software(S)$	
	$\omega_4 : capec\_rel\_weaknesses(CWE\_List) \leftarrow capec\_id(T\_ST)$	
	$\omega_5 : cwe\_observed(CVE\_List) \leftarrow capec\_rel\_weaknesses(CWE\_List)$	
	$\omega_6 : nvd\_cve(Vuln\_info) \leftarrow cwe\_observed(CVE\_List)$	
	$\omega_7 : known\_techst(T\_ST) \leftarrow accomp\_tactic(T)$	
	$\omega_8 : known\_techst(T\_ST) \leftarrow platform\_available(P)$	
Φ	$\phi_1 : mitigation(M) \mathrel{-\!\!\prec}$	
	$\phi_2 : likelihoodAttack(CAPEC\_ID, Value) \mathrel{-\!\!\prec}$	$af(\phi_1) = e_1$
Δ	$\delta_1 : prev\_techsub(T\_ST) \mathrel{-\!\!\prec} mitigation(M)$	$af(\phi_2) = e_2$
	$\delta_2 : known\_mit(M) \mathrel{-\!\!\prec} tech\_subtech(T\_ST)$	$af(\delta_1) = e_3$
	$\delta_3 : tech\_in\_use(T\_ST) \mathrel{-\!\!\prec} adv\_group(G)$	$af(\delta_2) = e_4$
	$\delta_4 : soft\_in\_use(S) \mathrel{-\!\!\prec} adv\_group(G)$	$af(\delta_3) = e_5$
	$\delta_5 : pos\_threat(T\_ST, S) \mathrel{-\!\!\prec} tech\_in\_use(T\_ST), soft\_in\_use(S)$	$af(\delta_4) = e_6$
	$\delta_6 : {\sim}impl\_techsub(T\_ST) \mathrel{-\!\!\prec} prev\_techsub(T\_ST)$	$af(\delta_5) = e_7$
	$\delta_7 : intensify\_mit(M) \mathrel{-\!\!\prec} known\_mit(M), tech\_in\_use(T\_ST),$	$af(\delta_6) = e_8$
	$likelihoodAttack(T\_ST, high)$	$af(\delta_7) = e_9$

The former indicates that *account discovery* is used as an attack technique, since the advanced persistent threat group 29 (APT29, also known as Cozy Bear) is active and uses it. The latter refers to the use of *credential access protection* as a mitigation technique to prevent the use of *OS credential dumping*. This is a clear example of an argument that involves uncertainty, since credential access protection is not a foolproof endeavor. An example of this is the well-known *Heartbleed* vulnerability (CVE-2014-0160) that affected OpenSSL implementations, leaving them open to credential dumping. For reasons of space, in this simple example, we only label AM components with probabilistic events (e_1–e_9; elements with no annotation are simply labeled with *true*) and do not describe how they are related in the EM. One example could be to simply assume pairwise independence (as in many probabilistic database models [26]), or a Bayesian network [27], as described in Section 5.

Queries. We lastly present two queries that we revisit in the next section:

- $pos\_threat(T1134, SO344)$:
  What is the probability that *access token manipulation* (technique T1134) uses leveraging the *Azorult* malware (software id SO344) to attack our systems?
- $intensify\_mit(M1026)$:
  What is the probability that *privileged account management* (mitigation strategy M1026) should be deployed? M1026 mitigates T1134.

In the next two sections, we discuss the design of a software system for implementing this kind of functionalities based on DeLP3E, and a preliminary evaluation of query answering in DeLP3E via sampling techniques.

4. P-DAQAP Platform

In an early version of the platform called DAQAP [19], we developed a web-based client-server platform that offers an interface to visualize the interaction of the arguments generated from an input DeLP program via dialectical trees and graphs, as well as the abstract defeat relationships in a Dung-like graph environment. In this section, we present the extension that incorporates probabilistic reasoning based on DeLP3E knowledge bases, first briefly discussing the platform's architecture and workflow, and then moving on to presenting a set of features that could eventually support human-in-the-loop reasoning and XAI functionalities.

4.1. Architecture and Workflow

Figure 3 shows an overview of the tool's architecture and workflow that is a mock-up of a possible user interface that we are currently developing. The architecture is divided into two main modules, the front end and the back end. Within the former, there are two main sections: the DeLP and abstract argumentation section manages classical (nonprobabilistic) models and is described in detail in [19]; we focus on the DeLP3E section, which is the extension presented here. The back end is organized analogously, with the addition of three other submodules that implement the probability model (for the EM), sampling methods, and approximation algorithms.

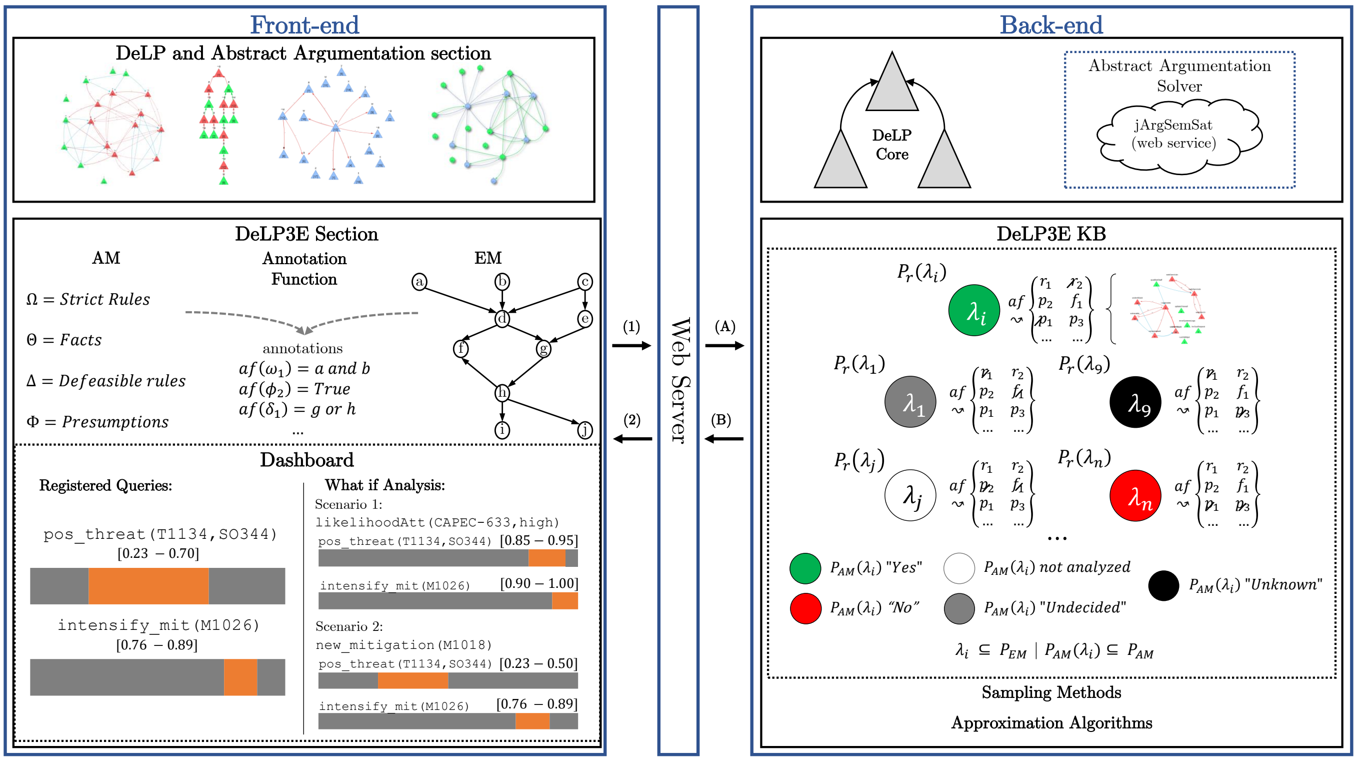


Figure 3. P-DAQAP platform architecture, including a mock-up of a dashboard for displaying query-answering results related to our use case.

Table 1 describes the workflow focused on DeLP3E tasks in the order of the steps labeled at the interaction between the two main modules in Figure 3 (1 $\rightarrow$ A $\rightarrow$ B $\rightarrow$ 2). This workflow is iterative in nature, and implements the human-in-the-loop model mentioned in Section 1. In Step B, an *anytime algorithm* approach may be applied, in which results are iteratively improved, and the user can decide when to stop the job depending on the amount of time available and/or the quality of the result currently being obtained. After Step 2, the analyst can now interact through the dashboard in response to the results received, for example by choosing to modify the DeLP3E KB, modifying the query issued in the first step, or a combination of such actions.

Table 1. P-DAQAP Workflow

	Front-end
Step 1	Loads a DeLP3E knowledge base and specifies a *task*.
	Back-end
Step A	Web server sends the job to be executed by the Probabilistic Argumentation module.
Step B	Generate data structures and executes the job; when results become available, it returns the output data in JSON format to the web server.
	Front-end
Step 2	Client receives the response, and the data are presented to the user.

In the next section, we explore some of these functionalities, illustrating them via the use case presented in Section 3.

4.2. P-DAQAP Functionalities

We begin by describing the design of two functionalities based on our use case, which are illustrated in the *Dashboard* section of Figure 3, and then discuss the next steps to be developed.

4.2.1. Current State: *Registered Queries*

The values of a subset of the EM variables are set depending on the current state of the system (observed evidence). The analyst registers a set of queries of interest in order to monitor the associated probabilities. Consider the queries presented in Section 3; the user is interested in monitoring a possible threat and degree of application of a corresponding mitigation strategy. In Figure 3 (bottom left), we can see that in the current state the query

$$pos\_threat(T1134, SO344)$$

(referring to the probability that access token manipulation is used, levaraging Azorult) is currently warranted by the KB with probability interval $[0.23, 0.7]$; this interval is quite wide, which points to a large amount of uncertainty and lack of actionable insight.

On the other hand, the query

$$intensify\_mit(M1026)$$

(which refers to the probability that privileged account management should be deployed as a mitigation strategy) yields an interval of $[0.76, 0.89]$, which signals a high probability of the need to intensify mitigating actions associated with technique T1134.

Having this kind of insight is valuable for analysts, who can register queries regarding mitigation strategies and attack techniques of current interest. The results can inform, for instance, security alert levels and patching effort priorities for system administrators. As we discuss in Section 5, approximations can be computed whenever the cost of obtaining an exact answer is too high. In this case, the system can allow for the user to input the number of samples to be used or, given an explicit upper bound on the time that is available, decide on a budget for the sampling process.

4.2.2. "What-If" Scenarios

On the basis of the same setup as above, the user may wish to perform counterfactual reasoning, also known as *what-if* scenarios. In this case, instead of taking facts and EM variable settings from direct observations, the system allows for specifying scenarios as desired and shows the resulting probabilities.

Figure 3 illustrates this functionality with the same registered queries as before, showing how their associated probability intervals change under two scenarios. In the first, the analyst wants to know how the probabilities associated with the above queries change in case that the *token impersonation* technique is very likely to be implemented successfully, as reported by CAPEC:

$$likelihoodAttack(CAPEC\text{-}633, high).$$

The most drastic change is in the first query, which now yields a probability between 85 and 95%, while the other query's probability increases somewhat to 90%–100%. This is because token impersonation (CAPEC-633) is a technique that, if it has a high likelihood of success, is directly linked to privileged account management (mitigation strategy M1026).

In the second scenario, the analyst wants to know how the probabilities would change if *user account management* is added as a new mitigation strategy ($new\_mitigation(M1018)$). Now, the query:

$$pos\_threat(T1134, SO344)$$

becomes less probable (23%–50%, since the new mitigation strategy helps in preventing the T1134 technique), while for this scenario, the answer to the other query remains unchanged, since the two mitigation strategies are unrelated.

4.2.3. Next Steps: Explainability

In addition to being able to calculate query probabilities, it is possible to accompany such results with an *explanation* as to how the system arrived at that answer; explainability was recently identified as a key feature in cybersecurity domains [28]. We discuss two proposals for providing such insights into the kind of results presented in the previous sections. The first is centered on the probabilistic model (EM), while the second focuses on the rules used to derive query answers (AM).

Most Probable Scenarios. As a combination of the previous two functionalities, the system can compute a set of the k most probable scenarios given the current set of observations. In the current implementation, which uses Bayesian networks to specify the probability distribution in the EM, this set can be computed by the probabilistic model module by returning the *most probable explanations* (MPEs) of the BN given the current evidence in the EM. Then, the result of this first step can be combined with the counterfactual analysis described above and each scenario can be explored taking into account its probability of occurrence and its consequences.

Though this kind of analysis is centered on the probabilistic model, knowing the most probable scenarios is a first step towards explaining why a given query is entailed with a certain probability interval. For instance, an analyst may be interested in knowing why the upper bound is lower than expected, and being shown a high-probability scenario in which the negation of the query is entailed would be a first explanation. If further details are needed, explanations can also be derived by analyzing the rules and arguments involved in the derivations, as discussed next.

Rule-based Explanations. Another possibility is to show the arguments that support the query in the subprogram generated by a particular scenario or set of scenarios. This provides the analyst with the set of rules and facts involved in the derivation, and precisely what role they played, which may highlight the need to revise one or more of these components (for example, facts coming from an outdated data source); an approach in this direction was recently reported in [29]. Another benefit of rule-based approaches is that they can be rendered more interpretable by, for instance, using templates to translate rules into natural language, as proposed in [30]. Lastly, it is also possible to show the user minimal sets of EM elements (BN variables or worlds) that allow for the generation of supporting arguments for the query, thus pointing to the uncertain elements that play a role in the logical derivations of interest.

As a concluding remark, taking into account the general considerations of *explainable AI* approaches [2], we consider that adding a probabilistic module to a platform like DAQAP provides additional possibilities for building explanations. On the one hand, as explained in Section 2.2, the answers in P-DAQAP consist of probability intervals that represent two types of uncertainty (probabilistic and epistemic), which allows for us to provide more information about the nature of knowledge that is being processed. On the other hand, as previously detailed, it is possible to accompany the answers with different types of explanations, which demonstrates the potential of involving the probabilistic component when generating explanations. All this accompanying information provides analysts with tools that allow for them to confidently accept the obtained answer, or revise pieces of information or knowledge that do not apply to the current situation.

5. Empirical Evaluation

We now report on the results of a preliminary empirical evaluation designed to test the effectiveness and efficiency of a world sampling-based approximation to query answering in DeLP3E. We used Bayesian networks for the EM and sampled directly from the distributions they encode. The experiments focus on varying three key dimensions: *number of random variables* (which determines the number of possible worlds), *number of sampled worlds*, and the *entropy* of the probability distribution associated with the EM. Intuitively, entropy is a measure of disorder. For probability distributions, it measures how "spread out" the probability mass is over the space of possible worlds, so a low value indicates a highly concentrated mass. Extreme cases thus range from a single world having probability one, to all worlds having the same probability.

All runs were performed on a computer with an Intel Core i5-5200U CPU at 2.20GHz and 8GB of RAM under the 64-bit Debian GNU/Linux 10 OS. Probability computations were carried out using the pyAgrum (https://agrum.gitlab.io, accessed on 21 August 2022) Python library.

5.1. Experimental Setup

All problem instances (DeLP3E knowledge bases and queries) were synthetically generated to be able to adequately control the independent variables in our analysis. To obtain an instance, we first randomly generate the AM as a classical DeLP program with a balanced set of facts and rules; rule bodies and heads are generated in such a way as to ensure overlap, in order to yield nontrivial arguments (see [31] for details on such a procedure). The general design of the program generator consists of the following steps:

1. Generating the basic components on which the more complex structures are created, that is, facts and assumptions are generated first.
2. Arguments are organized in *levels*, where each level indicates the maximal number of rules used in its derivation chain until a basic element is reached.
3. Dialectical trees are generated only for top-level arguments because they have a greater number of possible points of attack, given that they have more elements in their body.

For the Bayesian networks in the EM, we randomly generated a graph on the basis of the desired number of EM variables (and a random number of edges set to the number of nodes as a maximum) using the `networkx` library (https://networkx.github.io, accessed on 21 August 2022). To control the entropy of the encoded distribution, we took each node probability table entry and randomly choose between *true* and *false*; then, we randomly assigned a probability to that outcome in the interval $[\alpha, 1]$, where α is a parameter varied in $\{0.7, 0.9\}$.

Annotation functions are lastly randomly generated by assigning to each element in the AM an element randomly chosen from the set of (possibly negated) EM variables plus "*true*" (AM elements annotated with *true* hold in all worlds).

Quality Metric. Given a probability interval $i_1 = [a, b]$, we used the following metric to gauge the quality of a sound approximation $i_2 = [c, d]$ (that is $[a, b] \subseteq [c, d]$ always holds):

$$Q_{i_1}(i_2) = \frac{1 - (d - c)}{1 - (b - a)}$$

Intuitively, this metric calculates the probability mass that is *discarded* by one interval in relation to another. The resulting value is always a real number in $[0, 1]$, where a value of zero indicates the poorest possible approximation ($[0, 1]$, which is always a sound approximation for problem instance), and a value of 1 yields the best possible approximation, which corresponds precisely with the exact interval. Thus, we generally apply this metric by using the result of the exact algorithm in the numerator and an approximation in the denominator.

5.2. Results

Figure 4a shows the average running time taken *per sample* over all configurations based on a set of 100 runs. We calculated the running time in this manner to adequately compare the times for the different EM sizes. Even though the impact of this dimension on individual running time is not significant, it may become so when sampling hundreds of thousands of worlds. For example, consider the difference between running time per sample for 1 billion worlds vs. 1 million worlds: $0.0289420 - 0.0289165 = 0.0000255$ s; for a sample size of 100,000 worlds, this difference amounts to 2.55 s. In the third column, we include an estimation of running times of the brute-force algorithm based on these values. Both running times are worst-case since optimization is possible (for instance, in our system we avoid recomputing warrant statuses of induced subprograms for which these values had been computed).

Figure 4b shows results concerning approximation quality; the metric was calculated with respect to the exact result for up to 20 EM variables ($\approx$1M worlds). For the case of 30 EM variables ($\approx$1B worlds), we approximated the metric using 250,000 worlds (which amounts to approximately 0.023% of the set of possible worlds), since the exact algorithm becomes intractable for instances of this size.

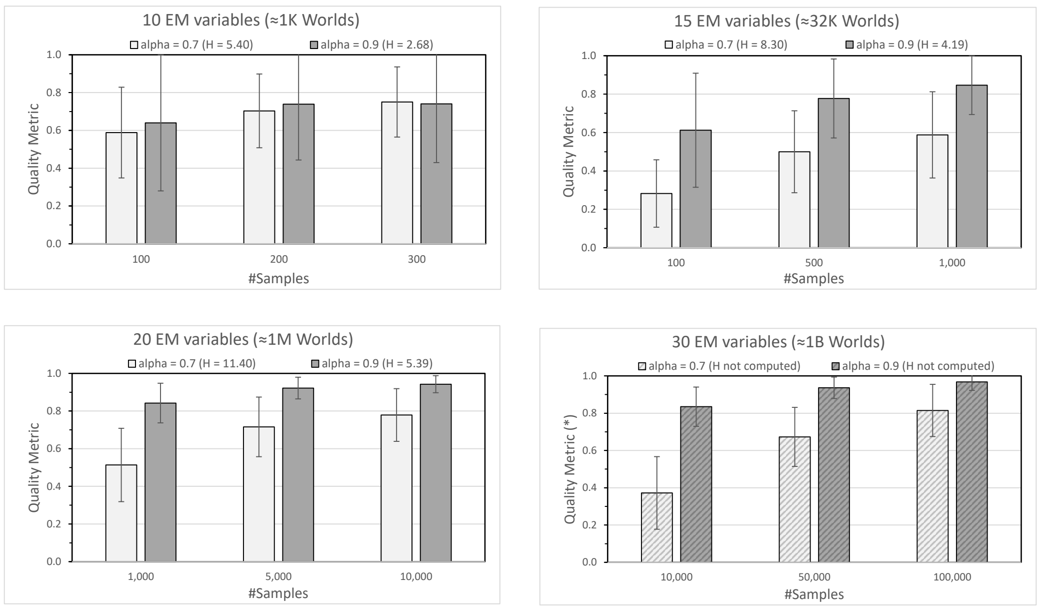
The following general observations arise from these results:

- First, sampling larger sets of worlds leads to higher quality approximations. Though this is expected, there are two interesting details:
  1. For the 20 EM variable case, the quality obtained by 5000 vs. 10,000 samples was not statistically significant (two-tailed two-sample unequal variance Student's t-tests yielded p-values greater than 0.08 for $\alpha = 0.7$ and greater than 0.16 for $\alpha = 0.9$), which means that only 5000 samples sufficed to obtain a good approximation.
  2. The proportion of repeated samples (i.e., wasted effort) was quite high for both entropy levels; for $\alpha = 0.7$ (higher entropy) on average 52% of samples were repeated, while for $\alpha = 0.9$ (lower entropy), an average of 87% were not unique. For the 20 EM variable case, the quality levels were achieved with only 2293 and 469 unique samples, respectively. Larger sample sizes also lead to lower variation in quality (shorter error bars).
- Next, entropy noticeably impacted solution quality (except for 10 EM variables, the smallest setting). Since our approximation algorithm samples worlds directly from the BN's distribution, it is natural to observe better effectiveness with lower (less spread out) entropy distributions. A smaller number of worlds represents a larger portion of the probability mass.
- Lastly, even for higher values of entropy, we observed adequate quality levels for modest numbers of samples compared to the size of the full sample space.

These results shed light on the applicability of P-DAQAP on real-world problems such as the CTA use case, given that relatively low numbers of effective (i.e., nonrepeated) samples yield good approximations of the exact values.

#EM Variables	Run. Time/Sample (seconds)	Est. Brute Force Run. Time (hours)
10 (1K worlds)	0.0286015	0.008
15 (32K worlds)	0.0288155	0.262
20 (1M worlds)	0.0289165	8.422
30 (1B worlds)	0.0289420	8632 (≈360 days)

(**a**)



(**b**)

Figure 4. (**a**) Average running times per world sampled ($n = 100$ runs). For each case, we estimate the running time (in hours) required to run the exact (brute force) algorithm. (**b**) Average solution quality varying #EM variables (log of #worlds), #samples, and the parameter that controls the entropy (H) of the probability distribution. For 30 EM variables (1B worlds, bottom right), quality is *approximated* on the basis of a sample of 250,000 worlds. Error bars correspond to standard deviation ($n > 50$ for the top charts, $n > 15$ for the bottom charts).

5.3. Results in the Context of Practical Applications

We now analyze the results we obtained in these experiments in the context of the MITRE ATT and CK data that we focused on for our use case in Section 3. For the purposes of this brief analysis, let us consider the Enterprise segment of the dataset, which contains 191 techniques and 385 subtechniques, and this translates into a large number of constants that would certainly lead to an intractable probabilistic model if tackled directly. Fortunately, there is a well-understood independence relation among such techniques, and they can, thus, be effectively pruned depending on the tactics to which they are associated. For instance, the *Privilege Escalation* tactic (TA0004) that we refer to in the use case has 13 associated techniques, while the rest of the techniques in the dataset associated at most 30 (with the exception of *Defense Evasion* (TA0005) that has 42, though additional filtering according to the specific operating system in question allows to bring this number down significantly). Our preliminary results therefore show that having the capacity to scale to 30 EM variables is within the realm of this kind of application, though further efforts are required to effectively arrive at submodels derived from the general one that can be used to solve specific query answering tasks. In this same vein, there are multiple research and development efforts to manipulate, adapt, and export data and knowledge from the ATT and CK dataset [32–35].

6. Conclusions and Future Work

We presented an extension of the DAQAP platform to incorporate probabilistic knowledge bases, giving rise to the P-DAQAP system, which is, to the best of our knowledge, the first system of its kind for probabilistic defeasible reasoning. After discussing the details of its design and describing applications to cybersecurity, we performed an empirical evaluation whose goal it was to explore the effectiveness and efficiency of world sampling-based approximate query answering. Our study showed that the entropy associated with the probability distribution over worlds has a large impact on expected solution quality, but even a modest number of samples suffices to reach good-quality approximations. Compared to classical (nonprobabilistic) approaches, the results of our experiments show that P-DAQAP allows for representing, effectively and efficiently reasoning with different types of uncertainty, modeling complex domains in more detail, and providing more informed answers that can be accompanied by explanations. In critical environments, having outputs of this kind increases credibility and trust in the system by its users.

Future work involves carrying out a broader evaluation investigating other sampling methods, avoiding repeated samples, and testing other probabilistic models. One of the goals of this research line is to develop a method to guide knowledge engineering efforts on the basis of domain features, requirements in terms of expressive power, approximation quality, and query response time.

Author Contributions: Conceptualization, M.A.L., A.J.G., P.S., and G.I.S.; methodology, M.A.L., P.S. and G.I.S.; validation, M.A.L., P.S. and G.I.S.; formal analysis, G.I.S.; investigation, M.A.L. and G.I.S.; writing—original draft, M.A.L.; writing—review and editing, M.A.L., A.J.G., P.S. and G.I.S.; project administration, G.I.S.; supervision and funding acquisition, A.J.G. and G.I.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universidad Nacional del Sur (UNS) grant numbers PGI 24/ZN34 and PGI 24/N046, Universidad Nacional de Entre Ríos grant number PDTS-UNER 7066, and Agencia Nacional de Promoción Científica y Tecnológica, Argentina grant number grants PICT-2018-0475 (PRH-2014-0007). P.S. is supported by internal funding from the ASU Fulton Schools of Engineering.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: the authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AM	Analytical Model
CAPEC	Common Attack Pattern Enumeration and Classification
CPE	Common Platform Enumeration
CTA	Cyberthreat Analysis
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DeLP	Defeasible Logic Programming
DeLP3E	Defeasible Logic Programming with Presumptions and Probabilistic Environments
EM	Environmental Model
KB	Knowledge Base
P-DAQAP	Probabilistic Defeasible Argumentation Query Answering Platform
NVD	National Vulnerability Database
XAI	Explainable Artificial Intelligence

References

1. Mumford, E. The story of socio-technical design: Reflections on its successes, failures and potential. *Inf. Syst. J.* **2006**, *16*, 317–342.
2. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* **2019**, *267*, 1–38.
3. Arrieta, A.B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **2020**, *58*, 82–115.
4. Gunning, D. Explainable Artificial Intelligence (XAI). Defense Advanced Research Projects Agency (DARPA). 2017. Available online: https://nsarchive.gwu.edu/sites/default/files/documents/5794867/National-Security-Archive-David-Gunning-DARPA.pdf (accessed on 21 August 2022).
5. Viganò, L.; Magazzeni, D. Explainable security. In Proceedings of the 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 7–11 September 2020; pp. 293–300.
6. Castelvecchi, D. Can we open the black box of AI? *Nat. News* **2016**, *538*, 20.
7. Mahdavifar, S.; Ghorbani, A.A. DeNNeS: deep embedded neural network expert system for detecting cyber attacks. *Neural Comput. Appl.* **2020**, *32*, 14753–14780.
8. Kuppa, A.; Le-Khac, N.A. Black Box Attacks on Explainable Artificial Intelligence (XAI) methods in Cyber Security. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
9. Szczepański, M.; Choraś, M.; Pawlicki, M.; Kozik, R. Achieving explainability of intrusion detection system by hybrid oracle-explainer approach. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
10. Malatji, M.; Sune, V.S.; Marnewick, A. Socio-technical systems cybersecurity framework. *Inf. Comput. Secur.* **2019**, *27*, 233–272.
11. Alsmadi, I. *The NICE Cyber Security Framework: Cyber Security Management*; Springer Nature: Cham, Switzerland, 2020.
12. Leiva, M.A.; Simari, G.I.; Simari, G.R.; Shakarian, P. Cyber threat analysis with structured probabilistic argumentation. In Proceedings of the AI^3. CEUR-WS, Rende, Italy, 19–22 November 2019; Volume 2528, pp. 50–64.
13. Shakarian, P.; Simari, G.I.; Moores, G.; Parsons, S.; Falappa, M.A. An Argumentation-based Framework to Address the Attribution Problem in Cyber-Warfare. In Proceedings of the CyberSecurity, ASE, Stanford, CA, USA, 27–31 May 2014.
14. Kuppa, A.; Le-Khac, N.A. Adversarial xai methods in cybersecurity. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 4924–4938.
15. Liu, H.; Zhong, C.; Alnusair, A.; Islam, S.R. FAIXID: A framework for enhancing ai explainability of intrusion detection results using data cleaning techniques. *J. Netw. Syst. Manag.* **2021**, *29*, 1–30.
16. Srivastava, G.; Jhaveri, R.H.; Bhattacharya, S.; Pandya, S.; Rajeswari; Maddikunta, P.K.R.; Yenduri, G.; Hall, J.G.; Alazab, M.; Gadekallu, T.R. XAI for Cybersecurity: State of the Art, Challenges, Open Issues and Future Directions. *arXiv* **2022**. arXiv:2206.03585.
17. Hariharan, S.; Velicheti, A.; Anagha, A.; Thomas, C.; Balakrishnan, N. Explainable Artificial Intelligence in Cybersecurity: A Brief Review. In Proceedings of the 2021 4th International Conference on Security and Privacy (ISEA-ISAP), Dhanbad, India, 27–30 October 2021, pp. 1–12.
18. Shakarian, P.; Simari, G.I.; Moores, G.; Paulo, D.; Parsons, S.; Falappa, M.A.; Aleali, A. Belief revision in structured probabilistic argumentation. *AMAI* **2016**, *78*, 259–301.
19. Leiva, M.A.; Simari, G.I.; Gottifredi, S.; García, A.J.; Simari, G.R. DAQAP: Defeasible Argumentation Query Answering Platform. In Proceedings of the FQAS 2019, Amantea, Italy, 2–5 July 2019; pp. 126–138.
20. Simari, G.R.; Loui, R.P. A mathematical treatment of defeasible reasoning and its implementation. *Artif. Intell.* **1992**, *53*, 125–157.
21. Toni, F. A tutorial on assumption-based argumentation. *Argum. Comput.* **2014**, *5*, 89–117.
22. Modgil, S.; Prakken, H. The ASPIC+ framework for structured argumentation: A tutorial. *Argum. Comput.* **2014**, *5*, 31–62.
23. García, A.J.; Simari, G.R. Defeasible logic programming: DeLP-servers, contextual queries, and explanations for answers. *Argum. Comput.* **2014**, *5*, 63–88.
24. Besnard, P.; Garcia, A.; Hunter, A.; Modgil, S.; Prakken, H.; Simari, G.; Toni, F. Introduction to structured argumentation. *Argum. Comput.* **2014**, *5*, 1–4.
25. Martinez, M.V.; García, A.J.; Simari, G.R. On the Use of Presumptions in Structured Defeasible Reasoning. In *COMMA*; Verheij, B., Szeider, S., Woltran, S., Eds.; IOS Press: Amsterdam, The Netherlands, 2012; Volume 245, pp. 185–196.
26. Suciu, D.; Olteanu, D.; Ré, C.; Koch, C. Probabilistic databases. *Synth. Lect. Data Manag.* **2011**, *3*, 1–180.
27. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann: San Francisco, CA, USA, 1988.
28. Paredes, J.; Teze, J.C.; Simari, G.I.; Martinez, M.V. On the Importance of Domain-specific Explanations in AI-based Cybersecurity Systems (Technical Report). *arXiv* **2021**. arXiv:2108.02006.
29. Buron Brarda, M.E.; Tamargo, L.H.; García, A.J. Using Argumentation to Obtain and Explain Results in a Decision Support System. *IEEE Intell. Syst.* **2021**, *36*, 36–42. https://doi.org/10.1109/MIS.2020.3042740.
30. Grover, S.; Pulice, C.; Simari, G.I.; Subrahmanian, V.S. BEEF: Balanced English Explanations of Forecasts. *IEEE Trans. Comput. Soc. Syst.* **2019**, *6*, 350–364.
31. Alfano, G.; Greco, S.; Parisi, F.; Simari, G.I.; Simari, G.R. Incremental computation for structured argumentation over dynamic DeLP knowledge bases. *Artif. Intell.* **2021**, *300*, 103553.

32. Al-Shaer, R.; Spring, J.M.; Christou, E. Learning the Associations of MITRE ATT & CK Adversarial Techniques. In Proceedings of the 2020 IEEE Conference on Communications and Network Security (CNS), Avignon, France, 29 June–1 July 2020; pp. 1–9. doi:10.1109/CNS48642.2020.9162207.
33. Kuppa, A.; Aouad, L.; Le-Khac, N.A. Linking CVE's to MITRE ATT&CK Techniques. In Proceedings of the 16th International Conference on Availability, Reliability and Security, Vienna, Austria, 17–20 August 2021; pp. 1–12.
34. Hong, S.; Kim, K.; Kim, T. The Design and Implementation of Simulated Threat Generator based on MITRE ATT&CK for Cyber Warfare Training. *J. Korea Inst. Mil. Sci. Technol.* **2019**, *22*, 797–805.
35. Choi, S.; Yun, J.H.; Min, B.G. Probabilistic attack sequence generation and execution based on mitre att&ck for ics datasets. In Proceedings of the Cyber Security Experimentation and Test Workshop, Virtual, CA, USA, 9 August 2021; pp. 41–48.