Dear Editor,

please find enclosed our manuscript entitled “Beyond efficiency: A systematic classification of 48 RDFS-based Semantic Web reasoners and applications” by the authors S. Colucci, F.M. Donini, and E. Di Sciascio, submitted for possible publication to “Semantic Web Journal”.

The article is a revised version of the paper submitted to this journal with the tracking number 2483-3697 and has not been submitted for publication anywhere else. It will not be submitted to a different journal until a decision has been made by Semantic Web Journal.

Attached to the article, we provide a response letter addressing comments by reviewers.

All the authors are very grateful to you in advance for the valuable efforts you will put in place in handling our submission.

Sincerely,

On behalf of all co-authors

Simona Colucci, Ph. D.
DEI - Politecnico di Bari
Via Orabona, 4 - 70125 - Bari - Italy
Phone: +39 080 5963 222
Fax: +39 080 5963 410
sisinflab.poliba.it/colucci
Response letter

Response to Reviewer #1 (Anonymous submitted on 14/Sep/2020)

Suggestion: Major Revision

Reviewer #1:

Comment:

This manuscript was submitted as 'Survey Article' and should be reviewed along the following dimensions: (1) Suitability as introductory text, targeted at researchers, PhD students, or practitioners, to get started on the covered topic. (2) How comprehensive and how balanced is the presentation and coverage. (3) Readability and clarity of the presentation. (4) Importance of the covered material to the broader Semantic Web community.

Strong points:

1. This paper proposes a maturity model to evaluate reasoning engines. The maturity considers three orthogonal dimensions, e.g., consider or not blank nodes denotation, include different subsets of RDFS rules, provide or not explanation facilities.
2. This paper detailed illustrate these three dimensions and classify 48 RDFS-based SW reasoning engines on the maturity model. It makes sense to sort and classify reasoning machines.
3. This paper also provides some guidelines for extending their maturity model to other RDFS-based SW applications. This makes their model extensible.

Weak Points:

1. Language. It would help if you double-checked your grammar, commas, whitespace, etc.
   - At abstract, "made up by" should be "made up of" and "e.g.," misses a comma. "e.g.," should be ", e.g.,".
   - At introduction, the first sentence misses a comma after the introductory phrase "In this paper". "this sources" should be "these sources". It would be best if you considered adding a space before "Server by Intellimension". "we contribute to build" should be "we contribute to building".

Authors:

We thank the reviewer for the editorial comments. We fixed the issues and tried to improve grammar. However, the proper name of the cited system is “Semantics.Server” (with a dot in the middle of the name, and without white space), so we skipped that comment.

Reviewer #1:

2. This paper should contain an introduction to the basic concepts, not "We assume the reader familiar to RDF/RDFS syntax and semantics".

Authors:
We added a section (Section 3) for the introduction on RDF and RDFS basic concepts.

**Reviewer #1:**

3. Even though you've done some research, I don't think it's enough work, nor is it innovative enough.

From the perspective of workload:

- When evaluating the deductive capability of the system, you should select some data and queries actually to run the test system and then measure the deductive capacity of the system by comparing the results of the queries.

- "In 12 cases, such an ability is not even mentioned, and we cannot suppose it, so these applications are conservatively classified as discarding triples with blank nodes." Although not mentioned, can it be verified experimentally?

**Authors:**

We did the experimental work you suggest in a previous paper, w.r.t one specific system (S. Colucci, F.M. Donini and E. Di Sciascio, Checking compliance of semantic web applications with RDFS-semantics, Internet Technol. Lett. 2(3) (2019)) [85].

Here we only explain, in Section 5, how to extend experiments to other systems.

This paper is only about documentation, and running experiments on all 48 systems is out of the scope of a 6-weeks revision (as asked by the editor managing our submission).

**Reviewer #1:**

From the perspective of innovation:

In addition to the three dimensions mentioned in this article, you can also consider measuring the memory consumption of the system because the memory consumption is one of the critical factors for us to choose the reasoning engine.

**Authors:**

Memory consumption can be taken into account only for systems exhibiting the same reasoning behavior, e.g., the memory consumption of a triple-store retrieval is obviously different from the one of a RDFS-complete reasoner. One has first to choose if the SW application needs triple retrieval, or RDFS deductions, then use our classification to find which systems solve this need, and only then compare the memory consumption of the systems in the same class.

We added a half column to the introduction to defend our opinion on that.
Response to Reviewer #2 (Anonymous submitted on 21/Sep/2020)

Suggestion: Minor Revision

Reviewer #2:

This manuscript was submitted as 'Survey Article' and should be reviewed along the following dimensions: (1) Suitability as introductory text, targeted at researchers, PhD students, or practitioners, to get started on the covered topic. (2) How comprehensive and how balanced is the presentation and coverage. (3) Readability and clarity of the presentation. (4) Importance of the covered material to the broader Semantic Web community.

ORIGINALITY:

The authors present a classification system for RDFS-based reasoners and applications. The idea of the system comes from the software engineering maturity models. According to the maturity model in this paper, RDFS-based reasoners and applications are divided into three dimensions: processing of empty nodes, deductive capabilities and explanation of reasoning. Each dimension is divided into different levels. The idea of applying maturity model to RDFS-based reasoners and applications classification is relatively novel and interesting.

SIGNIFICANCE:

In this paper, 48 applications are selected from the list of Semantic Web Journal and W3C. The authors have devoted considerable efforts to verify whether the selected reasoners and applications conform to the various level of the proposed maturity model. This paper establishes a standard for evaluating the inference engine and its application, which is convenient for the user to select the reasoning tools and helps the developer to understand the current situation of the application.

From the charts provided in the paper, the author has done a lot of experiments. In the experiment, whether the data sets selected by different reasoners and applications are different. Please describe the data set used in the experiment.

Authors:

For this paper, we assume that the available documentation (including technical reports, and API settings) is reliable w.r.t the dimensions we proposed in our maturity model.

In section 6 we set experimental guidelines for assessing the maturity level of applications whose documentation is insufficient (possibly for commercial reasons), which is not the case of the systems we analyze here.
Response to Reviewer #3 (Anonymous submitted on 11/Oct/2020)

Suggestion: Minor Revision

Reviewer #3:

Review Comment:

Intelligent and automatic information processing requires that data and knowledge are modeled in a structured and semantic related way. Thus more and more knowledge and data are constructed and published based on Semantic Web technologies, especially RDF and RDFS. This has forced the development of many tools and systems to consume RDF(S) data sources and support Semantic Web technology-based applications. However, for end-users, too many choices usually mean no choice. This makes comparisons of the existing tools and systems much more important. Thus the authors of this work have made a thorough comparison of 48 RDFS-based Semantic Web reasoners and applications from the dimensions of considering or not blank nodes denotation, the sets of RDFS rules supported as well as the ability to explain results. These three dimensions compose the novelty of this work. The reviewer has not seen other works comparing or classifying the existing systems and tools from the perspectives provided by this work.

The paper is in general well written. It is also very easy to understand partially because it does not contain theoretical results as well as technique details. However, I have the following main questions of this paper.

----------------

1. In the Abstract, the authors concluded that their classification can be used by implementers of RDFS-based Semantic Web applications, for choosing a suitable reasoning engine, or to decide at what level an in-sourced reasoning service could be implemented and documented.

However, I think a more accurate conclusion is that their classification can be used for the applications where the three dimensions they have considered are crucial to choosing a suitable reasoning engine. Since the classification they have provided can not be referred by the applications where efficiency and scalability are much more important.

Authors:

We thank the reviewer for the opinion. Since it is another outcome of our work, we tried to add a phrase on this viewpoint too.

Reviewer #3:

2. For the systems and tools compared, the authors decided to include 32 applications presented in the Semantic Web Journal as well as 16 RDFS-based reasoners officially listed by W3C as of April 2020.

Just considering the systems and tools presented in the Semantic Web Journal sounds strange. I think that other readers may have the same feeling.

Two points make the success of the work that compares the existing systems and tools. One is the dimensions or aspects considered. And the other is the systems and tools considered.
I think the authors should consider the systems and tools recommended by W3C and/or at the same time consider those presented at JWS, ISWC, ESWC, VLDB, VLDB J, SIGMOD and so on which are high quality in the Semantic Web domain and Database domain, such as the following ones (just mention a few):


I understand that choosing the systems and tools to be included was not easy work. We may need to collect and read a lot of papers. However, if you decided to consider those not listed in https://www.w3.org/2001/sw/wiki/Category:RDFS_Reasoner, in terms of fairness and comprehensiveness, you had better consider the works presented in the Journals and Conferences mentioned above.

Authors:

We explicitly motivated our choice criteria at the beginning of Section 5 (now):

“The choice of which applications to include in this set followed the criteria below:

- **objectivity**: we searched for applications officially recognized as SW applications, rather than subjectively judging the focus to SW of available applications;
- **applicability**: applications have to manage at least RDF, given the focus of this paper;
- **availability**: information useful for the classification of applications must be available—and possibly verifiable.

In order to guarantee the objectivity of our choice, we first tried to follow a classification by W3C, and initially evaluated its list of Semantic Web Case Studies and Use Cases. However, we noticed that not all applications in the list manage RDF, making our classification not applicable to the whole list. Also, the case studies and use cases are described at a really high level, causing a low availability of information and making them hard to classify, even for applications endowed with deductive capabilities.

Thus, we reverted to the papers accepted by the Semantic Web Journal in the “Tools and Systems” section so far. These papers make available in a sufficient detail all information functional to our classification needs.”

In particular, in the description of the objectivity criterion, we clarify our intention to avoid the subjective judgement of what should be included in a catalogue of “Semantic Web applications”. Such a choice, in fact, is highly subjective and opens the way to an endless disagreement between researchers.

Our objective choice was to focus only on mature SW applications, usually presented in journal papers. In the choice of the journals to include in our analysis, we considered both SWJ and The Journal of Web Semantics by Elsevier (JWS). JWS allows for the submission of “system papers”, but only 19 of such papers have been accepted since 2003 (we browsed through all the issues 2003-2020). Moreover, only 14 of them use RDF as data model and the most recent one dates back to 2015.
SWJ, instead, has an explicit, easy to access, and up to date “Tools and Systems” section, that already included 37 tools/systems by April 2020. Such systems must include always, by editorial charter, a thorough documentation “...indicating clearly the capabilities of the described tool or system”.

Thus, our choice reverted to the most up-to-date and rich set of papers. We do know that other SW applications have been, are being, and will be proposed to conferences and journals other than the one we considered, but to exhaustively address such a moving target would make our research work not publishable at any time. The fact that we did not include some other SW systems does not mean in any way that we do not consider them as "SW". It is a matter on declaring a transparent criterion about what to include in the analysis and what to leave out now, maybe for a future extension of this analysis.

As an example, the applications suggested by the reviewer may be classified as in the following table.

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Model Dimensions</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TripleBit [Yuan et al, 2013]</td>
<td>II: Explicit treatment of blank nodes</td>
<td>I: No capability</td>
<td>I: No capability</td>
</tr>
<tr>
<td>H2RDF+ [Papaliou et al, 2014]</td>
<td>II: Implicit reference to RDF data</td>
<td>I: No capability</td>
<td>I: No capability</td>
</tr>
<tr>
<td>AdHash [Harbi et al, 2015]</td>
<td>II: Implicit reference to RDF data</td>
<td>I: No capability</td>
<td>I: No capability</td>
</tr>
</tbody>
</table>

By the way, we are skeptical about the value added by these new entries to our work, because, in our opinion, the suggested works describe RDF data management systems that we consider more a database application than a SW one (in fact, the venues in which such works were presented -VLDB and SIGMOD- are mainly focused on database research area).

With reference to the cited work by Motik et al., the reviewer seems to be not aware that we already classified the system presented in it, RDFox, in our first submission.

Probably, this is because the work cited by the reviewer presents an embryonic version of RDFox, an RDFS reasoner that now is mature and then listed by W3C. As the reviewer may notice, when an application presented in any location reaches a maturity status recognized by the SW community, the W3C should be able to detect and list it.

Thus, we are fully convinced that a subjective selection of papers to consider in a classification is out of our criteria and scope. All of us are researchers and all of us are reviewers, and the role we cover in the paper production process is not a guarantee of a higher objectivity in the selection.

Reviewer #3:

3. Undoubtedly, there are have already existing a lot of works comparing the systems and tools managing RDF(S) data sources, such as the following ones:


• Duan, S., and so on: Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets, In: SIGMOD, 2011.


Although these works compared RDF(S) systems from different aspects and views, the author should have considered these works and compare their work with these.

**Authors:**

We added to the related work (Section 2) a subsection (2.2) for the analysis of surveys on RDF-based systems. We did not cite the work by Quoc et al., because, as far as we can see, it has been submitted to the SWJ on May 2019, but not accepted so far (the last available decision is a major revision dated on October 2019).

In Section 2.2, we explicitly distinguish our contribution w.r.t. the cited surveys:

“Each of the above surveys provides an analysis that is meaningful because all the reviewed systems belong to the same category and can be compared in terms of provided features.

Our work is preliminary w.r.t. the comparative analysis of systems’ features, because it aims at determining classes of systems according to their deductive capabilities. A comparative analysis of features provided by such systems would make sense only inside the same class and has not been proposed in the literature, so far.

Moreover, we notice that the authors of cited surveys do not set criteria for choosing the analyzed systems, that makes this choice obscure and, apparently, subjective. Instead, we declare our choice criteria, making the reader aware of the reasons for the inclusion of systems in our analysis, thus ensuring the objectivity of this choice.”

**Reviewer #3:**

4. The last problem is about the dimensions or aspects that the authors have considered. The authors have emphasized multiple times that they compare and classify the systems chose from the aspects of considering or not blank nodes denotation, reasoning rules supported as well as the ability to explain reasoning results.

Although the authors have explained why considering these three dimensions, more compelling reasons should be provided to explicitly describe why these dimensions are crucial or for how many applications, these dimensions are crucial. I imagine that normally people are much more concerned about the efficiency, scalability, throughput as well as the models they use to store data and the dependence of main-memory.
The system BigData mentioned in this paper has very low speed to load RDF graphs, thus I usually ignore this system. Jena-TDB adopts a property table to manage RDF graphs, thus it usually can evaluate star-queries efficiently. On the other hand, Virtuoso adopts a three-dimension table to store RDF graphs, thus when the graphs are very large, it may spend more time evaluating the queries with many joins.

Authors:

(Sorry, but we disagree with your opinion; so we added this discussion in the introduction of our resubmission: )

“We are aware that a controversial question arises about our proposal. We try to summarize it as follows, to the best of our fairness: how important are the dimensions of blank nodes, deductive capabilities, and explanation of the results, with respect to dimensions regarding efficiency (both in terms of time and space, that is, speed and memory consumption)?

Our opinion is that such dimensions are orthogonal to efficiency, and should be considered prior to efficiency; we spend the following paragraph to defend such thesis.

Let’s start with an analogy: in the vehicles domain, it would be meaningless to compare the speed and gas consumption of a Ferrari F1 and a Toyota Pickup Truck, since they are built for different purposes. Only after the user has fixed its needs (an F1 race, or a tough terrain transport?) a comparison can be made, for artifacts built for the same purpose. In our setting, we think it would make no sense to compare, e.g., the speed of a simple RDF triple-retrieving system (no reasoning, no explanation, very fast indeed) with the speed of a complete RDFS reasoner with Explanation facilities (obviously slower), because they are artifacts built for different purposes. For a healthcare application, or a stock-market suggestion application, an explanation facility would be at a prime, because of the accountability requirements of such an application. The speed comes later, for applications solving the same need—i.e., with the same characteristics. So, a SW developer should first collect the reasoners/applications whose levels (in our proposed dimensions) fit the problem requirements; and then, compare time and space efficiency of (only) the suitable ones.”

Reviewer #3:

Next, let’s take about the dimensions/aspects this paper considered.

--

Blank nodes are used to describe the existence of some unnamed entities. The usage of blank nodes is controversial since they may have strange behavior and they make the reasoning problems to be NP-complete.

For simplicity, applications usually treat the RDF graphs with blank nodes as ground RDF graphs via considering the blank nodes as new terms.

**Authors:**

In the RDF summary section (now Section 3) requested by Reviewer#1, we included a paragraph about complexity in the presence of blank nodes, citing 3 of the 4 above papers.

Then, we spent more than one page in new Section 4 (“The Maturity Model used in the Classification”) to motivate this dimension in our submission, but we take note that we did not convince the reviewer of the usefulness of correctly managing blank nodes.

Thus, we provided in this resubmission some pointers to the SW contemporary literature that witness the importance of blank nodes in RDF. We, in fact, included the following paragraph in new Section 3:

“We stress from this point onwards the importance of blank nodes in RDF: initially proposed mainly for anonymous aggregations of data [27], blank nodes are essential in recent proposals about privacy-preserving Linked Data publishing [28–30].”


**Reviewer #3:**

Considering whether support reasoning is very useful, since most systems pursue performance and scalability, while usually ignore reasoning.

**Authors:**

We are not denying the importance of performance and scalability, that are specific declinations of efficiency. Yet, as we stated above, our opinion is that all model dimensions (including deductive capabilities/reasoning) are orthogonal to efficiency, and an efficiency comparison should be considered only among reasoning-equivalent systems.

In a dual perspective, we expect that researchers studying systems performance and scalability do not trivially “ignore” reasoning, or deny its importance, but consider it out of their scope.

As an example, Pan et al.[20] recognize the importance of reasoning in RDF systems and stress a general lack of inference ability in native RDF stores. The authors also discuss the integration
of RDF stores with available reasoners as an open issue, suggesting that storage&retrieval and reasoning are complementary abilities, which can be even implemented by different systems (to be evaluated according to different criteria).


Reviewer #3:

Explanation. RDFS reasoning is relatively simple. E.g., In the backward reasoning, the explanation can be realized by recording the rewriting procedure. Besides, end users can easily understand that the results are obtained by extending or specializing some concepts or roles. For expressive languages, such as OWL 2 DL, or machine learning-based systems, the ability to explain the results are much more important, since, for end-users, the reasoning procedures are more involved like black-boxes. The importance of the explanation of RDFS reasoning should be explained better.

Authors:

The importance of explaining a SW system conclusion is a consequence of a very general and increasing trend in eXplainable Artificial Intelligence (XAI), a research area that has now specialized workshops in main conferences, and several papers in main AI journals (we already cited some of them in our paper). The more the pervasiveness of SW applications in society—e.g., medical advice, stock market suggestions—the more the need for accountability of SW systems conclusions. In the Description Logic community OWL comes from, the research about explaining deductions is at least 20 years old (e.g., see papers by Deborah McGuinness). SW applications cannot escape such a need. We added some more arguments along this line in the paper.

Reviewer #3:

In summary, the authors should better argue that all these three dimensions are crucial.

Authors:

See the above improvements in each specific dimension.
Beyond efficiency: A systematic classification of 48 RDFS-based Semantic Web reasoners and applications

Simona Colucci a,*, Francesco Maria Donini b and Eugenio Di Sciascio a

a DEI, Politecnico di Bari, Via Orabona 4, 70125 Bari, Italy
E-mails: simona.colucci@poliba.it, eugenio.disciасcio@poliba.it
b DISUCOM, Università della Tuscia, Via S. Maria in Gradi, 4, 01100 Viterbo, Italy
E-mail: donini@unitus.it

Abstract. In this paper, we present a systematic classification of 48 RDFS-based Semantic Web reasoners and applications, with the aim of evaluating their deductive capabilities. In fact, not all such applications show the same reasoning behavior w.r.t. the RDF data they use as information source and the ability of reasoning is not a binary quality: it can, e.g., consider or not blank nodes denotation, include different subsets of RDFS rules, provide or not explanation facilities. For classification purpose, we propose a maturity model made up of three orthogonal dimensions for the evaluation of reasoners and applications: blank nodes, deductive capabilities, and explanation of the results. For each dimension, we set up a progression from absence to full compliance. Each RDFS-based Semantic Web reasoner/application is then classified in each dimension, based on both its documentation and published articles. In our evaluation, we did not consider efficiency on purpose, since efficiency could be compared only for systems providing an equal service in every of the above dimensions. Our classification can be used by Semantic Web developers, for choosing a suitable SW system, or to decide at what level an in-sourced application could be implemented and documented, in scenarios in which the three dimensions above are crucial.

Keywords: Keyword one, keyword two

1. Introduction

In this paper, we perform a systematic classification of 48 RDFS-based Semantic Web (SW) reasoners and applications. In order to compare the various systems in a uniform way, we propose three dimensions: blank nodes, deductive capabilities, and explanation of the results. For each dimension, we set some levels that a system can reach, according to the documentation available for that system. We set criteria to assess whether a system reached a given level, so that also systems not already classified here could be added to the classification later on.

The choice about which SW reasoner/application to include in our classification was not easy, since the number of such systems is large, and evolving over time: new systems may appear every month, and it is not always clear when older systems are no longer in use. Regarding RDFS-based SW applications, we decided to include 32 applications presented in the Semantic Web Journal (this journal). We also decided to include 16 RDFS-based reasoners officially listed by W3C as of April, 2020. This yielded 48 systems to classify, which were already enough for the comparative tables of a single paper. Systems left out from this initial choice could enter the classification following some general guidelines, that constitute a secondary result of this paper.

We deliberately left out SW reasoners and applications based on OWL, because literature about the eval-
ulation of such systems is already abundant (see the work by Dentler et al. [1], as an example), and includes the proceedings of a conference held for several years [2–6]. Besides that, the characteristics of OWL-based systems need dimensions very different from the ones we propose for RDFS-based SW reasoners and applications.

In order to illustrate the problems we encountered, consider the characteristics of W3C-listed RDFS reasoners: we examined in detail their technical documentation, and we noted how difficult it is to evaluate what reasoning they can perform. We just mention two cases:

1. The W3C list includes Corese, that is introduced with extreme synthesis in its Web site\(^2\), providing also some pointers to textual documentation and illustrative slides. None of these sources clearly describes the deductive capabilities of Corese, that are enclosed in fully detailed research papers and technical reports [7]. Yet, only the inspection of API documentation fully revealed the behavior of Corese;

2. another listed RDFS reasoner is Semantics.Server\(^3\), by Intellidimension, that is described at a really high level as a tool for storing and querying RDF data “using SPARQL and inference rules”. In order to learn something more about such inference rules, the reader has to look through the developers guide and its RDF API Overview, accessible at the same Web site, but not explicitly linked to the general description.

The problem is not in the documentation itself: prior to that, there should be an agreement in the Science & Technology community about what should be measured and described. What is missing is a consensus on what are the dimensions to describe, and what levels should be assessed.

As a result of our systematic classification, we contribute to build such an agreement, by proposing the dimensions above, whose notches can measure the characteristics of an RDFS-based SW reasoner/application. Such dimensions could be used by two kind of actors: (i) on one side, implementers may clearly describe the characteristics of their implementation, and (ii), on the other side, users could correctly evaluate the application they use. We observe that taken as a whole, such dimensions with recognizable levels constitute a maturity model—a well-known tool in software development, which we discuss in the next section.

We are aware that a controversial question arises about our proposal. We try to summarize it as follows, to the best of our fairness: how important are the dimensions of blank nodes, deductive capabilities, and explanation of the results, with respect to dimensions regarding efficiency (both in terms of time and space, that is, speed and memory consumption)? Our opinion is that such dimensions are orthogonal to efficiency, and should be considered prior to efficiency; we spend the following paragraph to defend such thesis.

Let’s start with an analogy: in the vehicles domain, it would be meaningless to compare the speed and gas consumption of a Ferrari F1 and a Toyota Pickup Truck, since they are built for different purposes. Only after the user has fixed its needs (an F1 race, or a tough terrain transport?) a comparison can be made, for artifacts built for the same purpose. In our setting, we think it would make no sense to compare, e.g., the speed of a simple RDF triple-retrieving system (no reasoning, no explanation, very fast indeed) with the speed of a complete RDFS reasoner with Explanation facilities (obviously slower), because they are artifacts built for different purposes. For a healthcare application, or a stock-market suggestion application, an explanation facility would be at a prime, because of the accountability requirements of such an application. The speed comes later, for applications solving the same need—i.e., with the same characteristics. So, a SW developer should first collect the reasoners/applications whose levels (in our proposed dimensions) fit the problem requirements; and then, compare time and space efficiency of (only) the suitable ones.

This survey is organized as follows: in the next section, we report on maturity models and surveys related to our work. Then, we included a brief summary of RDF in Section 3 for the broader audience. After that, in Section 4, we present our maturity model for RDFS-based SW reasoners and applications. Then, in Section 5, we show how to apply the above mentioned model, by classifying separately 32 RDFS-based SW applications and 16 RDFS-based SW reasoners. In Section 6, we provide some guidelines for extending our classification to other RDFS-based SW applications, by evaluating the level they reach in each dimension. A final section concludes the paper.

\(^2\)https://project.inria.fr/corese/

\(^3\)http://www.intellidimension.com/products/semantics-server/
2. Related Work

In analyzing related literature, we first focus on relevant maturity models proposed for the Information Technology domain in Section 2.1. Then, in Section 2.2, we analyze surveys that compare the features of RDF-based applications belonging to specific classes.

2.1. Maturity Models

In the literature, first attempts to evaluate the stage of growth of evolving entities dates back to almost 50 years ago, and include models specifically focused on Software Engineering [8]. Since then, several so-called “maturity models” have been proposed in heterogeneous research areas, ranging from Knowledge Management [9] and e-Government [10] to Information Systems [11]. Intuitively, each model comes with its own classification rules and evaluation features, but they all share some design elements, enclosed in the following definition: “a maturity model conceptually represents phases of increasing quantitative or qualitative capability changes of a maturing element in order to assess its advances with respect to defined focus areas” [12].

The maturity model we propose in this paper is designed for the classification of RDFS-based SW applications on the basis of their deductive capabilities. For this reason, we consider in this section only relevant models tailored to Information Technologies (IT) domain. Among most recognized models in IT, the Richardson Maturity Model [13] classifies services on the Web in four incremental maturity levels, based on service support to URIs, HTTP, and hypermedia (the lowest level refers to no support). A maturity model for Semantic RESTful Web APIs, called WS^3, has also been proposed [14]. WS^3 is not strictly incremental, but introduces a three-dimensional classification model for Web APIs, embedding a so-called Semantic dimension. APIs are classified as mature according to the Semantic dimension, if both resources and relationships amongst them are semantically described, without any mention to reasoning and deductive capabilities.

The problem of evaluating the maturity and applicability of SW has been investigated in the literature [15]. In particular, the work by Janev and Vranes, analyzes the status and trends in the SW at the time of writing (2009) and discusses the adoption of the SW technologies in practice. The paper provides a cross comparison of the key SW technology segments and the key application areas, as a result of the analysis of the W3C collection of Case Studies and Use Cases. Achieved results show, since 2009, that SW is not just a fashionable research issue, but a mature and applicable set of technologies.

Probably, the most well-known maturity model pertaining the SW is the five-star rating mechanism of Linked Open Data (LOD) [16]. It is an incremental model that measures the degree of availability and connection of data published as LOD in five levels of maturity. The mechanism, in other words, helps to evaluate how much linked (L) and open (O) Data classified as LOD actually are. To this aim, specific guidelines to the production of five-stars LOD have been designed [17]. The conformance of available data to such guidelines has been specifically investigated [18], through a systematic classification of several RDF datasets.

We follow an approach similar to the LOD classification: we (i) propose a maturity model for classifying RDFS-based SW reasoners and applications; (ii) classify RDFS-based systems w.r.t. this model, based on their available technical documentation; (iii) set the guidelines to classify all other RDFS-based SW systems w.r.t. the designed maturity levels.

2.2. Surveys on RDF-based applications

In this section, we report on some relevant surveys on RDF-based applications. We immediately point out that each survey compares applications sharing a common target, that may vary from survey to survey.

For example, the work by Otzu [19] compares different systems devoted to RDF data management, by classifying them in three main storage/access categories: centralized systems, distributed systems, and systems querying linked data.

Pan et al.[20] classify state-of-the-art RDF stores (defined as systems that store, query, and infer RDF data) in two distinguished categories: native and non-native. Native stores are systems built from scratch and making full use of the RDF data model to store and query. The paper compares 6 native RDF stores in terms of developing language, data model and support to inference. Regarding inference, the authors outline a general lack of inference ability and discuss the integration of RDF stores with available reasoners as an open issue. Non-native RDF stores, instead, add a specific RDF layer to an existing database storage, including relational database and NoSQL. The paper compares advantages and disadvantages of different approaches involving relational databases. Then, it intro-
3. RDF and RDFS: brief summary

Introductions to RDF are extensively covered in textbooks, e.g., [24, Chap.2–3]; we recall here only the main notions. The Resource Description Framework (RDF) is, in a nutshell, a data format equipped with a semantics. Files in the RDF data format are text files composed by triples of the form

\[ s \ p \ o . \]  

(with a full stop at the end), where \( s \) is the subject, \( p \) the property (or, predicate), and \( o \) the object of the triple. All of them are either (i) URI, or (ii) so-called blank nodes (whose names always begin with an underscore “_”), or (iii) literal values. In the initial proposal blank nodes could not appear in the predicate position, and literals could occur only as objects.\(^4\) but apart from that, every URI can appear in every position—meaning that a URI can occur in the property position in a triple, and in the subject/object position in another triple of the very same file. Typed literals can be added as special literals, whose type follows XML Schema types. We stress from this point onwards the importance of blank nodes in RDF: initially proposed mainly for anonymous aggregations of data [27], blank nodes are essential in recent proposals about privacy-preserving Linked Data publishing [28–30].

Differently from a simple data format, RDF is equipped with a model-theoretic semantics [31]; we do not delve here in the levels of such a semantics (Simple-, RDF-, and RDFS-semantics, plus the interpretation of Datatypes, called D-semantics) but we stress the fact that each level is said to be “stronger” than the previous one, in the sense that it makes some more triples to be entailed by an initial set of triples. The (increasingly stronger) entailment relations mirror the levels of semantics: Simple-, RDF-, RDFS, and D-entailment, respectively. When a system claims to adhere to one of such semantics, it is because it makes a reasoning about the logically implied triples, as if they were written inside the set.\(^5\)

We make just a simple example that is used in Section 6. In RDF-semantics the property \texttt{rdf:type} is interpreted as set membership, so that a triple of the form

\[ v v v \ \texttt{rdf:type} \ u u u . \]

is satisfied only by interpretations in which (i) a class extension is assigned to \( u u u \), and (ii) the interpretation of \( v v v \) belongs to such a class. In the (stronger) RDFS-semantics, a triple of the form

\[ u u u \ \texttt{rdfs:subClassOf} \ x x x . \]

is satisfied only by interpretations in which the class assigned to \( u u u \) is a subset of the class assigned to \( x x x \). A system adhering to RDFS-semantics, should derive from the above triples that also

\[ v v v \ \texttt{rdf:type} \ x x x . \]

is true, and use such a triple as if it occurred in the dataset. To derive such consequences, sets of correct and complete reasoning rules [24, Ch.3.3] can be used—where stronger entailment relations correspond to larger rule sets. RDF rules are given in the form of a premise \( P \) (one or more triples) and a consequence \( Q \), containing generic variables standing for actual terms

\[^{4}\text{Some authors relaxed such a constraint, e.g., in generalized triples [25, 26].}\]

\[^{5}\text{Some SW applications just materialize derived triples, so that after materialization, derived triples are actually inside the (augmented) dataset, e.g., a recent use of RDFox [32].}\]
(i.e., IRIs, literals or blank nodes) in a dataset. For instance, Rule \texttt{rdfs9} mirrors the semantic property illustrated above:

\[
\begin{align*}
P & \quad \text{uuu rdfs:subClassOf xxx .} \\
Q & \quad \text{vvv rdf:type uuu .}
\end{align*}
\]

We denote by \(\sigma\) a substitution for all variables in \(P\), that is, \(\sigma P\) contains only ground terms. An RDFS reasoner applies a rule when, given (i) a dataset \(D\), (ii) a rule whose premise is \(P\), and whose consequence is \(Q\). (iii) a substitution \(\sigma\) such that \(\sigma P \in D\), the reasoner derives from \(D\) the new triple \(\sigma Q\). Iterating rule applications, all triples that are implicitly true in RDFS entailment can be found. Systems may use only some of such rules, achieving a partial reasoning, so they can be compared on the basis of their deductive capabilities, as we do in Section 5. The lowest level in such a comparison is filled by systems that do not consider any derived triples—not even the ones entailed in Simple semantics. Such systems should be considered only as Data Management systems for the RDF data format.

We mention that the general deduction problem is NP-complete [25], even for the simplest entailment relation. However, the source of complexity lies in the size of RDF subgraphs whose nodes are all blank, all connected by the same predicate—a pattern whose size (if any) is negligible in real datasets. Without such a pattern, the problem becomes P-complete [26]—a complexity class whose complete problems are inherently serial [33], i.e., not efficiently parallelizable.

\[4.\text{ The Maturity Model used in the Classification}\]

In this section we propose a model for the classification of RDFS-based applications according to their ability to provide reasoning capabilities in RDFS.

The model is sketched in Figure 1 and includes three orthogonal dimensions, related to three capabilities crucial for a RDFS-based application providing inference: correctly treat blank nodes, perform deduction and explain reasoning. Each dimension is graduated in incremental levels of maturity, measuring to what extent an application is able to provide the above mentioned capabilities.

The three dimensions are briefly motivated in the rest of the section.

4.1. Treatment of blank nodes

Blank nodes belong to the RDF standard syntax and any application handling RDF datasets should be able to manage them for what they are: existential variables subject to different interpretations. So we identify our entry level in this dimension as the one in which triples with blank nodes are simply discarded.

Regarding the second level, we must enter a digression about what problems pose blank nodes to SW applications. Such problems do not come from basic deduction capabilities: RDFS-Rules [24] treat blank nodes correctly for what regards pure deduction, so a fair use of deduction rules (see next dimension) implies a correct treatment of blank nodes by RDFS applications. The problem comes from the numerical processing of data that most SW applications perform as a step subsequent to deduction—to compute, e.g., a similarity degree between resources, clustering, or some form of learning. In fact, almost all numerical data analysis tools need complete data, so they consider RDF data as a complete model of a domain, by implicitly adopting Closed World Assumption (CWA).

We briefly summarize CWA (in the context of SW) in the following paragraph for the interested reader.

First, we observe that a set of RDF triples forms a logic theory, not a model. The difference is that a model is always a complete representation of a domain of interest, while a theory describes the domain through a set of formulas, and it is almost always incomplete—which means that depending on how its information is completed, several different models could be obtained. The simplest way to complete a theory is to adopt Reiter’s Closed World Assumption (CWA) [34]: every formula which cannot be derived from the data is false. This is the usual assump-
tion for algorithms processing RDF data: since RDF does not allow one to express negative information, it is hypothesized when needed. For instance, a Kernel method based on random walks on RDF-graphs [35] that does not find a path in the graph, assumes the path does not exist—as if the RDF-graph were a model, not a logical theory. Reiter showed that CWA is sound when applied to a set of ground, positive, formulas (like an RDF-dataset without blank nodes), while it leads to inconsistencies when formulas describe incomplete information. Blank nodes state explicitly that data are incomplete: there is some resource, referenced by _:x, but we do not know which IRI/literal the resource is referenced by. So if a dataset D contains, say, the triple

```ex:a ex:p _:x .```

adopting CWA leads to the paradoxical conclusion that for every possible IRI/literals a, b, c, ..., (occurring or not in D)

```ex:a ex:p ex:a . is false,```
```ex:a ex:p ex:b . is false,```
```ex:a ex:p ex:c . is false,```

etc.

while the semantics of a blank node is that it denotes some resource, so at least one of the above triples is true in each model of D, hence no one of them could be assumed as definitely false.6

The usual way out from this inconsistent behavior is to skolemize blank nodes, i.e., substituting each one of them with a fictitious IRI, different from any other one, and treat them as new constants (what is proposed, e.g., by Beek et al.[37]). This approach recovers the cases like, say, the existence of a path that passes through a blank node and ends in some IRI/literal. Observe that although skolemization solves the denotation problem, it adopts the most “uninformative” denotation possible—one that invents constants that do not exist in the original domain. So we consider skolemization only as a way to preserve deductions without taking all denotations into account, and we consider this approach as our second level (more mature) of the blank node treatment.

However, note that skolemization may lead to counterintuitive results when used in real SW applications, outside deduction. For instance, imagine a dataset merging information about people, where some data sources specify a sex for each person, while others mask it through a blank node—a practice that has been recently proposed by several researchers [28–30] for anonymized, privacy-preserving data publishing. Skolemizing such privacy-motivated blank nodes as all new sexes (each one different from each other) is clearly incorrect, and may yield unpredictable results in similarity, clustering, etc.

These considerations lead us to identify a third level of maturity in the treatment of blank nodes, in which the RDF-based application takes into account several possible denotations for a blank node, and balances its behavior accordingly. For instance, blank nodes of a similar kind—e.g., based on the properties they are related to—could be assigned an IRI/literal based on some probability distribution—an already assessed practice for treating missing values in statistical analysis [38].

For convenience, we summarize below the above three levels of maturity we propose along this dimension.

1. **Discarding triples with blank nodes**: the RDF-based application ignores triples including blank nodes;
2. **Consider blank nodes (with no denotation)**: the RDF-based application considers blank nodes without considering all their possible denotations, e.g., by skolemizing them. This enables sound and complete RDFS-deduction, but may lead to unwanted results when the skolemized data are used as a model;
3. **Consider several denotations for blank nodes**: the RDF-based application takes into account different denotations for the same blank node, and its behavior is influenced by such different possibilities.

We immediately observe that the third maturity level is inapplicable for RDFS reasoners, because out of the scope of pure deduction.

### 4.2. Deductive Capabilities

This dimension evaluates the capability of RDFS-based applications to correctly apply sets of RDFS deduction rules [24] to known facts provided in the input dataset(s). So in this case, an entry level of this dimension would be no ability to apply RDFS rules: this level is not applicable for RDFS reasoners, while there may be applications based only on RDF, that do not consider deducible triples. Further levels mea-

---

6There have been proposals [36] on how to reconcile (more sophisticated forms of) CWA with existentials in datasets, but at the price of extending the syntax with Modal Logics.
s to what extent applications (including reasoners) are able to apply such rules: the second (respectively, the third) maturity level collects applications that infer triples by applying just a subset (respectively, the full set) of RDFS deduction rules.

We summarize below the three maturity levels of this dimension:

1. No capability: the RDFS-based application is only able to retrieve triples from the dataset;
2. Limited Capability: the RDFS-based application is able to deduce information by applying only a subset of RDFS deduction rules;
3. Full capability in RDFS: this level is reached by RDFS-based applications able to apply the full set of RDFS deduction rules to infer information.

4.3. Explanation Capabilities

The ability of an automated reasoning process to explain its deductions, has a long tradition of research in Artificial Intelligence [39], and received attention also in the SW community [40, 41]. Recently, thanks to widespread applications of Neural Networks and Data Science results, explanations capabilities of intelligent systems (the so-called eXplainable Artificial Intelligence, XAI) gained an ever-increasing attention, with a convergence of results from philosophy and psychology [42]. The relevance of XAI in the literature is witnessed by specialized workshops in main AI conferences7, and several papers in main AI journals [43–45].

Explanation is very important to make the end user confident about the SW application results—apart from helping implementors in the debugging phase. The more the pervasiveness of SW applications in society—e.g., medical advice, stock market suggestions—the more the need for explainability of SW systems conclusions.

We consider the ability to tell a user whether a triple was present in the dataset, or it was derived by the reasoning system inside the application, as the most basic possible form of explanation. Clearly, the next level is to show the line of reasoning leading to a conclusion, in the form of sequences of rule applications. Observe however that this kind of explanation would be meaningful for a developer, but unreadable for most end users of a SW application. For them, the explanation should be given in a human-readable format, which constitutes the highest level in this dimension.

We list the four levels of explanation ability below:

1. No explanation: the SW application presents just its results to the user, who can only decide to trust them or not.
2. Distinguish between told and derived data: the application can tell whether each triple was found in a dataset, or was deduced by the reasoner. No explanation is given about how a derived triple was deduced.
3. Explanation of deductions as rule sequences: the SW can explain its deductions, providing the sequence of RDFS rules it used to derive a specific triple. Such a sequence is meaningful only for SW experts, though.
4. Human Readable Format: the most mature level is reached by applications providing a human-readable format for the explanation of deduced information.

5. Classification of SW applications

In this section, we classify a specific set of SW applications w.r.t. the maturity model presented in Section 4. The choice of which applications to include in this set followed the criteria below:

- **objectivity**: we searched for applications officially recognized as SW applications, rather than subjectively judging the focus to SW of available applications;
- **applicability**: applications have to manage at least RDF, given the focus of this paper;
- **availability**: information useful for the classification of applications must be available—and possibly verifiable.

In order to guarantee the **objectivity** of our choice, we first tried to follow a classification by W3C, and initially evaluated its list of Semantic Web Case Studies and Use Cases8. However, the case studies and use cases are described at a really high level, causing a low **availability** of information and making them hard to classify, even for applications endowed with deductive capabilities.

Thus, we reverted to the papers accepted by the Semantic Web Journal in the “Tools and Systems” section9, as of April 2020. These papers make available...

---

7https://sites.google.com/view/xai2020/home
8https://www.w3.org/2001/sw/sweo/public/UseCases/
9http://www.semantic-web-journal.net/tools_and_systems
in a sufficient detail all information functional to our classification needs. We classify this objective list of RDFS-based SW applications in Section 5.1.

Yet, only a small portion of tools and systems in the above list provide deductive capabilities in RDFS. Thus, we moved our focus on applications surely endowed with deductive capabilities: RDFS reasoners listed by W3C, that keep applicability of our model (the ability to manage RDF is guaranteed). This objective list\(^{10}\) includes pointers to the reasoners Web sites, which, in general, make available the information we need to perform the classification.

In other words, we treat such RDFS reasoners as a subcategory of RDF-based applications, and classify their deductive capabilities w.r.t. the same model presented in Section 4.

The results of such a classification are shown in Section 5.2.

We stress the fact that when deciding the level of a system, we adopted a conservative approach: that is, when the documentation was insufficient to assess a given level of maturity, we opted for a lower level. We are not claiming that the system is definitely below a higher level; only that from its documentation, one cannot ascertain such a higher level. This sometimes points to a lack of documentation, not of system capabilities.

5.1. Classification of RDFS-based “Tools and Systems” in the SW Journal

This section shows the results of the classification we performed on the list of tools and systems published by the Semantic Web Journal (this journal). Such results derive from the study and the analysis of all the 37 papers presenting SW applications as of April 2020.

We immediately notice that our study brought us to exclude 5 tools/systems, because they do not manage RDF, making our classification inapplicable: TAO [65], Using Syntactic and Semantic Analyses to Improve the Quality of Requirements Documentation [66], ICOM 3.0 [67], Mastro [68] and OWLINK [69].

The remaining 32 RDF-based systems are classified in Table 1 and 2 according to the maturity model presented in Section 4.

We recall that the maturity model includes three dimensions: blank nodes, deductive capabilities and explanation. The reader interested in an explanation for the assigned levels in tables may check the Appendix.

Regarding the first model dimension (treating blank nodes), we found that no application is able to consider several denotations for blank nodes, as it would be imposed by their logical nature of existential variables. Yet we want to include this level now for comparison in future classifications of SW applications; in fact, when Data Analysis SW applications will meet privacy-preserving RDF datasets, well-known techniques about data imputation for missing data [83] are likely to carry over. At the second level, only 5 applications explicitly refer to the capability of treating blank nodes (with no reference to their denotation), while the rest of them just implicitly declare to be able to manage full RDF syntax. In 12 cases, such an ability is not mentioned, and since we cannot suppose it, we conservatively classify them at the lowest level of this dimension.

Regarding deductive capabilities (second model dimension), we report in Table 1 and 2 that only 13 out of 32 RDF-based SW applications are endowed with—either limited or full—deductive capabilities in RDFS. In other words, most of analyzed RDFS-based SW applications, though they manage RDF data, are not able in any way to reason about them. Moreover, 4 out of the 13 systems with deductive capabilities rely on external reasoners, whose services are invoked on-demand. Notably, all 4 systems but RacerPro[78], provide interfaces to more than one external reasoner, allowing users to choose in a selected subset of reasoners. Of course, the completeness of such “borrowed” reasoning capabilities depends on the chosen RDFS reasoner (see Appendix for details on single systems). Accordingly, we classified as “limited” (because they are outsourced) the deductive capabilities of such systems.

We also considered as “limited” the capabilities of 3 applications that, although providing some RDFS reasoning, are not fully compliant to RDFS semantics. In particular, the inference provided by Watson [79] and The Alignment API 4.0 [82] is limited to the constraint rdfs:subClassOf, while OWLIM [80] does not support reasoning with datatypes.

The remaining 6 applications (among those endowed with deductive capabilities) fully support reasoning in RDFS.

Regarding the third model dimension (explanation capabilities) we found that only 3 applications are able to provide a (simple) explanation of inference: VocBench [46], RacerPro[78] and OWLIM [80]—in

\(^{10}\)https://www.w3.org/2001/sw/wiki/Category:RDFS_Reasoner
### Table 1

Classification of the SW applications presented in SW Journal w.r.t. the proposed maturity model

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>VocBench 3 [46]</td>
<td>Explicit treatment of blank nodes</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Metaphactory[47]</td>
<td>Explicit treatment of blank nodes</td>
<td>Via external reasoner</td>
<td>x</td>
</tr>
<tr>
<td>ExConQuer[48]</td>
<td>Implicit reference to RDF data</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FRED[49]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MapOn[50]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SPARQLES[51]</td>
<td>Explicit reference to blank node checks</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SeMFIS[52]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>LOV[53]</td>
<td>Implicit reference to RDF</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Ontop[54]</td>
<td>Implicit reference to RDFS</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>YASGUI[55]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SPARKLIS[56]</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>UnifiedViews[57]</td>
<td>No mention</td>
<td>Via external reasoner</td>
<td></td>
</tr>
<tr>
<td>GERBIL[58]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>DataGraft[59]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SISSVoc[60]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TraitBank[61]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ClioPatria[62]</td>
<td>Implicit reference to RDFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>An Infrastructure for Probabilistic Reasoning with Web Ontologies[63]</td>
<td>Implicit reference to RDFS</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Dbpedia[64]</td>
<td>Implicit reference to RDFS</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Table 2
Classification of the SW applications presented in SW Journal w.r.t. the proposed maturity model

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Discard triples with blank nodes</td>
<td>Consider blank nodes (with no denotation)</td>
<td>Consider several denotations for blank nodes</td>
</tr>
<tr>
<td>Semantic Turkey [70]</td>
<td>Implicit reference to RDFS</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PowerAqua[71]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Injecting semantic annotations into (geospatial) Web service descriptions [72]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>BibBase[73]</td>
<td>No mention</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ETALIS[74]</td>
<td>Implicit reference to RDFS</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>WebProtege[75]</td>
<td>Implicit reference to RDFS</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>OntoBroker[76]</td>
<td>Implicit reference to RDFS</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S-Match[77]</td>
<td>Implicit reference to RDFS</td>
<td>Via external reasoner</td>
<td>x</td>
</tr>
<tr>
<td>RacerPro[78]</td>
<td>Implicit reference to RDF</td>
<td>Limited to sub-ClasoOf</td>
<td>x</td>
</tr>
<tr>
<td>Watson[79]</td>
<td>Implicit reference to RDF</td>
<td>RDFS without Dataypes</td>
<td>x</td>
</tr>
<tr>
<td>OWLIM[80]</td>
<td>Explicit treatment of blank nodes</td>
<td>RDFS without Dataypes</td>
<td>x</td>
</tr>
<tr>
<td>OWL API[81]</td>
<td>Implicit reference to RDF</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>The Alignment API 4.0 [82]</td>
<td>Implicit reference to RDF</td>
<td>Limited to sub-ClasoOf</td>
<td>x</td>
</tr>
</tbody>
</table>
detail, they provide a facility to distinguish told triples from those derived by inference. We also notice that only OWLIM holds this capability as an internal feature, while VocBench and RacerPro rely on explanation services provided by the external reasoners in use.

Summarizing, most of the analyzed RDF-based applications seem not to exploit the RDF semantics of the RDF data model they manage. This is true especially for the treatment of blank nodes, whose possible denotations are not considered by any application.

5.2. Classification of W3C-listed RDFS Reasoners

We classified RDFS reasoners recognized by W3C and listed at https://www.w3.org/2001/sw/wiki/Category:RDFS_Reasoner. In most cases, the referenced documentation consists in a general overview and does not give us immediate answers about the maturity levels in the three dimensions. Hence, our classification included an in-depth retrieval and analysis of the sources including the desired information. Table 3 shows such a classification. We report in the Appendix, for the interested reader, additional pointers and knowledge justifying the results of the classification in Table 3. In what follows, we just discuss some general issues related to the application of our maturity model presented in Section 4 to RDFS reasoners.

We discuss our analysis along the three dimensions of blank nodes, deductive capabilities and explanation.

Regarding blank nodes, we again point out that the third maturity level (i.e., considering several denotations for blank nodes) is not applicable to RDFS reasoners. The possibility of changing behavior according to different denotations of blank nodes pertains only to SW applications. In fact, the most mature form of treatment of blank nodes for a reasoner, is performing complete deduction also in the presence of triples including anonymous resources. To this aim, some reasoners could skolemize blank nodes, but this is an information not revealed in the documentation of analyzed tools. Of course, any SW application that includes an RDFS reasoner skolemizing blank nodes would not reach the third maturity level (see Section 4.1).

Going back to our classification, most of the RDFS reasoners explicitly declare just to treat blank nodes

\[\text{\footnotesize\textsuperscript{11}}\] We removed from the list two reasoners: OWLIM, that is just the old name of GraphDB—and is analyzed in its original version in Section 5.1.—and OpenAnzo, that seems to be permanently unavailable, as of April 2020.

in the deduction process. Some reasoners, instead, implicitly refer to RDF standard syntax or RDF complete terms, letting us suppose their ability to treat also blank nodes. Also, SHER\[\text{\footnotesize\textsuperscript{12}}\] seems to limit existential assertions only to those expressible in the Description Logic SHIN.

As for deductive capabilities, it is not surprising that all reasoners are able to deduce information, so each one of them outmatches the first maturity level. In addition, we discovered from our analysis that all reasoners are able to combine different datasets as basis for their deduction in RDFS. Nevertheless, not all RDFS reasoners attain the third level in this dimension: we highlighted some exceptions in Table 3: some reasoners do not apply the complete set of RDFS rules (see OpenLink Virtuoso\[\text{\footnotesize\textsuperscript{13}}\] and SHER, for example) or require the explicit specification of the inference rules by the user (see StrixDB\[\text{\footnotesize\textsuperscript{14}}\] and TopBraid\[\text{\footnotesize\textsuperscript{15}}\], for example).

Regarding explanations, the behavior of RDFS-reasoners is quite heterogeneous: the majority of them is either simply not able to explain deduction, or just returns the number of inferred triples. Only a few reasoners attain the second level: they distinguish told and derived triples, as a bare form of explanation. Four reasoners pass the third level: they are able to show the rule sequence leading to derived information; and two of them visualize such information in a human-readable format, attaining the highest level in this dimension.

6. Guidelines for extending the classification

Other SW applications may immediately join the previous classification if their documentation explicitly describes the features associated to each level.

Instead, when the documentation is insufficiently detailed, the application can be considered a black box. In this case, there is the need to conduct experiments along some guidelines, that we provide here.

We briefly discuss each dimension separately.

\[\text{\footnotesize\textsuperscript{12}}\] https://www.w3.org/2001/sw/wiki/SHER
\[\text{\footnotesize\textsuperscript{13}}\] http://docs.openlinksw.com/virtuoso/index/
\[\text{\footnotesize\textsuperscript{14}}\] http://oporel.free.fr/strixDB/
\[\text{\footnotesize\textsuperscript{15}}\] http://wiki.topquadrant.com/display/master/TopBraid+Suite+Documentation+6.2+Home
### Table 3
Classification of the RDFS reasoners listed by W3C w.r.t. the proposed maturity model

<table>
<thead>
<tr>
<th>RDFS Reasoner</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explaination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Discard triples with blank nodes</td>
<td>Consider blank nodes (with no denotation)</td>
<td>No capability</td>
</tr>
<tr>
<td>AllegroGraph</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Number of inferred triples</td>
</tr>
<tr>
<td>Apache Jena</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>Bigdata</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>CoRe</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>GraphDB</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>Intellidimension</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>OntoBroker</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>OpenLink Virtuoso</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>OWLRL</td>
<td>Implicit reference to complete RDF(S) terms</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>Parliament</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>Profium Sense</td>
<td>Implicit reference to RDF standard syntax</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>RDFox</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>Sesame</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>SHER</td>
<td>Limited to SHIN assertions with existential</td>
<td>SHIN</td>
<td>Only told triples</td>
</tr>
<tr>
<td>StrixDB</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
<tr>
<td>TopBraid</td>
<td>Explicit treatment of blank nodes</td>
<td>x</td>
<td>Only told triples</td>
</tr>
</tbody>
</table>
6.1. Devising experiments for blank nodes

If the SW application is a black box, a general guideline for experiments about blank nodes is to feed the application with two datasets $D_1$ and $D_2$: in $D_1$, one or more resources are explicitly referenced (e.g., an IRI referring to winners of a sports event, some literals representing their scores, etc.), while in $D_2$ some of these IRIs/literals are replaced by blank nodes. The application should:

1. take the blank nodes in $D_2$ into account (difference between 1st and 2nd level);
2. behave differently for the two datasets, in the sense that in $D_2$ it should take into account both the possibility that the blank nodes coincide with other resources, and the possibility that they do not coincide (difference between 2nd and 3rd level).

We already used such guidelines [84, 85] to run some experiments on a renowned data analysis software: RapidMiner with LODextension [86]. We recorded similarity values computed by the system for some IRIs in $D_1$, and then in $D_2$, and assessed that RapidMiner computes similarity as if the blank nodes denote resources different from any other one (so RapidMiner uses some form of skolemization).

6.2. Experiments on deductive capabilities

Again, also in this case the RDFS-based SW application can be fed with two different datasets $D_1$ and $D_2$. Intuitively, adding explicitly in $D_2$ the consequence of a rule in $D_1$ should make no difference for an application treating triples with RDFS-semantics. More formally, recall from Section 3 that RDFS rules are given in the form of a premise $P$ (one or more triples) and a consequence $Q$. If there exists a substitution $\sigma$ such that $\sigma P \subseteq D_1$, an RDFS reasoner should behave in the same way when fed with the two datasets: $D_1$, and $D_2 = D_1 \cup \{\sigma Q\}$.

Just to make a concrete example, consider Rule rdfs9 in Section 3. A test dataset $D_1$ may contain the triples:

```
  ex:b rdfs:subClassOf ex:c .
  ex:a rdf:type ex:b .
```

Such triples are an instance of premise $P$ of rdfs9; let denote this instance $\sigma P$, with $\sigma = \{uuu/ex:b, xxx/ex:c, vvv/ex:a\}$. The application of rdfs9 yields the triple $\sigma Q$, below, as a consequence:

```
  ex:a rdf:type ex:c .
```

Then, an application attaining at least the second level of the “Deductive capabilities” dimension should behave in the same way either when it is given $D_1$ as input, or $D_2 = D_1 \cup \{ex:a rdf:type ex:c .\}$.

Note that for RDFS-based SW applications, we insist that only the behavior of the application should be the same, not if and how the application is able to perform an explicit deduction. For example, if the application involves a similarity computation, or a clustering later in the processing, we insist that such computations do not depend on the absence/explicit presence of the logical consequences of all deductive rules.

We already applied this schema [85], to evaluate reasoning abilities of RapidMiner with LODextension on some RDFS rules, confirming that it behaved as if all deductions we tested were already performed in the system.

6.3. Explanation capabilities

Clearly, if a RDFS-based SW application provides an explanation, this is a sufficient evidence by itself of its explanation capabilities. So in this case, we did not devise experiments to check the explanation capabilities of an application.

We note however that what constitutes an explanation is a highly subjective judgment, that depends on the background of whom reads the explanation [87]—an explanation is a “communicative act” [42]. Hence in this dimension, the cognitive abilities of end users should be taken into account, verifying, e.g., that an explanation in “human-readable” format is really so. Language comprehension tests could be used for this verification, with an aim opposite of the usual one: in this case, what is tested is not the reader’s ability to comprehend text, but the clarity of the text providing the explanation, with respect to a specific class of users.

7. Conclusion

Maturity models help advancing development methodologies, by marking levels of maturity that an application may reach, better weighing benefits in a cost/benefit analysis, and most of all, by establishing a communication consensus—a common language—about which dimensions (and notches on them) an application can be evaluated along. In this paper, we proposed a maturity model for RDFS-based SW applications, and applied the model in the analysis of the documentation of 32 RDFS-based SW applications.
and 16 RDFS reasoners, yielding 48 systems in total. We also used the maturity levels of each dimension to guide the setup of simple black-box experiments assessing the level reached by an application, in order to extend our classification to systems for which the documentation is absent or insufficient.

Our maturity model tries to help standardizing in a uniform way the developers’ descriptions of the reasoning capabilities of the available RDFS-based SW applications:

1. from the viewpoint of a developer writing the system documentation, it could help describing—already in the system overview—salient characteristics of the application made available;
2. from the viewpoint of someone evaluating the application (an end user, a middleware developer combining systems), it could place on a scale these characteristics, and help choosing the one that best fits his/her needs.

Appendix: Explanation of classification results

In Section 5, we presented only the final results of the classification of 32 RDFS-based SW applications (Section 5.1) and 16 RDFS reasoners (Section 5.2) w.r.t. the maturity model in Section 4. In this appendix, we report in tables, for the interested reader, some additional knowledge explaining such results. In particular, Tables 4, 5, 6, 7, 8 explain the classification of applications provided in Section 5.1. Tables 9, 10, and 11 provide, instead, an explanation for the classification of RDFS reasoners shown in Section 5.2. In all tables, for each application/reasoner, we provide: i) a reference to a general document describing it, besides its name, in the first column of tables; ii) the maturity level assigned for each dimension (refer to the numbering of levels in Figure 1); iii) the information source used to assign such levels (where not specified, the level is assigned on the basis of the general reference immediately after the name of the application/reasoner).

References


Table 4

The Table summarizes, for the first a subset of analyzed SW Applications, the assigned maturity level in the three dimensions and the information source at the basis at the classification.

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Model dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blank Nodes</td>
</tr>
<tr>
<td>VocBench 3 [46]</td>
<td>Level</td>
</tr>
<tr>
<td>Metaphactory [47]</td>
<td>Level</td>
</tr>
<tr>
<td>FRED [49]</td>
<td>Level</td>
</tr>
<tr>
<td>Map-On [50]</td>
<td>Level</td>
</tr>
<tr>
<td>SPARQL-ES [51]</td>
<td>Level</td>
</tr>
<tr>
<td></td>
<td>Information source</td>
</tr>
<tr>
<td>VocBench 3 [46]</td>
<td>Information source</td>
</tr>
<tr>
<td>Metaphactory [47]</td>
<td>Information source</td>
</tr>
<tr>
<td>FRED [49]</td>
<td>Information source</td>
</tr>
<tr>
<td>Map-On [50]</td>
<td>Information source</td>
</tr>
<tr>
<td>SPARQL-ES [51]</td>
<td>Information source</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S. Colucci et al. / Beyond efficiency: A systematic classification of 48 RDF-based Semantic Web reasoners and applications 15
Table 5

The Table summarizes, for a subset of analyzed SW Applications, the assigned maturity level in the three dimensions and the information source at the basis of the classification.

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Model dimensions</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeMFIS[52]</td>
<td>Level</td>
<td>I: No mention</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Information source</td>
<td>&quot;SeMFIS does however not provide reasoning mechanisms, rule engines, or further constraints regarding the processing of the semantics in the ontologies. If needed, these functionalities can be easily added to SeMFIS using third-party tools and APIs via the subsequently discussed interfaces and scripting functionalities.&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOV[53]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Information source</td>
<td>&quot;Ontop implements the Sesame Storage And Inference Layer (SAIL) API supporting inferencing and querying over relational database...&quot; &quot;...the resulting T-mappings define all the triples in the virtual RDF graph that includes all the inferences due to the ontology (under the entailment regime)&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ontop[54]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>III</td>
<td>I</td>
</tr>
<tr>
<td>Information source</td>
<td>&quot;Blank nodes are correctly handled. They are shown in results, and suggestions are given about them, but they cannot be inserted in the query&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YASGUI[55]</td>
<td>Level</td>
<td>I: No mention</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Information source</td>
<td>&quot;Another limitation is when the end-point does not apply RDFS or OWL inference, so that some suggestions are missing. From its client side, SPARKLIS has no easy way to compensate for this lack of inference.&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPARKLIS[56]</td>
<td>Level</td>
<td>II: Explicit treatment of blank nodes</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Information source</td>
<td>&quot;Every backend uses its own RDF Working Store for storing temporary data which is produced by the pipeline during its execution. As the RDF Working Store, we currently support Sesame repositories, either as a local native store or on a remote server, and GraphDB. Experimental support for OpenLink Virtuoso is also in place and can be enabled by a configuration option. In general, any repository which supports the OpenRDF Sesame API can be used as an RDF Working Store&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnifiedViews[57]</td>
<td>Level</td>
<td>I: No mention</td>
<td>II: Via external reasoner</td>
<td>I</td>
</tr>
<tr>
<td>Information source</td>
<td>&quot;While SISSVoc does not explicitly feature logics-based reasoning in the API, this functionality could be included in a SISSVoc deployment by building on triple store implementations which support reasoning (e.g. OWLIM)&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6
The Table summarizes, for a subset of analyzed SW Applications, the assigned maturity level in the three dimensions and the information source at the basis of the classification.

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Model dimensions</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TraitBank[61]</td>
<td>Level</td>
<td>I: No mention</td>
<td>I</td>
<td>&quot;These links improve the discoverability and queriability of the data and provide interoperability with other semantic resources, but more principled inference is left to end users.&quot;</td>
</tr>
<tr>
<td>ClioPatria[62]</td>
<td>Level</td>
<td>II: Implicit reference to RDFS</td>
<td>III</td>
<td>I</td>
</tr>
<tr>
<td>An Infrastructure for Probabilistic Reasoning with Web Ontologies [63]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>III</td>
<td>I</td>
</tr>
<tr>
<td>Dbpedia[64]</td>
<td>Level</td>
<td>II: Implicit reference to RDFS</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Semantic Turkey[70]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>PowerAqua[71]</td>
<td>Level</td>
<td>I: No mention</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>
### Table 7

The Table summarizes, for a subset of analyzed SW Applications, the assigned maturity level in the three dimensions and the information source at the basis of the classification.

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Level</th>
<th>Model dimensions</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Injecting semantic annotations into (geospatial) Web service descriptions [72]</td>
<td></td>
<td>Blank Nodes</td>
<td>I. No mention</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deductive Capabilities</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Explanation</td>
<td></td>
</tr>
<tr>
<td>Information source</td>
<td></td>
<td></td>
<td>&quot;The proxy itself is not semantically enabled: it does neither enable users to create semantic annotations, nor does it perform reasoning on the semantic annotations.&quot;</td>
</tr>
<tr>
<td>BibBase[73]</td>
<td>Level</td>
<td>I: No mention</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deductive Capabilities</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Explanation</td>
<td></td>
</tr>
<tr>
<td>ETALIS[74]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>III</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deductive Capabilities</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Explanation</td>
<td>&quot;ETALIS can evaluate domain knowledge on-the-fly, thereby proving semantic relations among events and reasoning about them. This important feature has been recognised recently in various related approaches. In contrast to these, ETALIS follows a completely deductive rule-based paradigm, thereby providing an effective solution for CEP and Stream Reasoning.&quot; &quot;Event streams are expected to be represented as timestamped RDF triples [4], and background knowledge can be specified as an RDFS ontology&quot; &quot;Note that, by using deductive rules, ETALIS can be used to infer implicit knowledge (i.e., not only explicitly stated knowledge)&quot;</td>
</tr>
<tr>
<td>WebProtege[75]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deductive Capabilities</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Explanation</td>
<td></td>
</tr>
<tr>
<td>OntoBroker[76]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>III</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deductive Capabilities</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Explanation</td>
<td>&quot;OntoBroker is an ontology repository that includes a high performance deductive reasoning engine. RDF(S) and OWL are translated into ObjecLogic primitives and SPARQL queries are translated into ObjecLogic queries. &quot; &quot;As OntoBroker also provides rule materialization, i.e. executing all of the rules and storing the data in the internal ontology store, this also includes extraction of the data from the data sources and storing them locally in the own store. On the other hand, usually ontoprise’s customers prefer query-time integration as they want to continue to maintain their own data stores.&quot;</td>
</tr>
<tr>
<td>S-Match[77]</td>
<td>Level</td>
<td>II: Implicit reference to RDFS</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deductive Capabilities</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Explanation</td>
<td></td>
</tr>
<tr>
<td>RacerPro[78]</td>
<td>Level</td>
<td>II: Implicit reference to RDFS</td>
<td>II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deductive Capabilities</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Explanation</td>
<td>&quot;Ontologies can be read from files, or can be retrieved from the web as well as from an RDF triple store managed by the built-in AllegroGraph system (version 3) from Franz Inc. AllegroGraph can be used to store materialized inferences and also provides for a powerful query language based on SPARQL syntax (1).&quot; &quot;Query results (bindings for variables) can be interactively inspected and Aboxes can interactively explored in the Abox inspector showing told and inferred assertions (see Figure 6).&quot;</td>
</tr>
</tbody>
</table>
Table 8

The Table summarizes, for a subset of analyzed SW Applications, the assigned maturity level in the three dimensions and the information source at the basis at the classification.

<table>
<thead>
<tr>
<th>SW Applications</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Information source</td>
<td></td>
<td>&quot;Functions are provided that allow an application to access the content</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of an ontology, through exploring its entities and their connections. These</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>functions include the possibility to ask for the subclasses of an RDF, DAML</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>or OWL class in any of the collected ontologies, the labels of a given</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>entity or any relation pointing to a given individual. Some of these</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>functions are also available in a variant providing basic level reasoning,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in order to, for example, obtain all the subclasses of a class, i.e., both</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the directly declared ones and the ones inferred from the transitivity of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the subclass relation&quot;</td>
</tr>
<tr>
<td>Watson[79]</td>
<td>Level</td>
<td>II: Limited to subClassOf</td>
<td>I</td>
</tr>
<tr>
<td>OWLIM[80]</td>
<td>Level</td>
<td>II: Explicit treatment of</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blank nodes</td>
<td>&quot;Free variables in rule heads are treated as blank nodes&quot;</td>
</tr>
<tr>
<td></td>
<td>Information source</td>
<td></td>
<td>The inferencing strategy in OWLIM is one of total materialization (apart</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>from the owl:sameAs optimization discussed in Section 1) based on R-Entail-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ment (as defined by ter Horst), rdfs RDFS semantics using rule entailment,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>but without data-type reasoning, i.e. without the literal generalization and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>related rules;&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>II: RDFS without Datatypes</td>
<td>&quot;Since the semantics (both standard and custom) must be monotonic, insert</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>operations incrementally add to the set of explicit and inferred statements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>However, retracting explicit statements that are used to infer other</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>statements is more complicated.&quot;....any inferred statements affected by</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>inserts and deletes will also be subject to handling by the notification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mechanism, i.e. new implicit statements will also be notified to clients</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>when the requested triple pattern matches. The purpose of the notification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>service is to enable the efficient and timely discovery of newly added or</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>deleted RDF data.&quot;</td>
</tr>
<tr>
<td>OWL API[81]</td>
<td>Level</td>
<td>II: Implicit reference to RDF</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Information source</td>
<td></td>
<td>&quot;The OWL API provides a collection of powerful and flexible interfaces sup-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>porting the use of OWL ontologies within applications. The model explicitly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>supports the recent OWL 2 Recommenda-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tion. Common interfaces to reasoning engines are defined, facilitating the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>use of inference within applications&quot;</td>
</tr>
<tr>
<td>The Alignment API 4.0[82]</td>
<td>Level</td>
<td>II: Limited to subClassOf</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>Information source</td>
<td></td>
<td>&quot;Unasserted superclasses can be taken within those superclasses that can be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>obtained by limited reasoning methods like Inherited superclasses obtained</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>only by transitive closure computation&quot;</td>
</tr>
</tbody>
</table>
Table 9
The Table summarizes, for the first 4 analyzed RDFS reasoner, the assigned maturity level in the three dimensions and the information source at the basis of the classification.

<table>
<thead>
<tr>
<th>RDFS Reasoner</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllegroGraph 2</td>
<td>Level II: Explicit treatment</td>
<td>III</td>
<td>I: Number of inferred triples</td>
</tr>
</tbody>
</table>

Information source

"Determine how to handle blank node identifiers in N-TRiple and N-Quad files. STRATEGY must be one of file, job or none. By default, blank node identifiers are scoped to the source in which they appear. i.e., the blank node _:\_b1 in file file1.nt is considered to be different than the blank node _:\_b1 in file file2.nt. AllegroGraph calls this the file strategy and uses it as the default. Blank node strategy job will consider all blank nodes found in N-TRiple and N-Quad files to be in the same "scope". This means that the _:\_b1 in file1.nt will be considered to be the same as the one found in file2.nt. Blank node strategy none will cause agtool load to error if any sources contains blank nodes. Loading is faster when the blank node strategy is none." "AllegroGraph offers a very fast and practical RDFS++ reasoner. We support all the RDF and RDFS predicates and some in full OWL."  

Apache Jena 5

Level II: Explicit treatment | III | III |

Information source

"Standard RDF semantics for blank nodes requires that an export/import process maintains a mapping from the blank node ID to the internal BNode object used to model that blank node. This works fine as long as you export/import a KB as a single RDF document. However, if references to the same blank node ID appear in different RDF documents then they will be construed as distinct blank nodes!" "Bigdata also supports a "told bnodes" option. When using this option, the blank node IDs are treated in much the same manner as URIs. They are stable identifiers which may be used to refer to the blank node. In this case, the interchange of RDF data may be broken down into multiple documents and blank node identity will be preserved."  

Bigdata 6

Level II: Explicit treatment | III | II: told triples |

Information source

"When a KB contains materialized inferences you will typically want to export only the "told" triples (those explicitly written onto the KB by the application). After you import the data you can then recompute the materialized inferences. If you export the inferences and/or axioms as well then they will become "told" triples when you import the data into a new bigdata instance." 

Corese 9

Level II: Explicit treatment | III | I |

Information source

[88], [7]
The Table summarizes, for other seven analyzed RDFS reasoner, the assigned maturity level in the three dimensions and the information source at the basis at the classification, if needed.

<table>
<thead>
<tr>
<th>RDFS Reasoner</th>
<th>Model dimensions</th>
<th>Blank Nodes</th>
<th>Deductive Capabilities</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GraphDB</strong>&lt;sup&gt;11&lt;/sup&gt;</td>
<td>Level II: Explicit treatment</td>
<td>III</td>
<td>II</td>
<td></td>
</tr>
<tr>
<td><strong>IntelliDimension</strong>&lt;sup&gt;14&lt;/sup&gt;</td>
<td>Level II: Explicit treatment</td>
<td>III</td>
<td>II</td>
<td></td>
</tr>
<tr>
<td><strong>Ontobroker</strong>&lt;sup&gt;17&lt;/sup&gt;</td>
<td>Level II: Explicit treatment</td>
<td>III</td>
<td>IV</td>
<td></td>
</tr>
<tr>
<td><strong>OpenLink Virtuoso</strong>&lt;sup&gt;16&lt;/sup&gt;</td>
<td>Level II: Explicit treatment</td>
<td>II: rdfs:subClassOf rdfs:subPropertyOf in a Rule set</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td><strong>OWLRL</strong>&lt;sup&gt;21&lt;/sup&gt;</td>
<td>Level II: Implicit reference to complete RDF(S) terms</td>
<td>III</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td><strong>Parliament</strong>&lt;sup&gt;21&lt;/sup&gt;</td>
<td>Level II: Explicit treatment</td>
<td>III</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td><strong>Profium Sense</strong>&lt;sup&gt;25&lt;/sup&gt;</td>
<td>Level II: Implicit reference to RDF standard syntax</td>
<td>III</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>
The Table summarizes, for the last 5 analyzed RDFS reasoners, the assigned maturity level in the three dimensions and the information source at the basis of the classification, if needed.

<table>
<thead>
<tr>
<th>RDFS Reasoner</th>
<th>Model dimensions</th>
<th>Information source</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RDFox</strong></td>
<td>Level II: Explicit treatment</td>
<td>Information source</td>
<td>III</td>
</tr>
<tr>
<td><strong>Sesame</strong></td>
<td>Level II: Explicit treatment</td>
<td>Information source</td>
<td>III</td>
</tr>
<tr>
<td><strong>SHER [90]</strong></td>
<td>Level II: Limited to SHIN assertions with existential</td>
<td>Information source</td>
<td>IV</td>
</tr>
<tr>
<td><strong>StrixDB</strong></td>
<td>Level II: Explicit treatment</td>
<td>Information source</td>
<td>II: SHIN</td>
</tr>
<tr>
<td><strong>TopBraid</strong></td>
<td>Level II: Explicit treatment</td>
<td>Information source</td>
<td>I</td>
</tr>
</tbody>
</table>


List of RDFS reasoners

9. https://project.inria.fr/corese/
27. https://oxfordsemtech.github.io/RDFDocLib_versions/1.3.3/#04-using?id=managing-data-stores
28. https://oxfordsemtech.github.io/RDFDocLib_versions/1.3.3/#
33. http://oporel.free.fr/strixDB/