Check for updates

Received: 29 November 2022 | Revised: 20 February 2023 | Accepted: 12 April 2023

DOI: 10.1002/spe.3212

RESEARCH ARTICLE

WILEY

A web application for reasoning on probabilistic description logics knowledge bases

Riccardo Zese[1] | Elena Bellodi[2]

[1]Department of Chemical, Pharmaceutical and Agricultural Sciences, University of Ferrara, Ferrara, Italy

[2]Department of Engineering, University of Ferrara, Ferrara, Italy

Correspondence
Riccardo Zese, Department of Chemical, Pharmaceutical and Agricultural Sciences, University of Ferrara, Via Borsari 46, I-44121 Ferrara, Italy.
Email: riccardo.zese@unife.it

Funding information
None.

Abstract

The aim of the Semantic Web is making information and resources from the Web automatically processable by machines. Usually, the uncertainty characterizing much of this information is addressed by means of a probabilistic semantics. Following the vision of a "Probabilistic Semantic Web", a plethora of probabilistic semantics have been proposed: some of them change the syntax and/or the semantics itself of the knowledge representation language, others allow one to annotate axioms of a knowledge base with a probability value. Among the latter, the DISPONTE semantics exploits probabilistic annotations to extend query answering with the capability of returning the probability of a query being true in a domain. In order to promote the adoption of Probabilistic Semantic Web we first developed BUNDLE, a framework that can exploit different underlying (probabilistic and non-probabilistic) reasoners to perform probabilistic inference under the DISPONTE semantics. In this paper we present a web application for BUNDLE, to show how DISPONTE is easily usable even in already established applications and systems. It allows users to query a DISPONTE knowledge base written or uploaded directly in the application interface by using just a web browser, without the need to install any software on their machine. It is accessible on the web at https://bundle.ml.unife.it/ and also provides some examples for familiarizing with the application. The results of a usability evaluation involving human participants are also reported, showing the relevance and the practical impact of the tool and possible ways for improvement.

KEYWORDS

description logics, inference, probabilistic description logics, semantic web, web applications

1 | INTRODUCTION

The term "Semantic Web" refers to World Wide Web Consortium's vision of the *Web of linked data*. The ultimate goal of the Web of data is "to enable computers to do more useful work and to develop systems that can support trusted interactions over the network".[1] The Consortium standardized a family of knowledge representation formalisms for this

Abbreviations: DL, description logic; KB, knowledge base; BDD, binary decision diagram; BUNDLE, BDDs for uncertain reasoning on description logic theories; DISPONTE, distribution semantics for probabilistic ontologies.

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.
© 2023 The Authors. *Software: Practice and Experience* published by John Wiley & Sons Ltd.

data, called Web Ontology Language (OWL), which became a W3C recommendation in 2012 and is now at its second version[2] (OWL 2). These formalisms provide languages to model domains where it is necessary to represent specific information and to perform inference on it, such as software engineering, medical diagnosis, digital libraries, databases, and web-based information systems. These domain models are called ontologies, or knowledge bases (KBs). OWL 2 is designed to facilitate ontology development and sharing via the Web, with the ultimate goal of making web content more accessible to machines. Description Logics (DLs) are particularly useful for representing ontologies and have been adopted as the basis of the Semantic Web. They are fragments of First-Order Logic possessing nice computational properties such as decidability and, sometimes, low complexity.[3]

In the last decades, it has become evident that the Semantic Web data is often uncertain, so it is extremely important to represent and reason with uncertain information. Application areas in which uncertain aspects of domain concepts need to be modelled can greatly benefit from uncertainty representation of any kind in DLs. For instance, almost every bio-medical ontology contains uncertain concepts of some sort. The well-known medical ontology SNOMED CT[4] comprises a variety of classes with names such as “Probable tubo-ovarian abscess”, “Natural death with probable cause suspected”, “Probable diagnosis”. Other than the medical field, a wider use of such formalisms could positively affect many other domains. Applications of (certain) ontologies can be found in software engineering and software technology[5,6] for sharing knowledge of the problem domain and using a common terminology among all stakeholders and filtering the knowledge when defining models and metamodels. The W3 Consortium proposed a document for “potential uses of the Semantic Web in Systems and Software Engineering”.[*] Ontologies have been used also in the legal domain.[7] We believe that all these fields could benefit from an uncertain representation, as well as, for example, fault diagnosis or recommender systems, where probability could be used, in the former case, to calculate a fault risk value, showing whether a production line could be affected by a fault and, in the latter case, to rank possible recommendations. These needs motivated many researchers to study and present probabilistic semantics capable of handling the uncertainty inherent in the Semantic Web. For example, Fuzzy DLs have been widely studied in the literature for representing vagueness,[8-11] based on the idea that concepts and roles can be interpreted as fuzzy sets and fuzzy binary relations. Giordano et al.[12] show that the input/output behavior of self-organizing maps, which have been proposed as possible ways to explain the psychological mechanisms underlying category generalization, after training can be described by a fuzzy description logic interpretation as well as by a preferential interpretation. Yu et al.[13] propose a Causal Probability Description Logic framework which introduces a “causality probability” concept into the KB level of the Probabilistic Description Logic.[14] The goal is to help learn comprehensive health causal knowledge and relations among the diseases and symptoms from the UK National Health Service website linking with DBpedia datasets.

DISPONTE, for “distribution semantics for probabilistic ontologies”,[15-17] instead allows the user to annotate axioms of a knowledge base with a probability representing the degree of belief in their truth. It is inspired by the distribution semantics[18] proposed in the field of Probabilistic Logic Programming (PLP). This semantics for PLP allows one to compute the probability of a query from a probability distribution over normal logic programs.

In the literature several non-probabilistic reasoners are available—Pellet,[19] Fact++,[20] JFact,[†] HermiT[21]—which extract implicit information from ontologies, as well as probabilistic reasoners—Pronto,[22] BORN,[23] ONTO2PROBLOG,[24] TRILL,[25,26] TRILL^P[27] and TORNADO[28]—which can compute the probability of a query posed on a probabilistic DL.

BUNDLE (“BDDs for uncertain reasoning on description logic theories”)[29,30] is a system for performing probabilistic inference from DISPONTE knowledge bases by means of Binary Decision Diagrams (BDDs): BUNDLE computes the probability of queries from a covering set of explanations returned by an underlying reasoner. Initially built for exploiting Pellet as an underlying reasoner to collect explanations, it was extended[31] in order to: (1) interface with Fact++, JFact, and HermiT in the same way as Pellet; (2) interface with TRILL, TRILL^P and TORNADO which, instead, do not require the conversion of explanations into a BDD.

In this paper, we present a web application for BUNDLE for performing inference on user-defined KBs under DISPONTE (https://bundle.ml.unife.it/). The application allows the user to upload or write a KB in OWL – using several syntaxes such as Resource Description Framework (RDF)/XML, Turtle, or OWL/XML – and query it. The user can choose the maximum number of explanations to be returned and a reasoner among the ones with which BUNDLE interfaces. Then the query, the ontology and the user settings are sent to the server which sends back the answer. BUNDLE can answer different types of queries and can compute their probability, whether a query is entailed from the KB, the set of explanations that explain the entailment and the execution time split in the different inference phases.

[*] https://www.w3.org/2001/sw/BestPractices/SE/ODA/060103/.

[†] http://jfact.sourceforge.net/index.html.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

The first goal of this application is to demonstrate the ease of adopting and reasoning under the DISPONTE semantics. In fact, DISPONTE exploits the built-in annotations in the OWL language to add uncertainty in the KB, allowing one to integrate many systems designed for OWL in BUNDLE without modifying them internally, but only exploiting them to gather the information needed to calculate the probability of the query. The second goal is popularizing the use of probabilistic DLs, making it easy to test the potential of probabilistic DL reasoners, and addressing the lack of web interfaces for probabilistic reasoning on ontologies.

The web application is inspired by our previous work,[32] where a web interface for the TRILL, TRILL^P and TORNADO reasoners was presented. Besides them, to the best of our knowledge, there is no other application for probabilistic Semantic Web reasoning. In a broader sense, we can mention cplint-on-SWISH[33] and the ProbLog[34] online editor as other web applications for probabilistic reasoning, but they are tailored to PLP languages. For the OWL language, WebProtégé[35] is the web application of the well-known KB editor Protégé.[‡] However, despite the stand-alone version of Protégé can encapsulate (non-probabilistic) reasoners via plug-ins, its online counterpart does not allow to reason on the designed KB. Differently from WebProtégé, the BUNDLE web application can answer queries (also w.r.t. non-probabilistic KBs) without the need to install any software except a browser, thus it is available to everyone. On the other hand, without the web application, to try BUNDLE (or any other reasoners wrapped into BUNDLE except for TRILL) the user would have three different possibilities: (1) download the jar file or compile the code[§] after installing at least a Java Runtime Environment (similarly to the stand-alone version of Protégé), or (2) create an application importing BUNDLE as a dependency (possibly using Maven[¶]), or (3) download the Docker image from the Docker Hub.[#] These methods require a more advanced computer science knowledge.

We validated the application through a questionnaire administered to an audience of researchers from all over the world, reached via various mailing lists in the field of Semantic Web and DLs. The questionnaire consists of the Post-Study System Usability Questionnaire (PSSUQ) Version 3,[36,37] to which some questions specific to the application were added. The choice of PSSUQ is due to the fact that its results are significant even with few answers, which can help in questionnaires such as this one, where it is not known in advance how many people are going to fill it out.

The paper is organized as follows: Section 2 and 3 introduce Description Logics and probabilistic Description Logics, respectively. Section 4 describes the BUNDLE framework and Section 5 the Web application with some running examples. Section 6 discusses the results of the usability questionnaire, used to evaluate and validate the potentialities of the application. Section 7 concludes the paper.

2 | DESCRIPTION LOGICS

Descriptions Logics (DLs) are a family of knowledge representation formalisms that differ in how concepts (sets of individuals of the domain) and roles (sets of pairs of individuals) can be defined. In order to illustrate DLs, we describe in the following $\mathcal{SROIQ}(\mathbf{D})$, one of the most common fragments. It was introduced by Horrocks et al.[38] and it is of particular importance being semantically equivalent to OWL 2.

Let $\mathbf{C}$ be a set of *atomic concepts*, $\mathbf{R}$ a set of *atomic roles* and $\mathbf{I}$ a set of individuals. A *role* could be an atomic role $R \in \mathbf{R}$, the inverse R^- of an atomic role $R \in \mathbf{R}$ or a complex role $R \circ S$. We use $\mathbf{R}^-$ to denote the set of all inverses of roles in $\mathbf{R}$. Each $A \in \mathbf{C}$, $\bot$, and $\top$ are concepts and if $a \in \mathbf{I}$, then $\{a\}$ is a concept called *nominal*. If C, C_1, and C_2 are concepts and $R \in \mathbf{R} \cup \mathbf{R}^-$, then $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$, and $\neg C$ are concepts, as well as $\exists R.C$, $\forall R.C$, $\geq nR.C$, and $\leq nR.C$ for an integer $n \geq 0$.

A *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ consists of a TBox $\mathcal{T}$, an RBox $\mathcal{R}$, and an ABox $\mathcal{A}$. An RBox $\mathcal{R}$ is a finite set of *transitivity axioms Trans*(R), *role inclusion axioms* $R \sqsubseteq S$, and *role chain axioms* $R \circ P \sqsubseteq S$, where $R, P, S \in \mathbf{R} \cup \mathbf{R}^-$. A finite set of role inclusion axioms $\mathcal{R}_h$ is called *role hierarchy*. Given $\mathcal{R}_h$, we can define $\dot{\sqsubseteq}$ as the transitive-reflexive closure of $\sqsubseteq$ over $\mathcal{R} \cup \{R^- \sqsubseteq S^- | R \sqsubseteq S \in \mathcal{R}\}$. A role $R \in \mathbf{R} \cup \mathbf{R}^-$ is *simple* w.r.t. $\mathcal{R}$ if for all $S \mathrel{\dot{\sqsubseteq}} R$, S is not transitive. A *TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, where C and D are concepts. We use $C \equiv D$ to abbreviate the conjunction of $C \sqsubseteq D$ and $D \sqsubseteq C$. An *ABox* $\mathcal{A}$ is a finite set of *concept membership axioms* $a : C$ and *role membership axioms* $(a, b) : R$, where C is a concept, $R \in \mathbf{R}$ and $a, b \in \mathbf{I}$.

[‡] https://protege.stanford.edu/.

[§] Available at https://bitbucket.org/machinelearningunife/bundle/src/master/.

[¶] https://maven.apache.org/.

[#] Available at https://hub.docker.com/r/giuseta/bundle.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

A KB is usually assigned a semantics using interpretations of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function* that assigns an element in $\Delta^{\mathcal{I}}$ to each individual a, a subset of $\Delta^{\mathcal{I}}$ to each concept C, and a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role R. The mapping $\cdot^{\mathcal{I}}$ is extended to complex concepts as follows (where $R^{\mathcal{I}}(x, C) = \{y | \langle x, y\rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$ and $\#X$ denotes the cardinality of the set X):

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}}, \\
\bot^{\mathcal{I}} &= \emptyset, \\
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\}, \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
(C_1 \sqcup C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}, \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}, \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \cap C^{\mathcal{I}} \neq \emptyset\}, \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \subseteq C^{\mathcal{I}}\}, \\
(\geq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \geq n\}, \\
(\leq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \leq n\}, \\
(R^{-})^{\mathcal{I}} &= \{(y, x) | (x, y) \in R^{\mathcal{I}}\}, \\
(R_1 \circ \cdots \circ R_n)^{\mathcal{I}} &= R_1^{\mathcal{I}} \circ \cdots \circ R_n^{\mathcal{I}}.
\end{aligned}$$

$\mathcal{SROIQ}(\mathbf{D})$ also allows the definition of datatype roles, which connect an individual to an element of a datatype such as integers, floats and so forth.

The *satisfaction* of an axiom E in an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, denoted by $\mathcal{I} \models E$, is defined as follows: (1) $\mathcal{I} \models Trans(R)$ iff $R^{\mathcal{I}}$ is transitive, (2) $\mathcal{I} \models R \sqsubseteq S$ iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, (3) $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, (4) $\mathcal{I} \models a : C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$, (5) $\mathcal{I} \models (a, b) : R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, (6) $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$, (7) $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. I *satisfies* a set of axioms $\mathcal{E}$, denoted by $\mathcal{I} \models \mathcal{E}$, iff $\mathcal{I} \models E$ for all $E \in \mathcal{E}$. An interpretation $\mathcal{I}$ *satisfies* a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, denoted $\mathcal{I} \models \mathcal{K}$, iff $\mathcal{I}$ satisfies $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$. In this case, we say that I is a model of $\mathcal{K}$. A KB is satisfiable if it has a model. Inference in a DL is decidable if the problem of checking the satisfiability of any possible KB representable with that DL is decidable. In particular, $\mathcal{SROIQ}(\mathbf{D})$ is decidable iff there are no number restrictions on non-simple roles. Moreover, to maintain decidability every set of (complex) role inclusions must be acyclic.

A query Q over a KB $\mathcal{K}$ is usually an axiom for which we want to test the entailment from the KB, written $\mathcal{K} \models Q$. The entailment test may be reduced to checking the unsatisfiability of a concept in the knowledge base, that is, the emptiness of the concept. For example, the entailment of the axiom $C \sqsubseteq D$ may be tested by checking the unsatisfiability of the concept $C \sqcap \neg D$, while the entailment of the axiom $a : C$ may be tested by checking the unsatisfiability of $a : \neg C$.

Example 1. Consider the following KB "Crime and Punishment" containing 4 axioms α_i:

$$\begin{aligned}
\alpha_1 &= \text{Nihilist} \sqsubseteq \text{GreatMan}, & \alpha_2 &= \exists\text{killed}.\top \sqsubseteq \text{Nihilist}, \\
\alpha_3 &= (\text{raskolnikov, alyona})\text{: killed}, & \alpha_4 &= (\text{raskolnikov, lizaveta})\text{: killed}.
\end{aligned}$$

This KB corresponds to Example 1 at https://bundle.ml.unife.it/examples. It states that if you killed someone then you are a nihilist and whoever is a nihilist is a "great man" (TBox). It also states that Raskolnikov killed Alyona and Lizaveta (ABox). The KB entails the query `Q=raskolnikov:GreatMan`. However, he does not seem to be such a great man.

3 | PROBABILISTIC DESCRIPTION LOGICS

DISPONTE,[17,25,39] for "distribution semantics for probabilistic ontologies", applies the distribution semantics[18] of Probabilistic Logic Programming to Description Logics KBs. This semantics allows one to label any axiom (either in $\mathcal{T}$ or in $\mathcal{A}$) with a real value $p \in [0, 1]$ representing a probability. These axioms are called *probabilistic axioms* and take the form

$$p :: E,$$

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

where E is a DL axiom and $p :: E$ means that we have degree of belief p in axiom E. A DISPONTE KB can be defined as a pair of sets $(\mathcal{C}, \mathcal{P})$, where $\mathcal{C}$ is the set of non-probabilistic axioms and $\mathcal{P}$ is the set of probabilistic axioms. Following this definition, every non-probabilistic KB is a DISPONTE KB where $\mathcal{P} = \emptyset$.

DISPONTE is based on the idea that a set $\mathbf{X} = \{X_i | p_i :: E_i \in \mathcal{K}\}$ of *independent* Boolean random variables X_i is associated with the probabilistic axioms, so that $P(X_i = 1) = p_i$, that is, the probability of X_i taking on value 1 is p_i, the probability of axiom E_i.

Given a query Q, DISPONTE computes its probability by building *worlds*, that is, non-probabilistic KBs to which a semantics can be usually assigned. A query is entailed by a world if it is true in every model of the world. A world can be obtained by specifying *atomic choices*: an atomic choice is a couple (E_i, k) where E_i is the ith probabilistic axiom and $k \in \{0, 1\}$, indicating whether E_i is chosen to be included in a world ($k = 1$) or not ($k = 0$). A *composite choice* κ is a *consistent* set of atomic choices, that is, $(E_i, k) \in \kappa$, $(E_i, m) \in \kappa$ implies $k = m$ (only one decision is taken for each axiom). The probability of a composite choice κ is

$$P(\kappa) = \prod_{(E_i,1)\in\kappa} p_i \prod_{(E_i,0)\in\kappa} (1 - p_i),$$

where p_i is the probability associated with axiom E_i; the product is justified by the fact that the random variables associated with the axioms are independent. A total composite choice, that is, a composite choice that contains an atomic choice (E_i, k) for every probabilistic axiom, is called a *selection*. A *selection* σ identifies a non-probabilistic KB w_σ, called a *world*, in this way: $w_\sigma = \mathcal{C} \cup \{E_i | (E_i, 1) \in \sigma\}$, where $\mathcal{C}$ is the set of certain, that is, non-probabilistic, axioms. A composite choice κ identifies a set of worlds $\omega_\kappa = \{w_\sigma | \sigma \in \mathcal{S}, \sigma \supseteq \kappa\}$, where $\mathcal{S}$ is the set of all selections. The probability of a world w_σ, $P(w_\sigma)$, corresponds to the probability of a selection σ and is given by $P(w_\sigma) = P(\sigma) = \prod_{(E_i,1)\in\sigma} p_i \prod_{(E_i,0)\in\sigma}(1 - p_i)$.

$P(w_\sigma)$ is a probability distribution over worlds. Let us indicate with $\mathcal{W}$ the set of all worlds. Given a query Q and the set of all worlds $\mathcal{W}$, the probability of Q is the sum of the probabilities of the worlds w in which the query is true:

$$P(Q) = \sum_{w\in\mathcal{W}\,:\, w\models Q} P(w).$$

It is worth noting that in case that KB has $\mathcal{P} = \emptyset$, every query Q entailed by the KB has probability $P(Q) = 1.0$. The following example illustrates inference under DISPONTE semantics.

Example 2. Let us consider the knowledge base and the query `Q=raskolnikov:GreatMan` of Example 1 where three axioms have been made probabilistic:

$$\beta_1 = 0.2 :: \text{Nihilist} \sqsubseteq \text{GreatMan}, \qquad \alpha_2 = \exists\text{killed}.\top \sqsubseteq \text{Nihilist},$$
$$\beta_2 = 0.6 :: (\text{raskolnikov, alyona}): \text{killed}, \qquad \beta_3 = 0.7 :: (\text{raskolnikov, lizaveta}): \text{killed}.$$

Here $\mathcal{C} = \{\alpha_2\}$. We then believe that whoever is a nihilist is a "great man" with probability 0.2 (β_1) and Raskolnikov killed Alyona and Lizaveta with probability 0.6 and 0.7 respectively (β_2 and β_3). This KB has eight worlds and Q is true in three of them, corresponding to the selection σ:

$$\{\ \{(\beta_1, 1), (\beta_2, 1), (\beta_3, 1)\},\ \{(\beta_1, 1), (\beta_2, 1), (\beta_3, 0)\},\ \{(\beta_1, 1), (\beta_2, 0), (\beta_3, 1)\}\ \}.$$

The probability is $P(Q) = 0.2 \cdot 0.6 \cdot 0.7 + 0.2 \cdot 0.6 \cdot (1 - 0.7) + 0.2 \cdot (1 - 0.6) \cdot 0.7 = 0.176$.

However, as the number of worlds is exponential in the number of probabilistic axioms, for large KBs it is infeasible to find all the worlds where a query is true. To reduce reasoning time, we can resort to finding *justifications* for the query and then compute the probability of the query from them. The problem of finding the set of justifications for a given query is a non-standard reasoning service useful for tracing derivations and debugging ontologies. This problem has been investigated by various authors.[40-43] The definition of justification can be derived from that of "explanation".

An *explanation* for a query Q is a subset of logical axioms $\mathcal{E}$ of a KB $\mathcal{K}$ such that $\mathcal{E} \models Q$. A justification is an explanation minimal w.r.t. set inclusion. Formally, an explanation $\mathcal{J} \subseteq \mathcal{K}$ is a justification if, for all $\mathcal{J}' \subset \mathcal{J}$, $\mathcal{J}' \not\models Q$, that is, $\mathcal{J}'$ is not an explanation for Q. According to DISPONTE, *a composite choice κ is an explanation for* Q if Q is entailed by every world of ω_κ.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

In particular, algorithms find a covering set of explanations for the query, where a set of composite choices $\boldsymbol{K}$ is *covering* with respect to Q if every world $w_\sigma \in \mathcal{W}$ in which Q is entailed is in ω_κ. In other words, a covering set $\boldsymbol{K}$ identifies all the worlds in which Q succeeds.

In practice, the problem of computing the probability of a query is reduced to that of finding a covering set of justifications and then transforming it into a covering set of *pairwise incompatible* explanations. Two composite choices κ_1 and κ_2 are *incompatible* if their union is not consistent. A set $\boldsymbol{K}$ of composite choices is *pairwise incompatible* if for all $\kappa_1 \in \boldsymbol{K}$, $\kappa_2 \in \boldsymbol{K}$, $\kappa_1 \neq \kappa_2$ implies that κ_1 and κ_2 are incompatible. The probability of a pairwise incompatible set of composite choices is $P(\boldsymbol{K}) = \sum_{\kappa\in\boldsymbol{K}} P(\kappa)$.

Given a query Q and a covering set of pairwise incompatible explanations $\boldsymbol{K}$, the probability of Q is:

$$P(Q) = \sum_{w_\sigma\in\omega_{\boldsymbol{K}}} P(w_\sigma) = P(\omega_{\boldsymbol{K}}) = P(\boldsymbol{K}) = \sum_{\kappa\in\boldsymbol{K}} P(\kappa) \tag{1}$$

where $\omega_{\boldsymbol{K}}$ is the set of worlds identified by the set of explanations $\boldsymbol{K}$.

The set of all justifications for the query Q w.r.t. a KB $\mathcal{K}$ *is the covering set* of justifications for Q, denoted by ALL-JUST$(Q, \mathcal{K})$. This set can be retrieved through non-probabilistic justification finding algorithms. The problem of enumerating all justifications that entail a given query is called *axiom pinpointing*.[41]

Example 3. Consider the KB and the query `Q=raskolnikov:GreatMan` of Example 2. We have the following covering set of pairwise incompatible explanations for the query: $\boldsymbol{K} = \{\ \{(\beta_1, 1), (\beta_2, 1)\},\ \{(\beta_1, 1), (\beta_2, 0), (\beta_3, 1)\}\ \}$. The probability of the query, according to formula (1), is $P(Q) = 0.2 \cdot 0.6 + 0.2 \cdot 0.4 \cdot 0.7 = 0.176$, confirming the result obtained in Example 2.

4 | INFERENCE WITH THE BUNDLE FRAMEWORK

The reasoner BUNDLE[17,29,30] computes the probability of a query w.r.t. DISPONTE KBs by first computing all the justifications for the query, then converting them into a pairwise incompatible covering set of explanations by a Disjunctive Normal Form (DNF) Boolean formula $f_{\text{ALL-JUST}(Q,\mathcal{K})}(\mathbf{X}) = \bigvee_{\kappa\in\text{ALL-JUST}(Q,\mathcal{K})} \bigwedge_{(E_i,1)\in\kappa} X_i$.

The probability that $f_{\text{ALL-JUST}(Q,\mathcal{K})}(\mathbf{X})$ takes value 1 corresponds to the probability of Q and can be computed by translating $f_{\text{ALL-JUST}(Q,\mathcal{K})}(\mathbf{X})$ into the language of Binary Decision Diagrams (BDDs). A BDD is a rooted graph used to represent a function of Boolean variables, with one level for each variable. Each node of the graph has two children corresponding either to the 1 value or the 0 value of the variable associated with the node. Its leaves are either 0 or 1. BDDs are built combining simpler BDDs with Boolean operators. Once the diagram is built, function PROBABILITY described by Kimmig and colleagues,[44] a simple dynamic programming algorithm, can be applied to efficiently compute the probability of Q at the root of the graph.

Example 4 (Example 2 cont.). Let us consider the KB and the query of Example 2. If we associate random variables X_1 with axiom β_1, X_2 with β_2, and X_3 with β_3, the BDD representing the set of explanations is shown in Figure 1. By applying function PROBABILITY[44] to this BDD we get

$$\begin{aligned}
PROBABILITY(n_3) &= 0.7 \cdot 1 + 0.3 \cdot 0 = 0.7\\
PROBABILITY(n_2) &= 0.6 \cdot 1 + 0.4 \cdot 0.7 = 0.88\\
PROBABILITY(n_1) &= 0.2 \cdot 0.88 + 0.8 \cdot 0 = 0.176
\end{aligned}$$

and therefore $P(Q) = PROBABILITY(n_1) = 0.176$, which corresponds to the probability computed according to DISPONTE.

BUNDLE is characterized by a modular architecture,[31] shown in Figure 2, which allows inference:

1. from a *non-probabilistic* reasoner through implementations of the "Hitting Set algorithm",[45] which builds a Hitting Set Tree to incrementally obtain all the justifications. BUNDLE then builds a BDD for computing the probability of the query from the covering set of justifications. In this case, BUNDLE supports four OWL reasoners: Pellet

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

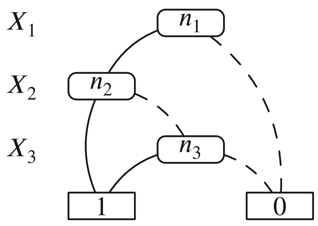


FIGURE 1 BDD representing the set of explanations for the query of Example 2.

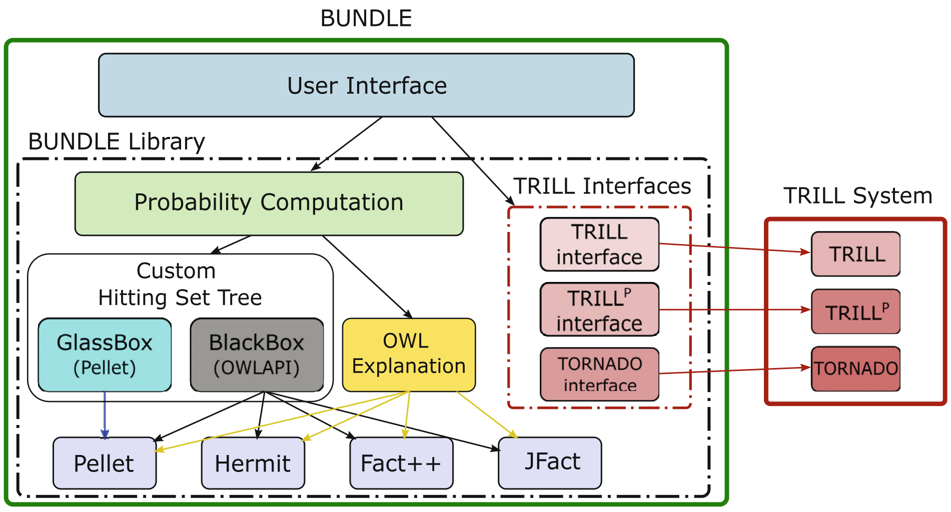


FIGURE 2 Software architecture of BUNDLE.

2.5.0[19] (written in Java), HermiT 1.3.8.413[21] (written in Java), Fact++ 1.6.5[20] (written in C++), and JFact 4.0.4[‖] (a Java port of Fact++). Three different strategies for finding a justification through the Hitting Set algorithm can be applied:

GlassBox A glass-box approach which depends on Pellet. It is a modified version of the `GlassBoxExplanation` class contained in the Pellet Explanation library.

BlackBox A black-box approach offered by the OWL API.[**][43] The OWL API is a Java API for the creation and manipulation of OWL 2 ontologies.

OWL Explanation A library that is part of the OWL Explanation Workbench.[43] The latter also contains a Protégé plugin, underpinned by the library, which allows users to find justifications for entailments in their OWL 2 ontologies.

All the supported non-probabilistic reasoners can be paired with the BlackBox or the OWL Explanation methods, while only Pellet can exploit the GlassBox method.

The GlassBox and the BlackBox methods implement a custom version of the Hitting Set algorithm, while OWL explanation implements the standard version.[45]

2. by directly querying the three *probabilistic* reasoners provided by the TRILL ("Tableau Reasoner for descrIption Logics in proLog") System: TRILL,[25,26] TRILL^P[27] and TORNADO,[28] written in Prolog in order to exploit Prolog's backtracking feature to perform inference on the KBs. The main differences between the three reasoners are that TRILL finds the set of justifications and builds the BDD from it, TRILL^P extracts a Boolean formula that represents the set of justifications and builds the BDD from it, while TORNADO directly builds a BDD that represents the set of

‖ http://jfact.sourceforge.net/.
** http://owlcs.github.io/owlapi/.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 Reasoners supported by the BUNDLE framework. The symbol ✓ means that the reasoner is compatible with the method used for computing the probability.

		Justification Finding				Pinpointing Formula
		Hitting Set Tree			Prolog backtracking	
Reasoner	**DL**	**GlassBox**	**BlackBox**	**OWL Expl.**	**built-in**	**built-in**
Pellet	$\mathcal{SROIQ}(\mathbf{D})$	✓	✓	✓		
HermiT	$\mathcal{SROIQ}(\mathbf{D})$		✓	✓		
Fact++	$\mathcal{SROIQ}(\mathbf{D})$		✓	✓		
JFact	$\mathcal{SROIQ}(\mathbf{D})$		✓	✓		
TRILL	$\mathcal{SHIQ}$				✓	
TRILL^P	$\mathcal{SHI}$					✓
TORNADO	$\mathcal{SHI}$					✓

justifications. When using one of the TRILL Interfaces, the Probability Computation module, which converts a covering set of justifications into a BDD, is not used. Note that BUNDLE can return both the probability of the query and the set of justifications, each one with the associated probability, if combined with TRILL, while it will return only the probability of the query if interfaced with TRILL^P and TORNADO (see also Table 1).

Table 1 provides an overview of the reasoners and methods supported by the BUNDLE framework.

The integration of both probabilistic and non-probabilistic reasoners in BUNDLE is made possible as DISPONTE exploits the "annotations", a type of OWL construct standardized in the language which allows for a smooth conversion of a certain KB into a probabilistic one, with minimal changes to the syntax. For instance, probabilistic axiom β_1 of Example 2 can be defined as:

```
<SubClassOf>
  <Annotation>
    <AnnotationProperty IRI="https://ml.unife.it/disponte#probability"/>
    <Literal datatypeIRI="http://www.w3.org/2000/01/rdf-schema#Literal">0.2</Literal>
  </Annotation>
  <Class IRI="#Nihilist"/>
  <Class IRI="#GreatMan"/>
</SubClassOf>
```

Note that the IRI to add the DISPONTE annotation is https://ml.unife.it/disponte#probability.

In this way, every DISPONTE KB can also be queried by a non-probabilistic reasoner without any change. Clearly, probabilistic inference is this case will not be possible.

More details about the use of the DISPONTE annotation can be found at https://ml.unife.it/disponte.

Thanks to its modularity, BUNDLE can be easily extended in three main ways:

- By adding a non-probabilistic reasoner that implements the `OWLReasoner` interface of the OWL API library;
- By adding a non-probabilistic reasoner that is able to perform the reasoning task of justification finding. This reasoner should implement the `ExplanationReasoner` interface defined in BUNDLE;
- By adding a probabilistic reasoner that implements the `ProbabilisticReasoner` interface defined in BUNDLE, as done for the TRILL System.

Up to now BUNDLE was available as a standalone desktop application or as a library (see the work presenting the BUNDLE's modular architecture[31] for technical details). In the next section we are going to present a Web application, which allows the user to test the framework without installing any software on the local machine.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

5 | BUNDLE WEB APPLICATION

5.1 | Design

The application design is based on the Spring Framework, which provides a comprehensive programming and configuration model for modern Java-based applications and allows to focus on application-level business logic, without unnecessary ties to specific deployment environments. In particular, Spring Boot and Spring MVC were employed in order to respectively embed the web server and configure its execution (the former), and for structuring the Web application according to the model-view-controller pattern (the latter).

5.2 | User interface

The Web application is composed of three main views, all accessible from the menu bar placed at the top of the page:

Home the main view, where a user lands when navigating to the URL https://bundle.ml.unife.it/ (Figure 3);

Examples a static page containing the description of both toy and real-world ontologies included in the Web application and testable by clicking on the correspondent button (Figure 4);

Tutorial a static page, showing how to use the Web application.

The **Home** page is divided in two panels. On the left side users can write or upload an ontology. On the right side, users can select several queries, set the corresponding parameters, and visualize the results. In the following

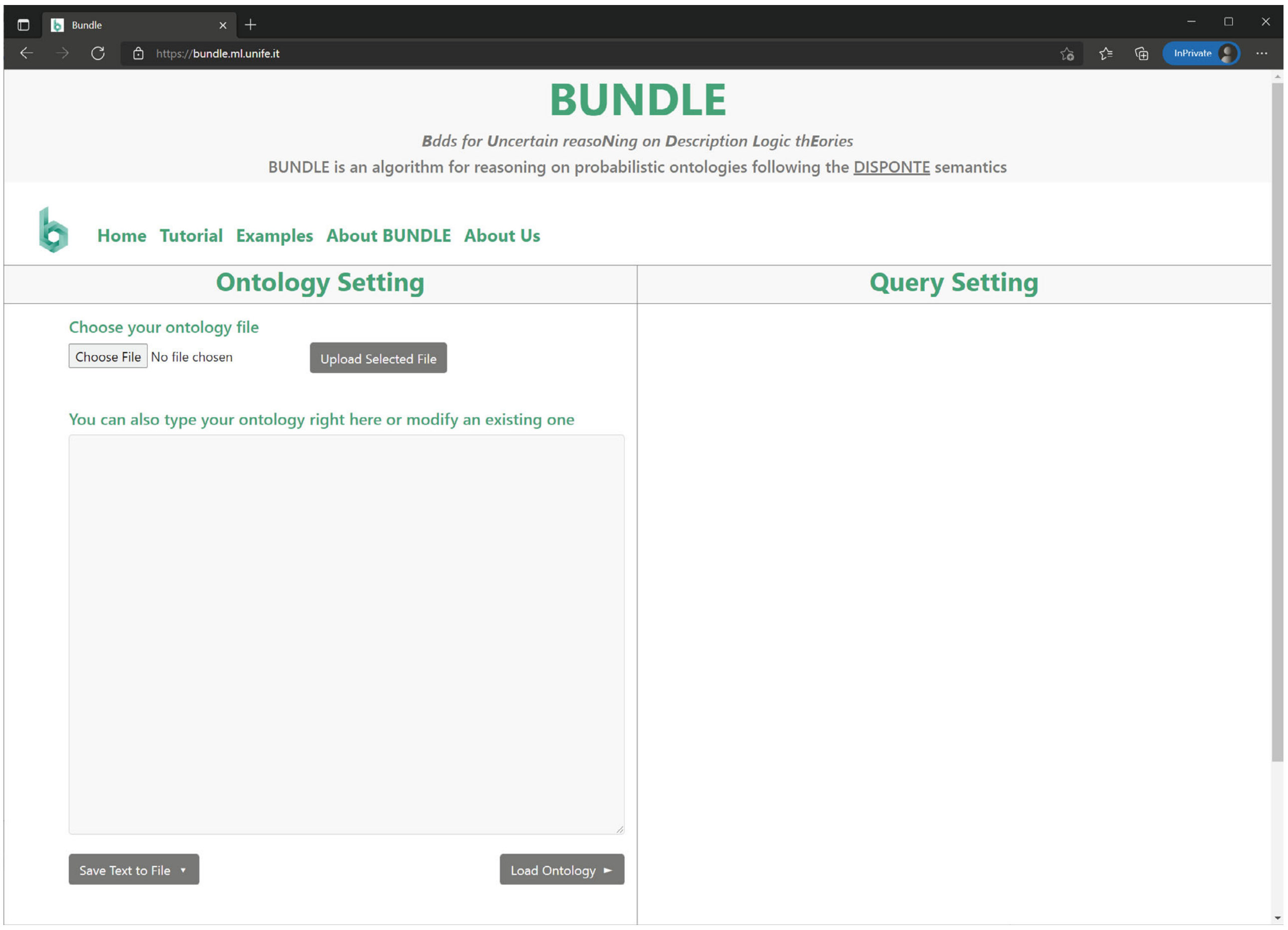
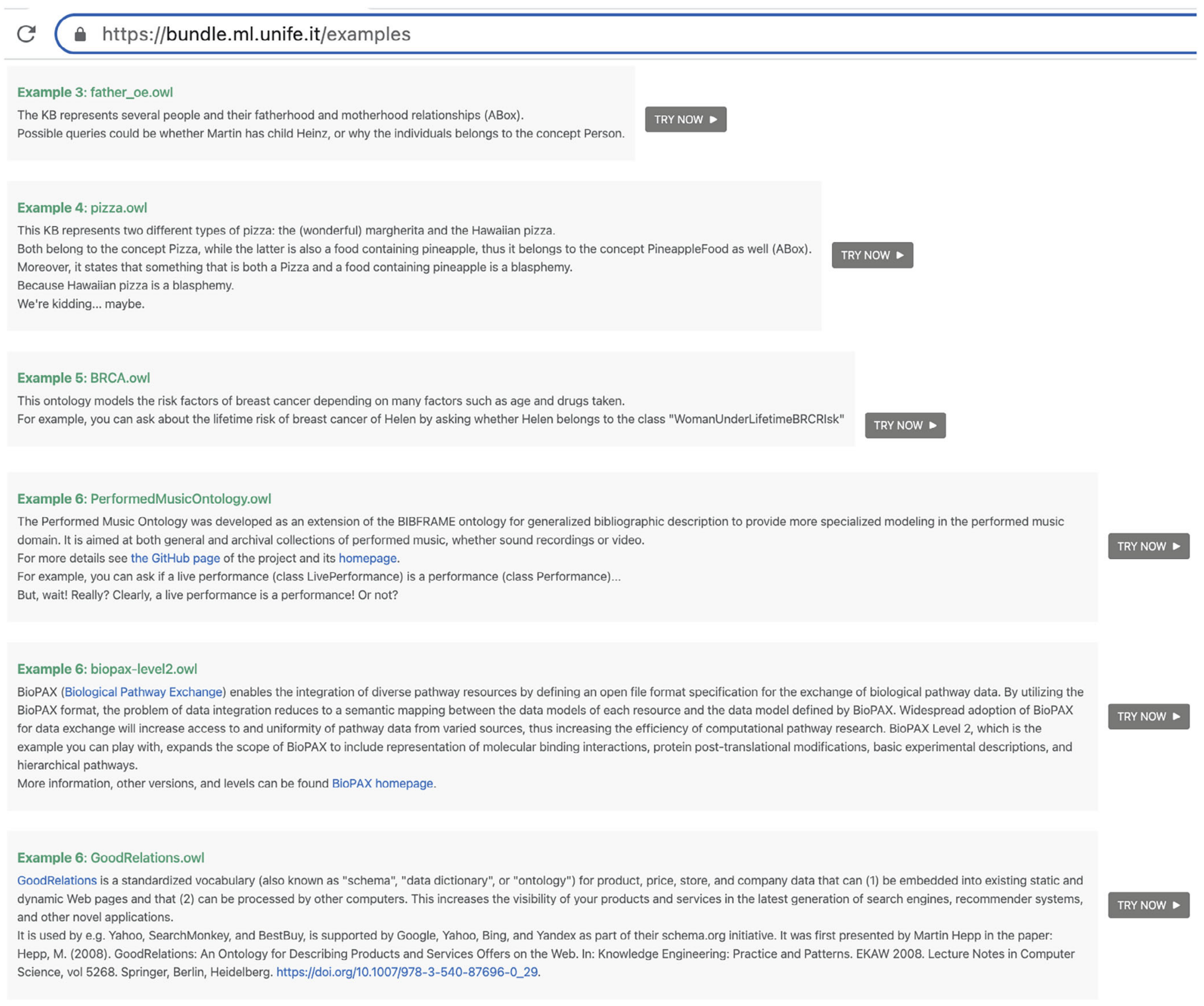

Bundle
https://bundle.ml.unife.it
InPrivate
BUNDLE
Bdds for Uncertain reasoNing on Description Logic thEories
BUNDLE is an algorithm for reasoning on probabilistic ontologies following the DISPONTE semantics
Home
Tutorial
Examples
About BUNDLE
About Us
Ontology Setting
Query Setting
Choose your ontology file
Choose File
No file chosen
Upload Selected File
You can also type your ontology right here or modify an existing one
Save Text to File
Load Ontology

FIGURE 3 Home Page.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License


https://bundle.ml.unife.it/examples

Example 3: father_oe.owl
The KB represents several people and their fatherhood and motherhood relationships (ABox).
Possible queries could be whether Martin has child Heinz, or why the individuals belongs to the concept Person.

Example 4: pizza.owl
This KB represents two different types of pizza: the (wonderful) margherita and the Hawaiian pizza.
Both belong to the concept Pizza, while the latter is also a food containing pineapple, thus it belongs to the concept PineappleFood as well (ABox).
Moreover, it states that something that is both a Pizza and a food containing pineapple is a blasphemy.
Because Hawaiian pizza is a blasphemy.
We're kidding... maybe.

Example 5: BRCA.owl
This ontology models the risk factors of breast cancer depending on many factors such as age and drugs taken.
For example, you can ask about the lifetime risk of breast cancer of Helen by asking whether Helen belongs to the class "WomanUnderLifetimeBRCRIsk"

Example 6: PerformedMusicOntology.owl
The Performed Music Ontology was developed as an extension of the BIBFRAME ontology for generalized bibliographic description to provide more specialized modeling in the performed music domain. It is aimed at both general and archival collections of performed music, whether sound recordings or video.
For more details see the GitHub page of the project and its homepage.
For example, you can ask if a live performance (class LivePerformance) is a performance (class Performance)...
But, wait! Really? Clearly, a live performance is a performance! Or not?

Example 6: biopax-level2.owl
BioPAX (Biological Pathway Exchange) enables the integration of diverse pathway resources by defining an open file format specification for the exchange of biological pathway data. By utilizing the BioPAX format, the problem of data integration reduces to a semantic mapping between the data models of each resource and the data model defined by BioPAX. Widespread adoption of BioPAX for data exchange will increase access to and uniformity of pathway data from varied sources, thus increasing the efficiency of computational pathway research. BioPAX Level 2, which is the example you can play with, expands the scope of BioPAX to include representation of molecular binding interactions, protein post-translational modifications, basic experimental descriptions, and hierarchical pathways.
More information, other versions, and levels can be found BioPAX homepage.

Example 6: GoodRelations.owl
GoodRelations is a standardized vocabulary (also known as "schema", "data dictionary", or "ontology") for product, price, store, and company data that can (1) be embedded into existing static and dynamic Web pages and that (2) can be processed by other computers. This increases the visibility of your products and services in the latest generation of search engines, recommender systems, and other novel applications.
It is used by e.g. Yahoo, SearchMonkey, and BestBuy, is supported by Google, Yahoo, Bing, and Yandex as part of their schema.org initiative. It was first presented by Martin Hepp in the paper:
Hepp, M. (2008). GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In: Knowledge Engineering: Practice and Patterns. EKAW 2008. Lecture Notes in Computer Science, vol 5268. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-87696-0_29.

FIGURE 4 Examples page containing some predefined ontologies.

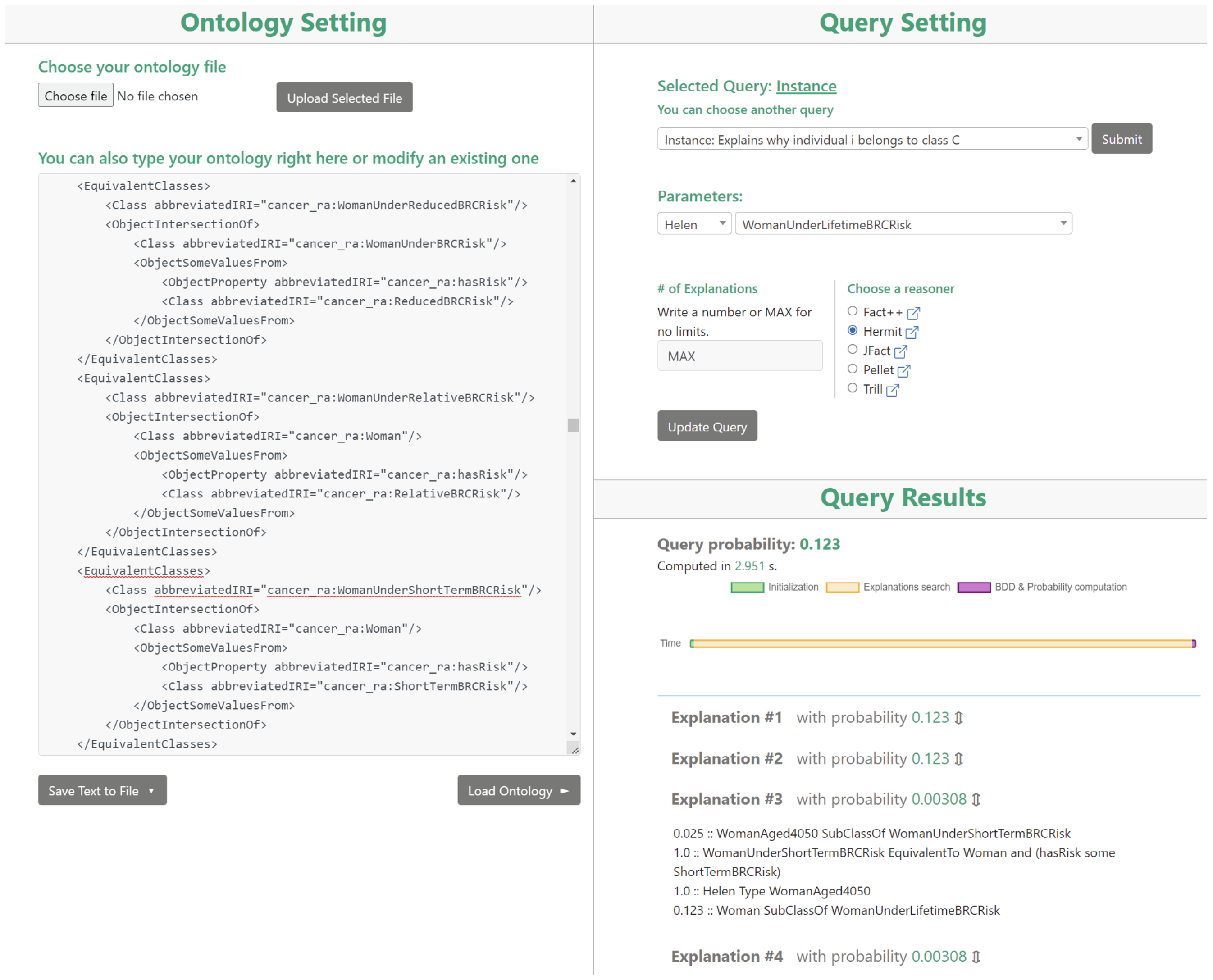
we illustrate how the Web application works, both the front-end and the back-end, by referring to the execution of Example 2.

A DISPONTE KB can be loaded (Figure 5 left side "Ontology Setting") in three different ways: by (1) uploading an OWL file, (2) writing from scratch the KB in the text area on the left side, or (3) loading a predefined ontology from the **Examples** page by pressing the button *Try now* (Figure 4). Note that, when a KB is uploaded or chosen from the examples, its content is shown in the text area, where it can be eventually modified and saved as an `.owl` file. Then users can load the KB into BUNDLE, by pressing the button *Load Ontology*. At this point, the KB will be sent to the server, as shown in step 1 in Figure 6. Here, the back-end initializes a data structure containing the current session ID, reads the KB, and instantiates an `OWLOntology` object (which is saved into the `session` data structure), and extracts all the individuals, classes, and properties to populate the drop-down menus in the right panel.

In the right panel (Figure 5, "Query Setting") the user is asked to choose the query type from a first drop-down menu (step 2, Figure 6). The available query types are:

- Subclass: compute the probability of class A to be a subclass of B explaining why ($Q = A \sqsubseteq B$)
- Instance: compute the probability of individual i to belong to class C explaining why ($Q = i : C$)
- ObjectProperty: compute the probability of individuals s and o to be linked by property P explaining why ($Q = (s, o) : P$)

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License



```
<EquivalentClasses>
    <Class abbreviatedIRI="cancer_ra:WomanUnderReducedBRCRisk"/>
    <ObjectIntersectionOf>
        <Class abbreviatedIRI="cancer_ra:WomanUnderBRCRisk"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty abbreviatedIRI="cancer_ra:hasRisk"/>
            <Class abbreviatedIRI="cancer_ra:ReducedBRCRisk"/>
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
<EquivalentClasses>
    <Class abbreviatedIRI="cancer_ra:WomanUnderRelativeBRCRisk"/>
    <ObjectIntersectionOf>
        <Class abbreviatedIRI="cancer_ra:Woman"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty abbreviatedIRI="cancer_ra:hasRisk"/>
            <Class abbreviatedIRI="cancer_ra:RelativeBRCRisk"/>
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
<EquivalentClasses>
    <Class abbreviatedIRI="cancer_ra:WomanUnderShortTermBRCRisk"/>
    <ObjectIntersectionOf>
        <Class abbreviatedIRI="cancer_ra:Woman"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty abbreviatedIRI="cancer_ra:hasRisk"/>
            <Class abbreviatedIRI="cancer_ra:ShortTermBRCRisk"/>
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
```

Selected Query: Instance

You can choose another query

Instance: Explains why individual i belongs to class C — Submit

Parameters: Helen — WomanUnderLifetimeBRCRisk

# of Explanations: Write a number or MAX for no limits. MAX

Choose a reasoner: Fact++ [✓ Hermit] JFact Pellet Trill

Update Query

Query probability: 0.123

Computed in 2.951 s.

🟩 Initialization 🟨 Explanations search 🟪 BDD & Probability computation

Time

Explanation #1 with probability 0.123

Explanation #2 with probability 0.123

Explanation #3 with probability 0.00308

0.025 :: WomanAged4050 SubClassOf WomanUnderShortTermBRCRisk
1.0 :: WomanUnderShortTermBRCRisk EquivalentTo Woman and (hasRisk some ShortTermBRCRisk)
1.0 :: Helen Type WomanAged4050
0.123 :: Woman SubClassOf WomanUnderLifetimeBRCRisk

Explanation #4 with probability 0.00308

FIGURE 5 Home page showing the results of the instance query Q = Helen:WomanUnderLifetimeBRCRisk to the "BRCA" ontology with the HermiT reasoner.

- DataProperty: compute the probability of individual i to have value v for property P explaining why $(Q = (i, v) : P)$
- AllUnsat: looks for unsatisfiable classes and explains why
- Unsat: compute the probability of class C to be unsatisfiable explaining why
- Inconsistent: compute the probability of the ontology to be inconsistent explaining why

Depending on the query type, a set of consequent drop-down menus will pop up below, after pressing the *Submit* button, for setting the query parameters (step 3, Figure 6). For example, by selecting the "Instance" query as shown in Figure 5 on the right, two drop-down menus for parameter setting will be shown, one with all the individuals and one with all the concepts in the ontology.

Finally, it is possible to set the maximum number of justifications to find and select one of the reasoners integrated into BUNDLE. By now, HermiT, Pellet, JFact, Fact++, and TRILL are available. TRILL^P and TORNADO are not available because they do not return a set of justifications.

By hitting the *Execute Query* button, the arguments of the query are collected and sent to the server, where the selected reasoner will be initialized and the query performed. The collected justifications are sent to the client and shown to the user, together with the execution time (in seconds), the query probability, the probability of each justification and the probability of each axiom used in the justifications, as shown in the bottom right panel "Query Results" in Figure 5. Together with the total inference time, the time taken in the different reasoning steps is shown by means of a color bar.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

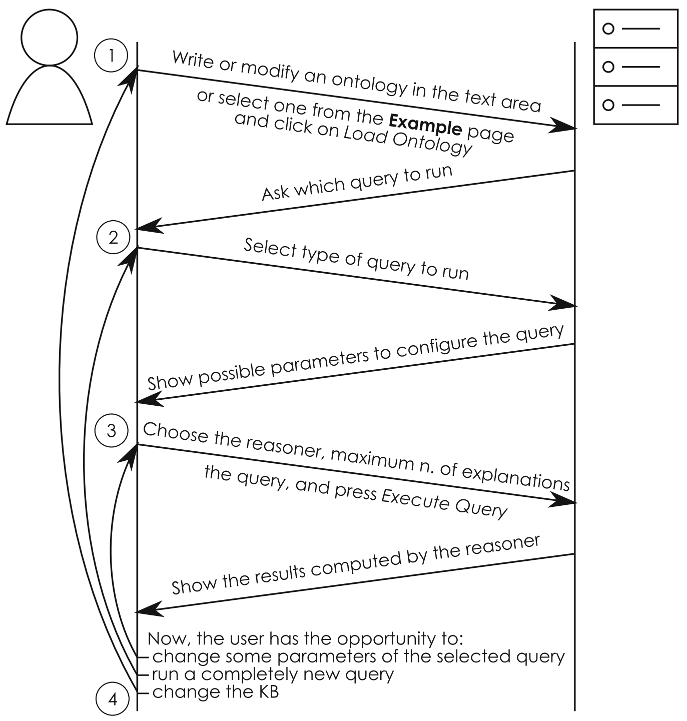

1
Write or modify an ontology in the text area
or select one from the **Example** page
and click on *Load Ontology*
Ask which query to run
2
Select type of query to run
Show possible parameters to configure the query
3
Choose the reasoner, maximum n. of explanations
the query, and press *Execute Query*
Show the results computed by the reasoner
Now, the user has the opportunity to:
- change some parameters of the selected query
- run a completely new query
4
- change the KB

FIGURE 6 The client-server interactions to solve a query on the BUNDLE Web application.

Each color represents a step (in order): initialization of the reasoner (green), search for justifications (yellow), construction of the BDD, and computation of the probability of the query (purple). To see how much time each step took, the user can hover the segment with the mouse pointer or touch it in case a touch screen is used. When running TRILL, the two segments for the justification search and probability computation are replaced by a single segment considering both operations. This is due to the fact that TRILL builds the BDD and computes the probability internally.

Now the user can (step 4, Figure 6): (1) change the parameters of the query, (2) change the query type, (3) modify the KB in the text area and re-load it, or change the KB by uploading a new one. All the computation is done server-side, in this way it is possible to change the parameters of the queries without reloading the KB at each operation.

The entire workflow of the interaction between client and server is shown in Figure 6.

The application applies a timeout on the execution of the query. This timeout was set to 180 s, at the end of which BUNDLE shows the justifications found so far and computes the probability of the query based on these justifications. When the timeout is reached, a message warns the user about the fact that not all the justifications may be shown and that the probability may be a lower bound of the effective probability (Figure 7). The motivation behind the use of a timeout is that BUNDLE is a testing tool useful for developing and experimenting, but it is not suitable for heavy computations, for which a local installation should be used.[31]

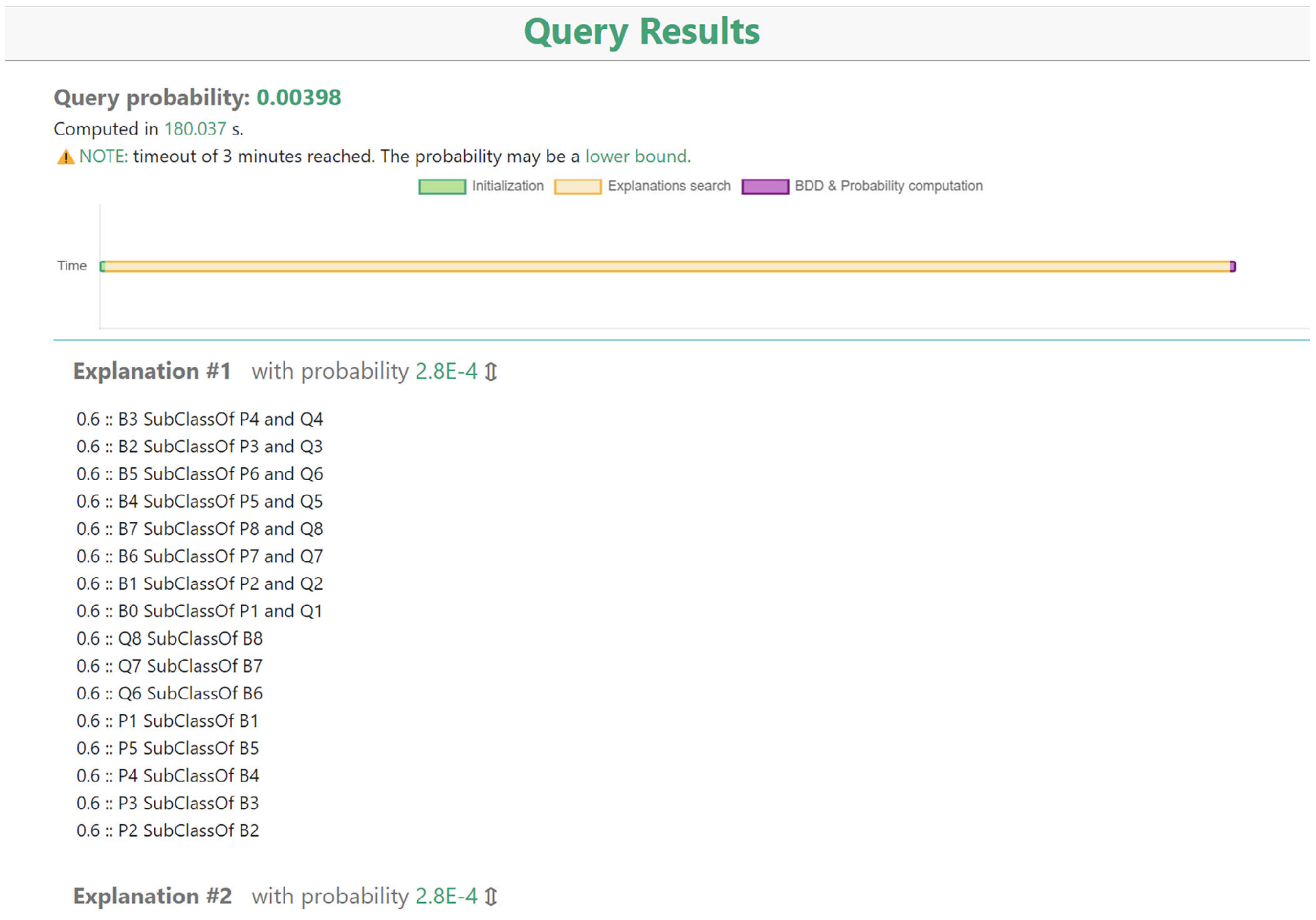
However, the application is extremely useful to perform a comparison between the embedded reasoners. For example, if we consider the following probabilistic KB for $1 \leq i \leq n$, where $n \geq 1$:

$$C_{1,i} = 0.6 :: B_{i-1} \sqsubseteq P_i \sqcap Q_i, \quad C_{2,i} = 0.6 :: P_i \sqsubseteq B_i, \qquad C_{3,i} = 0.6 :: Q_i \sqsubseteq B_i,$$

the query $Q = B_0 \sqsubseteq B_n$ has 2^n justifications, even if the KB has a size that is linear in n. For $n = 2$ for example, there are four different justifications:

$$\{C_{1,1}, C_{2,1}, C_{1,2}, C_{2,2}\},$$
$$\{C_{1,1}, C_{3,1}, C_{1,2}, C_{2,2}\},$$
$$\{C_{1,1}, C_{2,1}, C_{1,2}, C_{3,2}\},$$
$$\{C_{1,1}, C_{3,1}, C_{1,2}, C_{3,2}\},$$

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License


Query Results

Query probability: 0.00398

Computed in 180.037 s.

⚠ NOTE: timeout of 3 minutes reached. The probability may be a lower bound.

Explanation #1 with probability 2.8E-4

0.6 :: B3 SubClassOf P4 and Q4
0.6 :: B2 SubClassOf P3 and Q3
0.6 :: B5 SubClassOf P6 and Q6
0.6 :: B4 SubClassOf P5 and Q5
0.6 :: B7 SubClassOf P8 and Q8
0.6 :: B6 SubClassOf P7 and Q7
0.6 :: B1 SubClassOf P2 and Q2
0.6 :: B0 SubClassOf P1 and Q1
0.6 :: Q8 SubClassOf B8
0.6 :: Q7 SubClassOf B7
0.6 :: Q6 SubClassOf B6
0.6 :: P1 SubClassOf B1
0.6 :: P5 SubClassOf B5
0.6 :: P4 SubClassOf B4
0.6 :: P3 SubClassOf B3
0.6 :: P2 SubClassOf B2

Explanation #2 with probability 2.8E-4

FIGURE 7 The warning message shown by the application in case the timeout is reached. Results are relative to the example KB with an exponential number of justifications.

TABLE 2 Average running time in seconds for each KB's size for the reasoners Fact++, HermiT, JFact, Pellet, and TRILL to answer the query $Q = B_0 \sqsubseteq B_n$.

	Avg. running time (s) per size			
Reasoner	**2**	**4**	**6**	**8**
Fact++	34.587	1.045	3.819	17.370
HermiT	1.876	10.417	57.614	TO
JFact	0.417	1.005	3.414	13.669
Pellet	0.338	0.949	3.468	14.177
TRILL	0.118	0.334	1.230	8.851

Note: TO stands for timeout, that is, the reasoner was not able to collect all justifications within the 180 s timeout.

where the $C_{i,j}$s stand for the atomic choices $(C_{i,j}, 1)$, while with $n = 8$ there are 256 justifications. Table 2 shows the average running times to solve the query Q with the different reasoners with $n = 2, 4, 6, 8$. Times are averaged over 5 runs for each reasoner for each size of the KB.

As shown in Table 2, Fact++, HermiT, JFact, Pellet, and TRILL can find all the justifications for the KBs with n equals to 2, 4, and 6 (with 4, 16, and 64 justifications respectively and query probability of 0.25402, 0.06452, and 0.01639 respectively). On the other hand, with $n = 8$ HermiT could not find all the 256 justifications within the given time limit of 180 s. In this case, while the exact probability $P(Q) = 0.00416$, HermiT computes the lower bound $P(Q) = 0.00398$ from

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

```
...
<owl:Class rdf:about="&cancer_ra;LackOfExercise">
<rdfs:subClassOf rdf:resource="&cancer_ra;KnownFactor"/>
</owl:Class>

<owl:Class rdf:about="&cancer_ra;LateFirstChild">
<rdfs:subClassOf rdf:resource="&cancer_ra;KnownFactor"/>
</owl:Class>
...
```

FIGURE 8 Axioms from BRCA. They state that the lack of exercise may increase the risk of having breast cancer as well as having the first child at old age.

147 justifications (HermiT was able to find 147 justifications on average in 180 s). Each justification has a probability of $2.8e^{-4}$, as shown in Figure 7.

5.3 | Examples

In this section more examples of queries available in the application are presented:

- Probabilistic reasoning with a medical ontology, where the subjective probabilities representing risk factors for a certain individual can be combined with objective probabilities representing the statistical knowledge. This ontology is the "Breast Cancer Risk Assessment" (BRCA)[††] whose original version was presented by Klinov and Parsia:[46] it states the risk factors of breast cancer depending on several factors such as age, drugs taken, ethnicity and so forth. It is a very challenging ontology to reason with and the creators state[46] "that reasoners that can handle this ontology will work for an interesting range of applications". BRCA has $\mathcal{ALCHF}(\mathbf{D})$ expressiveness and contains 491 axioms, 154 different classes, and 15 different properties (12 object properties and 3 data properties). The development of the ontology was driven by the idea that risk assessment were reduced to probabilistic entailment by introducing probabilistic axioms, defining for example that women aged 20–30 are unlikely to develop the disease in the short term:

$$0.004 :: \text{WomanAged2030} \sqsubseteq \text{WomanUnderShortTermBRCRisk}.$$

Figure 8 shows that known factors for the disease are lack of exercise or having the first child at a late age. They can be encoded using the following axioms:

$$\begin{gathered}\text{LackOfExercise} \sqsubseteq \text{KnownFactor},\\ \text{LateFirstChild} \sqsubseteq \text{KnownFactor}.\end{gathered}$$

In the version available in the Web application, we added an individual, "Helen", aged 40–50. If we query the KB to know the risk of developing breast cancer in her lifetime (`Q=Helen:WomanUnderLifetimeBRCRisk`) we obtain a probability value of 0.123 with 4 different explanations, as shown in Figure 5, where the reasoner HermiT was used. Instead, if we query to know Helen's short term risk (`Q=Helen:WomanUnderShortTermBRCRisk`) we obtain the following explanation with probability 0.025, telling us that a woman aged 40–50 has an overall low risk to develop breast cancer[‡‡]:

$$\begin{gathered}0.025 :: \text{WomanAged4050} \sqsubseteq \text{WomanUnderShortTermBRCRisk}\\ \text{Helen} : \text{WomanAged4050}.\end{gathered}$$

[††]The KB can be found in the examples page at https://bundle.ml.unife.it/examples.

[‡‡]The probability values are not updated in the medical literature. The values shown in this example are used to discuss the KB and a possible use case.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Figure 9 shows the results obtained by the Web application.

- the "Inconsistent" and "Unsat" (unsatisfiable) query types.

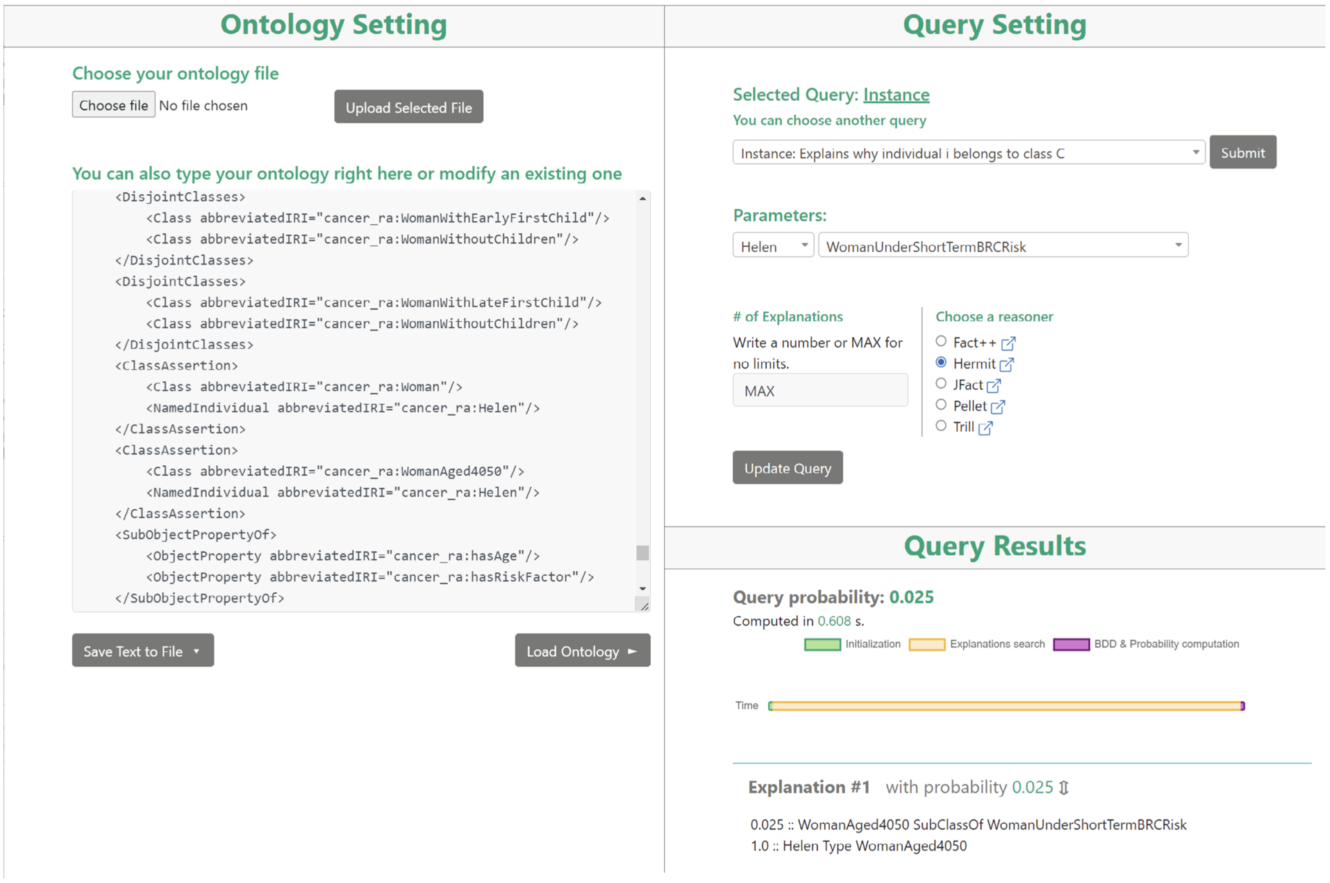
  A KB is *inconsistent* if it contains contradictory information, so no parameters are required for running this kind of query. Consider the KB of Example 2: if we add the following axioms

$$\begin{aligned}&\exists \mathrm{killed}.\top \equiv \mathrm{Killer},\\ &\mathrm{Killer}\ \mathtt{disjointWith}\ \mathrm{GreatMan},\end{aligned}$$

we will obtain that `raskolnikov` is both a killer and a great man, leading to a contradiction. As shown in Figure 10, a possible justification is, indeed:

$$\begin{aligned}0.7 ::\ &(\mathrm{raskolnikov}, \mathrm{lizaveta}): \mathrm{killed},\\ &\exists \mathrm{killed}.\top \equiv \mathrm{Killer},\\ &\exists \mathrm{killed}.\top \sqsubseteq \mathrm{Nihilist},\\ 0.2 ::\ &\mathrm{Nihilist} \sqsubseteq \mathrm{GreatMan},\\ &\mathrm{Killer}\ \mathtt{disjointWith}\ \mathrm{GreatMan}.\end{aligned}$$

A class C is *unsatisfiable* when the presence of at least one individual in class C forces any model of the ontology to contain a logical contradiction. In the previous example, the class Killer is unsatisfiable. If we remove from the KB all the individuals (raskolnikov, lizaveta, alyona) or just the individual belonging to Killer the resulting KB is consistent, because it does not contain any contradiction. With BUNDLE, it is possible to get the justifications explaining why a class (or more classes) is unsatisfiable, as shown in Figure 11. In this case, the justification



Ontology Setting

Choose your ontology file

Choose file No file chosen Upload Selected File

You can also type your ontology right here or modify an existing one

```
    <DisjointClasses>
        <Class abbreviatedIRI="cancer_ra:WomanWithEarlyFirstChild"/>
        <Class abbreviatedIRI="cancer_ra:WomanWithoutChildren"/>
    </DisjointClasses>
    <DisjointClasses>
        <Class abbreviatedIRI="cancer_ra:WomanWithLateFirstChild"/>
        <Class abbreviatedIRI="cancer_ra:WomanWithoutChildren"/>
    </DisjointClasses>
    <ClassAssertion>
        <Class abbreviatedIRI="cancer_ra:Woman"/>
        <NamedIndividual abbreviatedIRI="cancer_ra:Helen"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class abbreviatedIRI="cancer_ra:WomanAged4050"/>
        <NamedIndividual abbreviatedIRI="cancer_ra:Helen"/>
    </ClassAssertion>
    <SubObjectPropertyOf>
        <ObjectProperty abbreviatedIRI="cancer_ra:hasAge"/>
        <ObjectProperty abbreviatedIRI="cancer_ra:hasRiskFactor"/>
    </SubObjectPropertyOf>
```

Save Text to File ▾ Load Ontology ►

Query Setting

Selected Query: Instance

You can choose another query

Instance: Explains why individual i belongs to class C ▾ Submit

Parameters:

Helen ▾ WomanUnderShortTermBRCRisk ▾

# of Explanations

Write a number or MAX for no limits.

MAX

Choose a reasoner

○ Fact++ ● Hermit ○ JFact ○ Pellet ○ Trill

Update Query

Query Results

Query probability: 0.025

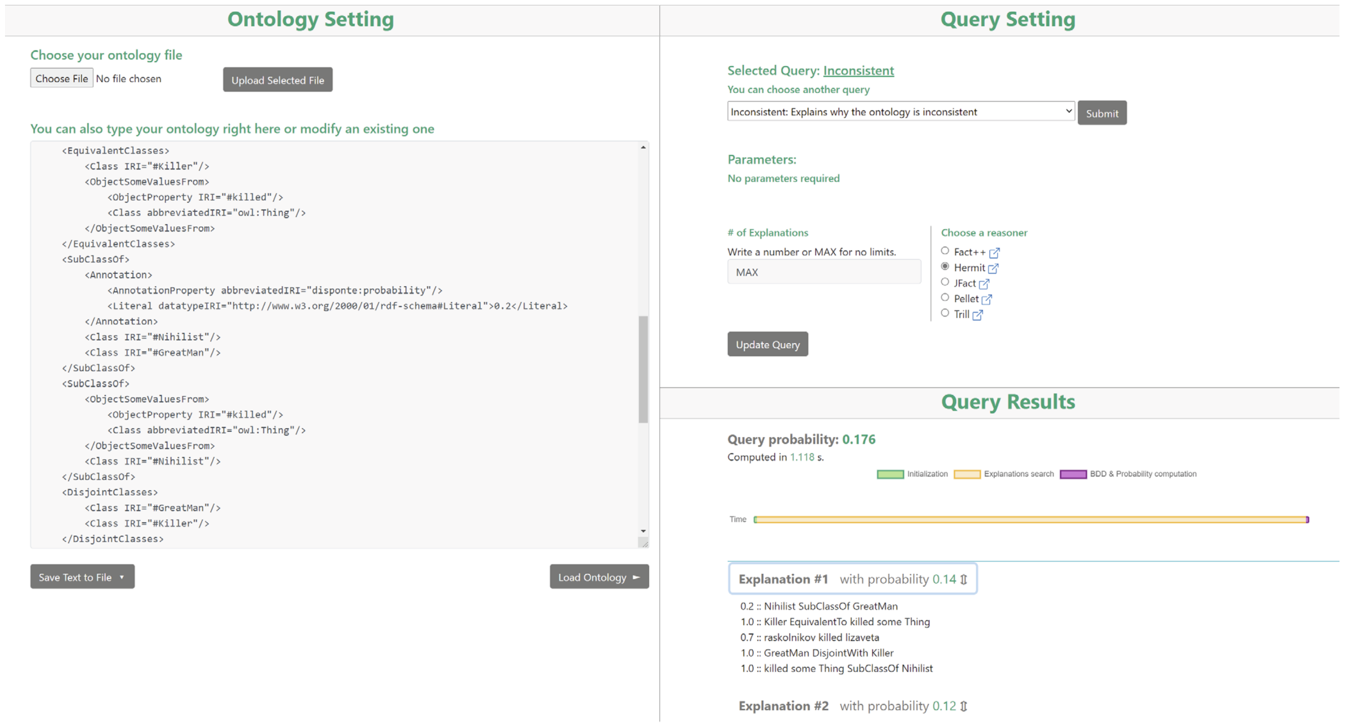
Computed in 0.608 s.

Explanation #1 with probability 0.025

0.025 :: WomanAged4050 SubClassOf WomanUnderShortTermBRCRisk
1.0 :: Helen Type WomanAged4050

FIGURE 9 BUNDLE used to find justifications for the query $Q = \mathrm{Helen} : \mathrm{WomanUnderShortTermBRCRisk}$ w.r.t. the BRCA KB.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License



Ontology Setting

Choose your ontology file

Choose File No file chosen Upload Selected File

You can also type your ontology right here or modify an existing one

```
    <EquivalentClasses>
        <Class IRI="#Killer"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#killed"/>
            <Class abbreviatedIRI="owl:Thing"/>
        </ObjectSomeValuesFrom>
    </EquivalentClasses>
    <SubClassOf>
        <Annotation>
            <AnnotationProperty abbreviatedIRI="disponte:probability"/>
            <Literal datatypeIRI="http://www.w3.org/2000/01/rdf-schema#Literal">0.2</Literal>
        </Annotation>
        <Class IRI="#Nihilist"/>
        <Class IRI="#GreatMan"/>
    </SubClassOf>
    <SubClassOf>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#killed"/>
            <Class abbreviatedIRI="owl:Thing"/>
        </ObjectSomeValuesFrom>
        <Class IRI="#Nihilist"/>
    </SubClassOf>
    <DisjointClasses>
        <Class IRI="#GreatMan"/>
        <Class IRI="#Killer"/>
    </DisjointClasses>
```

Save Text to File ▾ Load Ontology ►

Query Setting

Selected Query: Inconsistent

You can choose another query

Inconsistent: Explains why the ontology is inconsistent Submit

Parameters:

No parameters required

# of Explanations

Write a number or MAX for no limits.

MAX

Choose a reasoner

○ Fact++ [● Hermit] ○ JFact ○ Pellet ○ Trill

Update Query

Query Results

Query probability: 0.176

Computed in 1.118 s.

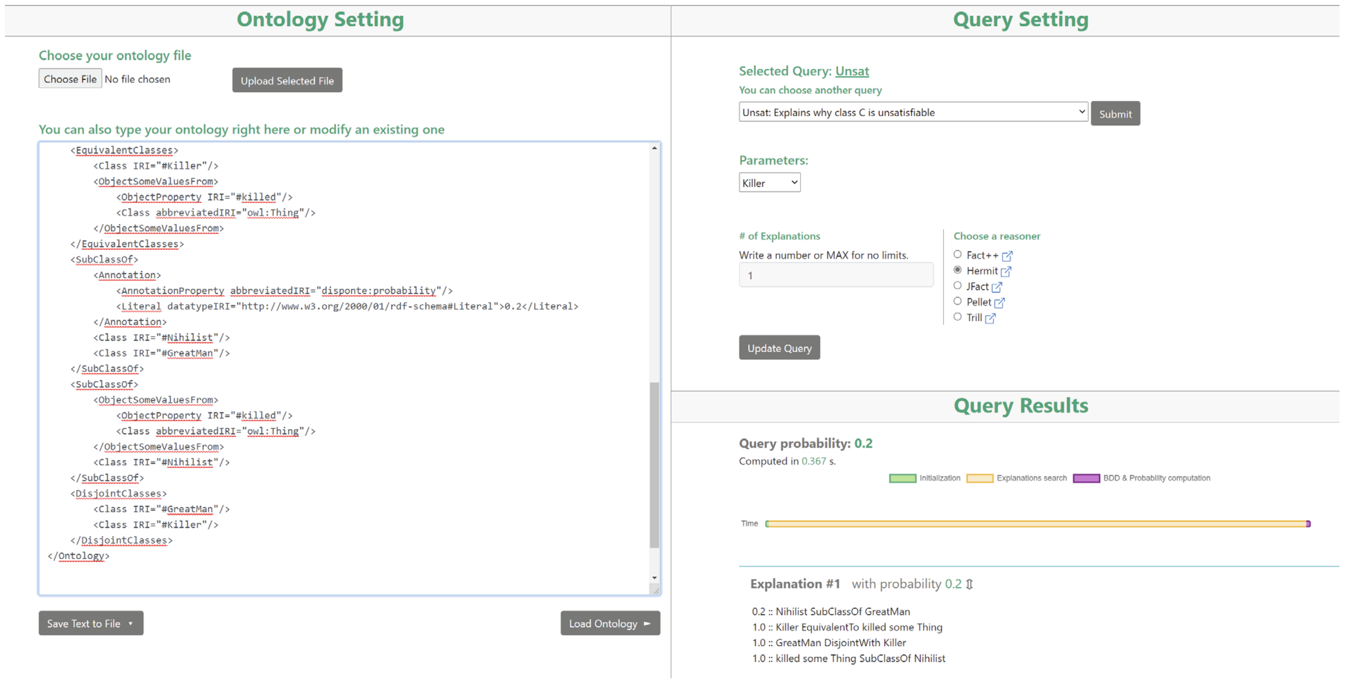
🟩 Initialization 🟨 Explanations search 🟪 BDD & Probability computation

Explanation #1 with probability 0.14

0.2 :: Nihilist SubClassOf GreatMan
1.0 :: Killer EquivalentTo killed some Thing
0.7 :: raskolnikov killed lizaveta
1.0 :: GreatMan DisjointWith Killer
1.0 :: killed some Thing SubClassOf Nihilist

Explanation #2 with probability 0.12

FIGURE 10 BUNDLE used to find justifications about the inconsistency of a KB.



Ontology Setting

Choose your ontology file

Choose File No file chosen Upload Selected File

You can also type your ontology right here or modify an existing one

```
    <EquivalentClasses>
        <Class IRI="#Killer"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#killed"/>
            <Class abbreviatedIRI="owl:Thing"/>
        </ObjectSomeValuesFrom>
    </EquivalentClasses>
    <SubClassOf>
        <Annotation>
            <AnnotationProperty abbreviatedIRI="disponte:probability"/>
            <Literal datatypeIRI="http://www.w3.org/2000/01/rdf-schema#Literal">0.2</Literal>
        </Annotation>
        <Class IRI="#Nihilist"/>
        <Class IRI="#GreatMan"/>
    </SubClassOf>
    <SubClassOf>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#killed"/>
            <Class abbreviatedIRI="owl:Thing"/>
        </ObjectSomeValuesFrom>
        <Class IRI="#Nihilist"/>
    </SubClassOf>
    <DisjointClasses>
        <Class IRI="#GreatMan"/>
        <Class IRI="#Killer"/>
    </DisjointClasses>
</Ontology>
```

Save Text to File ▾ Load Ontology ►

Query Setting

Selected Query: Unsat

You can choose another query

Unsat: Explains why class C is unsatisfiable Submit

Parameters:

Killer

# of Explanations

Write a number or MAX for no limits.

1

Choose a reasoner

○ Fact++ [● Hermit] ○ JFact ○ Pellet ○ Trill

Update Query

Query Results

Query probability: 0.2

Computed in 0.367 s.

🟩 Initialization 🟨 Explanations search 🟪 BDD & Probability computation

Explanation #1 with probability 0.2

0.2 :: Nihilist SubClassOf GreatMan
1.0 :: Killer EquivalentTo killed some Thing
1.0 :: GreatMan DisjointWith Killer
1.0 :: killed some Thing SubClassOf Nihilist

FIGURE 11 BUNDLE used to find the justification about the unsatisfiability of class Killer.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

returned is:

$$\begin{aligned}
&\exists \text{killed}.\top \equiv \text{Killer},\\
&\exists \text{killed}.\top \sqsubseteq \text{Nihilist},\\
0.2 :: \;&\text{Nihilist} \sqsubseteq \text{GreatMan},\\
&\text{Killer } \texttt{disjointWith} \text{ GreatMan}.
\end{aligned}$$

Note that in this KB the class Nihilist is satisfiable because there may be individuals belonging to it that did not kill anyone. This is because of the use of subsumption, which tells that if an individual killed someone, she/he belongs to the class Nihilist but the vice versa is not always true (there can be individuals belonging to Nihilist that did not kill anyone). On the other hand, a Killer belongs also to class $\exists \text{killed}.\top$ and vice versa because of the equality of the two concepts. Differently, if an individual is a killer, it is certainly a nihilist, leading to a contradiction as seen before.

6 | VALIDATION OF BUNDLE WEB APPLICATION

To validate the effectiveness and practical impact of the application we prepared a questionnaire to be administered to the users of the application. The questionnaire was divided in two sections, one about usability and one targeted to the specific features of the application. The link to the questionnaire and to the application was published on several international mailing lists, both related to Semantic Web or, more generally, to artificial intelligence. In this way, we ensured we could reach interested researchers who could give more appropriate and accurate answers. We collected 13 answers, despite the number of accesses to the application was 77 in the same period of time. Knowing in advance the difficulty to collect responses for such questionnaires, the first section of the questionnaire was prepared based on the Post-Study System Usability Questionnaire (PSSUQ) Version 3 template,[36,37] also called Computer System Usability Questionnaire (CSUQ). PSSUQ has a high reliability,[47] even with a small number of participants: for example, Tullis and Stetson[48] found that a sample size of 12 answers generated the same results as a larger sample size 90% of the time.

PSSUQ Version 3 consists of 16 questions following a 7-point Likert Scale, that is, every question has 7 options to choose from on a scale between Strongly Agree (score 1) to Strongly Disagree (score 7):

1. Overall, I am satisfied with how easy it is to use this system.
2. It was simple to use this system.
3. I was able to complete the tasks and scenarios quickly using this system.
4. I felt comfortable using this system.
5. It was easy to learn to use this system.
6. I believe I could become productive quickly using this system.
7. The system gave error messages that clearly told me how to fix problems.
8. Whenever I made a mistake using the system, I could recover easily and quickly.
9. The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.
10. It was easy to find the information I needed.
11. The information was effective in helping me complete the tasks and scenarios.
12. The organization of information on the system screens was clear.
13. The interface of this system was pleasant.
14. I liked using the interface of this system.
15. This system has all the functions and capabilities I expect it to have.
16. Overall, I am satisfied with this system.

We then added 4 extra questions for the second section, explicitly written for our application and following the same 7-point Likert Scale:

1. I was able to use the interface without the help of the tutorial.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 3 Distribution of the scores 1–7 for each question of the questionnaire. For each score and question also the corresponding percentage is reported (rounded to the nearest integer—so values in rows may not sum exactly to 100%).

	Answer distribution—Count (%)							
Question #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Avg ± std. dev.**
1. Overall, I am satisfied with how easy it is to use this system.	5 (38%)	5 (38%)	0 (0%)	2 (15%)	0 (0%)	1 (8%)	0 (0%)	2,23 ± 1,54
2. It was simple to use this system.	3 (23%)	7 (54%)	1 (8%)	1 (8%)	0 (0%)	1 (8%)	0 (0%)	2,31 ± 1,38
3. I was able to complete the tasks and scenarios quickly using this system.	5 (38%)	5 (38%)	1 (8%)	2 (15%)	0 (0%)	0 (0%)	0 (0%)	2,00 ± 1,08
4. I felt comfortable using this system.	5 (38%)	4 (31%)	3 (23%)	1 (8%)	0 (0%)	0 (0%)	0 (0%)	2,00 ± 1,00
5. It was easy to learn to use this system.	4 (31%)	6 (46%)	3 (23%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1,92 ± 0,76
6. I believe I could become productive quickly using this system.	2 (15%)	7 (54%)	2 (15%)	1 (8%)	0 (0%)	1 (8%)	0 (0%)	2,46 ± 1,33
7. The system gave error messages that clearly told me how to fix problems.	4 (31%)	3 (23%)	4 (31%)	2 (15%)	0 (0%)	0 (0%)	0 (0%)	2,31 ± 1,11
8. Whenever I made a mistake using the system, I could recover easily and quickly.	5 (38%)	4 (31%)	1 (8%)	1 (8%)	1 (8%)	1 (8%)	0 (0%)	2,38 ± 1,66
9. The information provided with this system was clear.	6 (46%)	4 (31%)	2 (15%)	1 (8%)	0 (0%)	0 (0%)	0 (0%)	1,85 ± 0,99
10. It was easy to find the information I needed.	5 (38%)	6 (46%)	2 (15%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1,77 ± 0,73
11. The information was effective in helping me complete the tasks and scenarios.	5 (38%)	5 (38%)	3 (23%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1,85 ± 0,80
12. The organization of information on the system screens was clear.	6 (46%)	4 (31%)	3 (23%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1,77 ± 0,83
13. The interface of this system was pleasant.	4 (31%)	3 (23%)	5 (38%)	1 (8%)	0 (0%)	0 (0%)	0 (0%)	2,23 ± 1,01
14. I liked using the interface of this system.	6 (46%)	3 (23%)	3 (23%)	1 (8%)	0 (0%)	0 (0%)	0 (0%)	1,92 ± 1,04
15. This system has all the functions and capabilities I expect it to have.	4 (31%)	4 (31%)	2 (15%)	2 (15%)	0 (0%)	1 (8%)	0 (0%)	2,46 ± 1,51
16. Overall, I am satisfied with this system.	5 (38%)	5 (38%)	2 (15%)	0 (0%)	1 (8%)	0 (0%)	0 (0%)	2,00 ± 1,15
17. I was able to use the interface without the help of the tutorial.	5 (38%)	1 (8%)	2 (15%)	2 (15%)	2 (15%)	1 (8%)	0 (0%)	2,85 ± 1,82
18. The tutorial is complete and sufficient to use the interface.	8 (62%)	3 (23%)	2 (15%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1,54 ± 0,78
19. The available examples are interesting.	5 (38%)	6 (46%)	2 (15%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1,77 ± 0,73
20. More complex and representative real-life examples are not necessary.	2 (15%)	2 (15%)	5 (38%)	2 (15%)	1 (8%)	0 (0%)	1 (8%)	3,15 ± 1,63

Note: The last column reports the average score for each question and its standard deviation.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 4 Score obtained in each sub-scale averaged on the collected answers.

Scale	Score
System usefulness	2,15
Information quality	1,99
Interface quality	2,21
PSSUQ score	2,09
System features	2,33
Overall score	2,14

2. The tutorial is complete and sufficient to use the interface.
3. The available examples are interesting.
4. More complex and representative real-life examples are not necessary.

Every question was associated with a text field allowing the user to comment the score given to the answer if greater or equal than 4. Moreover, a final free text field was added to give the possibility to the users to write suggestions or comments about the BUNDLE Web Application.

The PSSUQ provides an "overall score" by averaging the scores of all 16 items. Moreover, it also has 3 sub-scales, namely System Usefulness considering items 1–6, Information Quality considering items 7–12, and Interface Quality considering items 13–15. Note that the 16th item is used only in the global score. By adding 4 more questions, we added two scores: an Overall Score considering all the 20 questions, and a System Features score considering items 17–20. All the scores range between 1 and 7, the lower the better.

Table 3 reports the distribution of the scores for each question, while Table 4 reports the sub-scale and overall scores computed by averaging the collected answers. All the scores take on low values, indicating a general appreciation for the tool, with in most case a standard deviation near 1.0. This means that the application is easily usable and that most users are satisfied with the features and examples proposed, so it could indeed promote an effective spread of Probabilistic Semantic Web by means of an easy and ready-to-use reasoning system accessible from a web browser.

It is worth noting that, together with the scores, we decided to collect also "free" comments in order to improve the interface. Some of them suggested interesting features that we are going to implement as a future work: improvements to the graphic interface, to the indication of errors, filters in dropdown menus, more indication on how specify the queries, support for browsing the ontology and for composing a query. Both the available tutorial and examples are considered sufficiently useful and interesting, while regarding the necessity to add more real life examples there is more variability in the answers, thus two new ontologies have been added: BioPAX Level 2,[§§] a well-known biological ontology modelling metabolic pathways, and Good Relations,[¶¶] an ontology for the exchange of e-commerce information, that is, data on products, offers, points of sale, prices, terms and conditions, on the Web. The project is supported by, among others, Google, Yahoo, Bing, and Yandex as part of their `schema.org` initiative.

7 | CONCLUSIONS AND FUTURE WORK

In this paper we presented a Web application based on the BUNDLE framework. BUNDLE computes the probability of a query w.r.t. DISPONTE KBs by first searching for all justifications for the query by means of several ontological sub-reasoners: it can exploit state-of-the art systems such as Fact++, JFact, HermiT, and Pellet to collect the justifications, and then compute the probability of the query by means of Binary Decision Diagrams. Alternatively, it can call the TRILL reasoner to get both the set of justifications and the probability of the query.

With this application we hope to further popularize the Probabilistic Semantic Web and to spread the adoption of its tools, by giving the opportunity to perform inference without the trouble of installing software on a machine.

[§§]Containing 2331 axioms. More information on the project at http://www.biopax.org/.

[¶¶]Containing 1141 axioms. More information on the project at https://www.heppnetz.de/projects/goodrelations/.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Since its first publication online, in June 2020, the application has seen many improvements. Moreover, up to the 1st of February 2023 a total of 681 different users visited the application from all over the world, with 1030 sessions. From July 2022 to February 2023, we observed 270 users with 417 sections and a mean session time of 6 min and 28 s, demonstrating the increasing interest in the tool and in its capabilities. Thanks to the usability questionnaire results, improvements to the application are being developed at this time.

The BUNDLE Web application is available at https://bundle.ml.unife.it/.

AUTHOR CONTRIBUTIONS

Riccardo Zese: Conceptualization; investigation; software; validation; writing-original draft; writing-review and editing.
Elena Bellodi: Conceptualization; investigation; software; validation; writing-original draft; writing-review and editing.

ACKNOWLEDGMENT

This work was partly supported by the "National Group of Computing Science (GNCS-INDAM)".

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Riccardo Zese https://orcid.org/0000-0001-8352-6304

REFERENCES

1. World Wide Web Consortium (W3C). Semantic web. 2015. https://www.w3.org/standards/semanticweb/.
2. W3C. OWL 2 web ontology language. 2012. http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.
3. Baader F, Horrocks I, Sattler U. Description logics. In: van Harmelen F, Lifschitz V, Porter B, eds. *Handbook of Knowledge Representation*. Elsevier; 2008:135-179.
4. Stearns MQ, Price C, Spackman KA, Wang AY. SNOMED clinical terms: overview of the development process and project status. *Proceedings of AMIA Symposium*. American Medical Informatics Association; 2001:662-666.
5. Calero C, Ruiz F, Piattini M. *Ontologies for Software Engineering and Software Technology*. 1st ed. Springer; 2006.
6. Pileggi SF, Lopez-Lorca AA, Beydoun G. Ontology in software engineering. *Proceedings of the 29th Australasian Conference on Information Systems, Sydney, Australia, December 3–5, 2018*; 2018:1-18.
7. Gavanelli M, Lamma E, Riguzzi F, Bellodi E, Riccardo Z, Cota G. Abductive logic programming for normative reasoning and ontologies. In: Otake M, Kurahashi S, Ota Y, Satoh K, Bekki D, eds. *New Frontiers in Artificial Intelligence*. Springer International Publishing; 2017:187-203.
8. Straccia U. Towards a fuzzy description logic for the semantic web (preliminary report). In: Gómez-Pérez A, Euzenat J, eds. *The Semantic Web: Research and Applications*. Springer Berlin Heidelberg; 2005:167-181.
9. Stoilos G, Stamou G, Tzouvaras V, Pan JZ, Horrocks I. Fuzzy OWL: uncertainty and the semantic web. *Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions*. CEUR Workshop Proceedings. Vol 188. Sun SITE Central Europe; 2005:1-10.
10. Lukasiewicz T, Straccia U. Description logic programs under probabilistic uncertainty and fuzzy vagueness. *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty ECSQARU'07*. Springer-Verlag; 2007:187-2198.
11. Bobillo F, Straccia U. Reasoning within fuzzy OWL 2 EL revisited. *Fuzzy Sets Syst*. 2018;351:1-40. doi:10.1016/j.fss.2018.03.011
12. Giordano L, Gliozzi V, Theseider DD. A conditional, a fuzzy and a probabilistic interpretation of self-organizing maps. *J Log Comput*. 2022;32(2):178-205. doi:10.1093/logcom/exab082
13. Yu HQ, Reiff-Marganiec S. Learning disease causality knowledge from the web of health data. *Int J Semant Web Informat Syst*. 2022;18(1):1-19.
14. Gutiérrez-Basulto V, Jung JC, Lutz C, Schröder L. Probabilistic description logics for subjective uncertainty. *J Artif Intell Res*. 2017;58(1):1-66. doi:10.1613/jair.5222
15. Bellodi E, Lamma E, Riguzzi F, Albani S. A distribution semantics for probabilistic ontologies. In: Bobillo F, Carvalho R, da Costa PCG, et al., eds. *7th International Workshop on Uncertainty Reasoning for the Semantic Web*. CEUR Workshop Proceedings. Vol 778. Sun SITE Central Europe; 2011:75-86.
16. Riguzzi F, Bellodi E, Lamma E, Zese R. Epistemic and statistical probabilistic ontologies. *Proceedings of the 8th International Workshop on Uncertainty Reasoning for the Semantic Web, Boston, USA, November 11, 2012*. CEUR Workshop Proceedings. Vol 900. Sun SITE Central Europe; 2012:3-14.
17. Riguzzi F, Bellodi E, Lamma E, Zese R. Probabilistic description logics under the distribution semantics. *Semant Web*. 2015;6(5):447-501. doi:10.3233/SW&hyphen;140154
18. Sato T. A statistical learning method for logic programs with distribution semantics. In: Sterling L, ed. *Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995*. MIT Press; 1995:715-729.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

19. Sirin E, Parsia B, Cuenca-Grau B, Kalyanpur A, Katz Y. Pellet: a practical OWL-DL reasoner. *J Web Semant*. 2007;5(2):51-53.
20. Tsarkov D, Horrocks I. FaCT++ description logic reasoner: system description. In: Furbach U, Shankar N, eds. *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*. Lecture Notes in Computer Science. Vol 4130. Springer; 2006:292-297.
21. Shearer R, Motik B, Horrocks I, Hermi T. A highly-efficient OWL reasoner. In: Dolbear C, Ruttenberg A, Sattler U, eds. *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, Collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008*. CEUR Workshop Proceedings. Vol 432. Sun SITE Central Europe; 2008:1-10.
22. Pronto KP. A non-monotonic probabilistic description logic reasoner. In: Bechhofer S, Hauswirth M, Hoffmann J, Koubarakis M, eds. *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*. Lecture Notes in Computer Science. Vol 5021. Springer; 2008:822-826.
23. Ceylan İİ, Mendez J, Peñaloza R. The Bayesian ontology reasoner is BORN! In: Dumontier M, Glimm B, Gonçalves RS, et al., eds. *Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015) co-Located with the 28th International Workshop on Description Logics (DL 2015)*. CEUR Workshop Proceedings. Vol 1387. CEUR-WS.org; 2015:8-14.
24. van Bremen T, Dries A, Jung JC. onto2problog: a probabilistic ontology-mediated querying system using probabilistic logic programming. *Künstliche Intell*. 2020;34(4):501-507. doi:10.1007/s13218&hyphen;020&hyphen;00670&hyphen;x
25. Zese R, Bellodi E, Lamma E, Riguzzi F, Aguiari F. Semantics and inference for probabilistic description logics. In: Bobillo F, Carvalho RN, Costa PC, et al., eds. *Uncertainty Reasoning for the Semantic Web III*. Lecture Notes in Computer Science. Vol 8816. Springer International Publishing; 2014:79-99.
26. Zese R, Bellodi E, Lamma E, Riguzzi F. A description logics tableau reasoner in Prolog. In: Cantone D, Asmundo MN, eds. *Proceedings of the 28th Italian Conference on Computational Logic, Catania, Italy*. CEUR Workshop Proceedings. Vol 1068. CEUR-WS.org; 2013:33-47.
27. Zese R, Bellodi E, Riguzzi F, Cota G, Lamma E. Tableau reasoning for description logics and its extension to probabilities. *Ann Math Artif Intell*. 2018;82(1–3):101-130. doi:10.1007/s10472&hyphen;016&hyphen;9529&hyphen;3
28. Zese R, Cota G, Lamma E, Bellodi E, Riguzzi F. Probabilistic DL reasoning with pinpointing formulas: a Prolog-based approach. *Theor Pract Log Prog*. 2019;19(3):449-476. doi:10.1017/S1471068418000480
29. Riguzzi F, Lamma E, Bellodi E, Zese R. BUNDLE: a reasoner for probabilistic ontologies. In: Faber W, Lembo D, eds. *7th International Conference on Web Reasoning and Rule Systems (RR 2013), Mannheim, Germany*. Lecture Notes in Computer Science. Vol 7994. Springer; 2013:183-197.
30. Riguzzi F, Bellodi E, Lamma E, Zese R. Reasoning with probabilistic ontologies. In: Yang Q, Wooldridge M, eds. *24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*. AAAI Press; 2015:4310-4316.
31. Cota G, Riguzzi F, Zese R, Bellodi E, Lamma E. A modular inference system for probabilistic description logics. In: Ciucci D, Pasi G, Vantaggi B, eds. *Scalable Uncertainty Management 12th International Conference, SUM 2018, Milan, Italy, October 3-5, 2018, Proceedings*. Lecture Notes in Computer Science. Vol 11142. Springer; 2018:78-92.
32. Bellodi E, Lamma E, Riguzzi F, Zese R, Cota G. A web system for reasoning with probabilistic OWL. *Softw Pract Exper*. 2017;47(1):125-142.
33. Riguzzi F, Bellodi E, Lamma E, Zese R, Cota G. Probabilistic logic programming on the web. *Softw Pract Exper*. 2016;46(10):1381-1396. doi:10.1002/spe.2386
34. Renkens J, Shterionov D, Van den Broeck G, et al. ProbLog2: from probabilistic programming to statistical relational learning, NIPS probabilistic programming workshop. 2012.
35. Tudorache T, Vendetti J, Noy NF. Web-Protege: a lightweight OWL ontology editor for the web. *Proceedings of the Fifth {OWLED} Workshop on {OWL:} Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008)*, Karlsruhe, Germany, October 26-27, 2008. CEUR-WS.org; 2008:432.
36. Lewis JR. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *Int J Human Comput Interact*. 1995;7(1):57-78. doi:10.1080/10447319509526110
37. Lewis JJR, Sauro J. Revisiting the factor structure of the system usability scale. *J Usability Stud*. 2017;12(4):183-192.
38. Horrocks I, Kutz O, Sattler U. The even more irresistible SROIQ. In: Doherty P, Mylopoulos J, Welty CA, eds. *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*. AAAI Press; 2006:57-67.
39. Zese R. *Probabilistic Semantic Web: Reasoning and Learning*. Studies on the Semantic Web. Vol 28. IOS Press; 2017.
40. Kalyanpur A. *Debugging and Repair of OWL Ontologies. PhD Thesis*. The Graduate School of the University of Maryland; 2006.
41. Schlobach S, Cornet R. Non-standard reasoning Services for the Debugging of description logic terminologies. In: Gottlob G, Walsh T, eds. *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Morgan Kaufmann Publishers Inc.; 2003:355-362.
42. Baader F, Peñaloza R, Suntisrivaraporn B. Pinpointing in the description logic $\mathscr{E}\mathscr{L}^{+}$mm. In: Hertzberg J, Beetz M, Englert R, eds. *KI 2007: Advances in Artificial Intelligence*. Springer Berlin Heidelberg; 2007:52-67.
43. Horridge M, Parsia B, Sattler U. The OWL explanation workbench: a toolkit for working with justifications for entailments in OWL ontologies. 2009:1-5. http://www.semantic-web-journal.net/system/files/swj992.pdf
44. Kimmig A, Demoen B, De Raedt L, Costa VS, Rocha R. On the implementation of the probabilistic logic programming language ProbLog. *Theor Pract Log Prog*. 2011;11(2-3):235-262.
45. Reiter R. A theory of diagnosis from first principles. *Artif Intell*. 1987;32(1):57-95.

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

46. Klinov P, Parsia B. Optimization and evaluation of reasoning in probabilistic description logic: towards a systematic approach. In: Sheth AP, Staab S, Dean M, et al., eds. *The Semantic Web-ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*. Lecture Notes in Computer Science. Vol 5318. Springer; 2008:213-228.
47. Lewis JR. Psychometric evaluation of the PSSUQ using data from five years of usability studies. *Int J Hum Comput Interact*. 2002;14(3-4):463-488. doi:10.1080/10447318.2002.9669130
48. Tullis TS, Stetson JN. A comparison of questionnaires for assessing website usability. *Usability Professional Association (UPA) Conference. Minneapolis, USA,7-11 June 2004*; 2004:1-12. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.396.3677

How to cite this article: Zese R, Bellodi E. A web application for reasoning on probabilistic description logics knowledge bases. *Softw: Pract Exper*. 2023;53(9):1741-1762. doi: 10.1002/spe.3212

1097024x, 2023, 9, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/spe.3212 by Universita Di Ferrara, Wiley Online Library on [27/09/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License