Taking Advantages of Automated Reasoning in Visual Ontology Engineering Environments

Germán BRAUN $^{\rm a,b,c},$ Laura CECCHI $^{\rm a}$ and Pablo FILLOTTRANI $^{\rm c,d}$

^a Informatics Faculty, Universidad Nacional del Comahue, Argentina ^b Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina ^c LISSI, Computer Science and Engineering Department, Universidad Nacional del Sur,

Argentina

^d Comisión de Investigaciones Científicas de la provincia de Buenos Aires (CIC), Argentina

AbstractIt is well-known that automated reasoning provides important support for tools in ontology engineering, particularly, revealing errors and unexpected (non-) entailments in models. This has been possible thanks to the standardisation of ontology languages and the subsequent development of tools and infrastructure to manipulate ontologies in such languages. However, in spite of the fact that nowadays reasoning systems are integrated into a huge range of these tools, their use for enhancing automated tasks in ontology engineering processes has not been fully explored in depth, particularly in closing the gap between graphical and formal representations of ontologies. In this work we detail two scenarios in which it is possible to take advantages of automated reasoning in this sense. The first scenario deals with the problem of tool interoperability, and the second analyses how to obtain non-trivial graphical inferences.

Keywords. automated reasoning, ontology engineering, modelling tools

1. Introduction

The *cross-fertilisation* between visual knowledge representation systems and logic formalisms has been extremely productive since the research area of Description Logics (DLs) emerged as a structured and well-understood way to provide a precise semantic characterisation of frames and semantic network tools [17]. Visual modelling languages are more human-centered, and as such they are more intuitive and promote applicability. New system developments generate representational and algorithmic challenges for logics. On the other hand, logical systems supply an unambiguous semantics and wellknown computational properties. Theoretical contributions induce the development of new technologies to be used in more effective tools. This lead to a very close interaction between basic and applied computer science, which distinguishes the research area.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

In the last years there have been important theoretical advances in precisely defining the semantics and computational properties of diverse DL profiles[1], shifting the attention focus towards tools for the development of these logical systems and thus taking advantages of reasoning services. Nevertheless, modelling using formal logic is a difficult task as it does not provide practical and methodological means for ontology engineering. In addition, engineers are required to understand the logical foundation underpinning such logic, which is very difficult for domain modellers. In this paper we aim at analysing how could ontology engineering processes, and the relevant tools, take advantages of automated reasoning to improve the quality of the outcome. The importance of this question has been already considered in [11], where it is stated that reasoning enabled tools provide vital support for ontology engineering by giving tools the ability to detect errors and unexpected (non-) entailments. There it is also marked the key benefits flowing from OWL standardisation [16] and the subsequent development of tools and infrastructure that can be used to support the development and deployment of OWL ontologies. Most recently, these statemens are being developed as the case of the VADALOG system [8] and the OBDA paradigm [22]. The former emerged as a platform aiming at managing knowledge graph applications and integrating machine learning and data analytics with logical reasoning, while the latter presents query answering as the main reasoning task for querying relational data sources through an ontology.

In spite of the fact that nowadays reasoning systems are integrated into a huge range of tools, their use for defining or enhancing automated tasks is still undervalued. In this work we detail two scenarios and experiments in which we describe how to take advantages of the automated reasoning for ontology designing tasks and how it should be used considering the diverse reasoning tools, protocols and underlying languages support. In particular, we focus on how this can be applied to ontology engineering in the context of a visual tool. At the end of each scenario we present algorithms to tackle the found weaknesses, which take advantages of automated reasoning a visual tool. We outline on how diverse tools provide different approaches to manage ontology engineering tasks involving automated reasoning, creating a mismatch between the theory of precise semantics and the practice.

This work is structured as followed. Section 2 details the context in which our research takes place and including most relevant related works. Section 3 presents the scenarios, experiments and discussions about how automated reasoning is integrated to ontology engineering tasks. Section 4 presents final discussion and conclusions aiming at generating a deeper insight in visual-formal interactions.

2. Context

Formalising the graphical and logical interaction has several advantages. First, it can be used as a basis for design and implementation of novel graphical applications for conceptual modelling and ontology engineering, providing a terminology and understanding of the requirements for design and implementation of this kind of environment. Second, it allows to evaluate and classify existing tools according to how they manage the integration of both aspects (logical and visual). Third, it is useful for neatly characterising the expressive power of any tool in both directions: visual and formal. In addition to VADALOG and OBDA systems, there are other related tools in this approach, particularly ICOM [6], NORMA [20], Menthor [15], OWLGrEd [5], VOWL [14] and eddy [13], which are considered here because they are based on graphical onotlogy representation with diverse treatment of automated reasoning. ICOM, NORMA, Menthor and OWLGrEd present some similarities in the treatment of automated reasoning, however, although they interact with reasoners, its integration to the visual languages and ontology engineering processes is still limited. ICOM and NORMA show implicit properties, subsumptions, equivalences, disjoints and cardinalities on the very same diagram, while Menthor and OWLGrEd use reasoning only for satisfiability checking. On the other hand, next tools define their own visual language: VOWL is a mere graph-based visualiser, while eddy is a graphical editor for the specification and visualization of ontologies. In both, the integration of reasoning with diagrams is absent. Obviously, we cannot leaving out to the well-known Protégé tool [12] and its recent cloud-based version [10], however, the visual support of these tools is not between their main strengths presenting limited plug-ins without a comprehensive interaction with reasoning services [21].

In addition, several of reviewed tools implement just a few ontology engineering tasks (only the popular Protégé is considered as an integrated ontology development tool), providing very different user interfaces, graphical and formal languages and reasoning capabilities. Generally, these processes appear to be fragmented across diverse tools and workarounds. This lack of adequate and seamless tools potentially hinders the broad uptake of ontologies by modellers not skilled in logics, especially OWL, as a knowledge representation formalism. As a consequence, this analysis gives a real insight into going towards an unified tool integrating not only the visual notations and their common features, but also supporting other activities of ontology engineering altogether in a single graphical and logical framework with a set of core functionalities.

In [4, 3], we present the idea of associating a well-defined semantic to a set of primitives from a visual language and formalising the key elements of this concept. The central concept of this approach is the definition of a graphical ontology, which states a visual model in terms of a DL knowledge base. Following these ideas we developed *crowd*¹, a web-based tool interacting with reasoning systems for ontology engineering. The tool provides a UML editor and uses DL-based reconstructions of such conceptual data modelling language, based on [2, 6], although it is being currently extended to support EER and ORM 2.

3. Taking Advantages of Automated Reasoning

We present here two scenarios where automated reasoning can be integrated into ontology engineering tasks in the context of visual environments. Particularly, we analyse issues related to: (a) interoperability between tools in the context of importing tasks; and (b) how the automated reasoning can be used for inferring new cardinality constraints in a graphical model of an ontology. The aim of this evaluation is to state how the automated reasoning can be applied to manipulating graphical ontologies. The scenarios of evaluating are described first, and subsequently the results and discussion thereof. We will go through simple examples, as shown in Fig. 1 in order to specifically show the issues that we have tried to undertake. This simple ontology, which is adapted from one in [6], states

¹http://crowd.fi.uncoma.edu.ar/

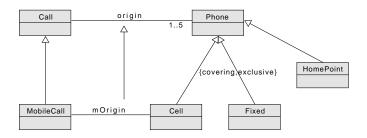


Figure 1. Starting graphical ontology in UML for scenarios and experiments, adapted from http://www.inf.unibz.it/~franconi/icom/

that mobile calls are a kind of calls; that phone points are partitioned between cell points and fixed points and that home points are among phone points. Mobile calls are related to cells through the mOrigin association. The binary association mOrigin is a included in the binary association origin. Finally, for a sake of simplicity, we will suppose for our example that each call has between one and five origin phones.

3.1. Scenario 1: Interoperability Support in Tools

For this scenario, we consider importing in a tool an OWL 2 ontology from a OWL 2 document in OWL/XML syntax aiming at visualising it, i.e. exploring its concepts, hierarchies and roles. Firstly, we present the case of Protégé for an ontology with two equivalent representations for the domains and ranges of object properties. Finally, we discuss and conclude about this issue and show an algorithm for importing OWL 2 ontologies, which aiming at accessing the OWL 2 documents through SPARQL-DL [18] queries on the ontology to be imported and visualised.

A possible encoding, following [2], for the binary association origin between Call and Phone from the conceptual model shown in Fig 1 is as follows:

From the resulting DL formalisation, we can generate document V1 in Table 1 using OWL/XML syntax. Alternatively, we can use ObjectPropertyDomain and ObjectPropertyRange OWL/XML tags, resulting in document V2. Intuitively, both documents represent the same graphical and formal meaning, but after importing V1 in Protégé, both domain and range axioms are imported as general axioms because the referred axioms cannot be directly associated with a named class. This is shown in Fig 2 (a). On other hand, importing of V2 is trivial because domains and ranges are displayed in the right view and associated to origin. Fig. 2 (b) shows its interpretation.

In Protégé, general class axioms cannot be directly associated with a named class so that they are not inferred as equivalent to domain and range of an ObjectProperty either when importing. Furthermore, reasoners in the tool do not conclude this after checking

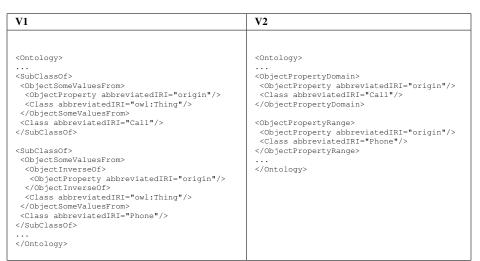


Table 1. Document (V1): OWL 2 encoding domain and range derived from Fig 1. Document (V2): Another OWL encoding for the same axioms but using ObjectPropertyDomain and ObjectPropertyRange tags. Both documents in OWL/XML syntax.

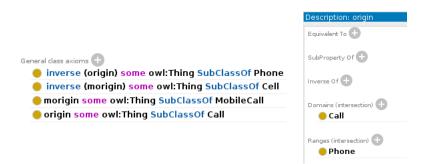


Figure 2. (a) left. Domain and range axioms according to V1: OWL 2 document are imported as general class axioms in Protégé. (b) right. Domain and range axioms according to V2: OWL 2 document are imported as domain and range at object property level in Protégé.

the ontology. In order to correct this situation, we propose to manipulate OWL documents through queries on the ontology instead of parsing OWL tags. To do this, we run a SPARQL-DL engine on the V1 document and the queries shown in Table 2. This engine is built on top of the Pellet reasoner [19] for SPARQL-DL [18], which is a substantial subset of SPARQL, significantly more expressive than existing DL QLs. Thus, SPARQL-DL outs the respective domain and range for origin independently of the underlying representations in a document, and disambiguating the general class axioms and returning the equivalents ones.

Discussion We can see that general axioms expressing domain and range for a property (as in the OWL document V1) can be properly represented in a visual tool when an ontology is being imported. Tools built on top of the OWL API [9] parse OWL documents by identifying each axiom in the document, i.e. travelling its structure. This is fine for tools manipulating DL axioms such as Protégé but not the case of visual tools. Even visual

SPARQL-DL Queries	Outputs
<pre>// Get Domain for each OP SELECT DISTINCT ?objectproperty ?domainop WHERE { ObjectProperty(?objectproperty), Domain(?objectproperty,?domainop) } // Get Range for each OP SELECT DISTINCT ?objectproperty ?rangeop WHERE { ObjectProperty(?objectproperty), Range(?objectproperty,?rangeop) }</pre>	<pre>{"domainop": { "type": "uri", "value": "http://crowd.fi.uncoma.edu.ar&Call" }, "objectproperty": { "type": "uri", "value": "http://crowd.fi.uncoma.edu.ar&origin"}} {"domainop": { "type": "uri", "value": "http://www.w3.org/2002/07/owl#Thing" }, "objectproperty": { "type": "uri", "value": "http://crowd.fi.uncoma.edu.ar&origin" }} {"rangeop": { "type": "uri", "value": "http://crowd.fi.uncoma.edu.ar#Phone" }, "objectproperty": { "type": "uri", "value": "http://crowd.fi.uncoma.edu.ar#Phone" }, "objectproperty": { "type": "uri", "value": "http://crowd.fi.uncoma.edu.ar#origin"}} {"rangeop": { "type": "uri", "value": "http://www.w3.org/2002/07/owl#Thing" }, "objectproperty": { "type": "uri", "value": "http://crowd.fi.uncoma.edu.ar#origin" }}</pre>

 Table 2. SPARQL-DL queries and responses for domains and ranges of each ObjectProperty in V1.

tools like OWLGrEd [5] and ICOM [6] hide these complex expressions behind UML classes.

	Algorithm 1 Importing ontologies in <i>crowd</i>
	Result: A visual representation of an ontology extracted from a OWL 2 document
1	$OWL-file \leftarrow load OWL 2$ document in <i>crowd</i> ;
2	\$answer ← execute SPARQL-DL queries on \$OWL-file;
3	$classes \leftarrow get Classes from \starswer;$ /* get all the classes from SPARQL-DL responses */
4	foreach \$class in \$classes do
5	$subclass \leftarrow subclass \cup getSubClass(sclass)$
6	end
7	$op-set \leftarrow get ObjectProperties from $answer;$
8	$triple-op \leftarrow void;$
9	foreach \$op in \$op-set do
10	$\texttt{$triple-op \leftarrow $triple-op \cup ($op, getDomain($op), getRange($op))$}$
11	end
12	$dp-set \leftarrow get DataProperties from answer;$
13	$triple-dp \leftarrow void;$
14	foreach \$dp in \$dp-set do
15	$\$triple-dp \leftarrow \$triple-dp \cup (\$dp, getDomain(\$dp), getRange(\$dp))$
16	end
	/* Composing the UML diagram: UML-D */
17	foreach \$class in \$classes do map \$class to a UML class in UML-D;
18	foreach \$sbc in \$subclass do map \$sbc to a UML generalisation in UML-D;
19	foreach (\$op,\$dop,\$rop) <i>in</i> \$triple-op do map \$op to a UML binary association between \$dop and \$rop with cardinality 0* on both sides in UML-D;
	foreach (\$dp,\$ddp,\$rdp) <i>in</i> \$triple-dp do map \$dp to a UML attribute between \$ddp and \$rdp in UML-D; send <i>UML-D</i> to graphical engine;

Aiming at providing an import functionality for a visual tool as *crowd*, we propose to query the ontologies, e.g. using SPARQL-DL and reasoning, and thus identifying axioms that can be mapped to graphical primitives. The pseudo-code in Alg. 1 shows the algorithm in *crowd* for importing OWL 2 ontologies in order to visualise them. The general idea of such algorithm is to get from an OWL 2 document all the classes, object and data properties and any other axioms, which primitives in the underlying graphical

language can be mapped to. For instance, Equivalence and Disjoint axioms do not have *direct* primitives in UML although they could be reconstructed from DL by adding generalisations and additional classes, as demonstrated in [2]. Those axioms without any primitive will remain saved in the source OWL 2 document for reasoning tasks. Lastly, in spite of the fact that the input of the algorithm depends on the SPARQL-DL engine in terms of complexity, the post processing of responses is lineal to the number of queries, i.e. m * n, where n are the queries and m their responses.

As one limitation, this approach only takes into account monolithic ontologies, i.e. ontologies saved into only one file, with no imports or mergers [7]. In this same direction, properties whose domain and range are defined as composed concepts could not be imported either.

3.2. Scenario 2: Inferring Stricter Cardinalities

Determining implicit consequences is helpful: on the one hand, to reduce the complexity of the diagram by removing those parts that implicitly follow from other ones, and on the other hand it can be used to make properties explicit, thus enhancing its readability. In particular, the refinement of properties involves detecting stricter cardinality constraints from analysing how the properties of classes and relationships interact with each other. From a theoretical point of view, these cardinalities can be identified by exploring the DL axioms (by hand), however, this is rather complex when involving external reasoning tools. Reasoning services provide ways to query about satisfiability, subsumption, equivalence and disjointness of classes, object and data properties, although in some situations these services should be used along with specific axioms in order to provide more insights on the current model.

In this scenario, we revisit the conceptual model from Fig. 1, identify some implicit cardinalities based on a related approach from ICOM, and present an algorithm for manipulating and inferring these cardinalities on visual models in a concrete tool. First of all, in order to analyse cardinalities, we encode both UML association origin and mOrigin with its respective cardinalities (axioms 1-5), and also the UML generalisations (axioms 6-9), as follows.

$\exists origin. \top \sqsubseteq$	Call	(1)	

$$\exists origin^-.\top \sqsubset Phone \tag{2}$$

$$Call \subseteq (\geq 1 \, origin. \top) \sqcap (\leq 5 \, origin. \top) \tag{3}$$

$$\exists mOrigin. \top \sqsubseteq MobileCall \tag{4}$$

$$\exists mOrigin^{-}.\top \sqsubseteq Cell \tag{5}$$

 $MobileCall \sqsubseteq Call \tag{6}$

$Phone \equiv Cell \sqcup Fixed \tag{7}$

- $HomePoint \sqsubseteq Phone \tag{8}$
 - $mOrigin \sqsubseteq origin \tag{9}$

After handing these axioms to Protégé and reasoning over it, no new hidden axiom is revealed to the modeller. However, if we carefully analyse the diagram, we can conclude that it is necessarily true that each MobileCall may have an origin from at most five Cell. This is true because any pair in mOrigin is also among the pairs in origin and also each Call participates at most five times as first argument in origin. So that any generic subclass of Call and origin inherits the very same maximum participation. One more time, current reasoning systems do not detect this new conclusion in a direct way but they can help to extract this implicit knowledge from ontology by supporting other ontology engineering processes. Based on the ICOM proposal, where non-trivial graphical cardinalities are inferred by asserting new axioms for 0 and 1 during the DL encoding of a diagram, we propose a new process for generalising the previous one and thus inferring cardinalities greater than ICOM ones. First of all, our algorithm asserts these axioms to original model:

$$\begin{split} MobileCall_mOrigin_min \equiv MobileCall \sqcap (\geq \ m \ mOrigin) \\ MobileCall_mOrigin_max \equiv MobileCall \sqcap (\leq \ n \ mOrigin) \end{split}$$

where m is the minimum cardinality and n is the maximum cardinality. For our model, the following axioms are asserted:

$$MobileCall_mOrigin_min \equiv MobileCall \sqcap (\geq 1 mOrigin)$$
(10)

$$MobileCall_mOrigin_max \equiv MobileCall \sqcap (\le 5 mOrigin)$$
(11)

Fig. 3 depicts this scenario. Initially, from axioms (10-11), it is possible state that both concepts $MobileCall_mOrigin_min$ and $MobileCall_mOrigin_max$ are sub-concepts of MobileCall because the sub-expressions ($\geq 1 mOrigin$) and ($\leq 5 mOrigin$) are empty sets, i.e. no stricter cardinality is defined for mOrigin. The resulting ontology after reasoning on the last model shows interesting conclusions. Particularly, it states that axiom 11 is necessarily true and as a consequence stating that the maximum participation for mOrigin is 5, as shown in Fig. 4. A simple trace of this result is:

$$MobileCall \sqsubseteq Call$$

 $mOrigin \sqsubseteq origin$
 $MobileCall_mOrigin_max \equiv MobileCall \sqcap (\leq 5 mOrigin)$

The general idea of this approach is to assert axioms such as (10) and (11) for each cardinality to be reviewed, choose the equivalents ones after reasoning and as a consequence, reveal the hidden constraints (if any). In this spite of the fact that the number of asserted axioms grows linearly on the number of binary association, it also depends on the minimum and maximum cardinality supported in the models, in which case the growth in the amount of axioms should be considered at the moment of evaluating the performance of a concrete tool. Still, we can consider this maximum cardinality number for all constraints in the model as a constant.

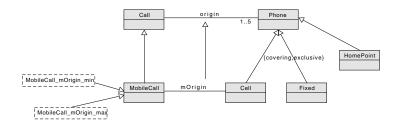


Figure 3. Original ontology showing how axioms (10) and (11), represented as classes, are related to the existing classes from which cardinalities will be analysed.

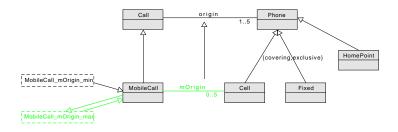


Figure 4. After reasoning on the previous graphical ontology in 3, one of such new classes, MobileCall_mOrigin_max is now equivalent to its related class indicating the new maximum cardinality for the association mOrigin.

Discussion Automated reasoning is key to ensure the quality of ontologies. Thus, exploiting it in depth in order to make impact on the visualisation of ontologies is an aspect relevant. In this case of study, we have analysed a feature not completely undertaken in existing tools. Only ICOM [6] proposed an initial approach for revealing cardinalities in conceptual models by taking advantages from automated reasoning in DL. However, its algorithm is limited to cardinalities between 0 and 1 and none comprehensive formal analysis of it has been yet reported.

In this work, we proposed a preliminary version of an enhanced algorithm, whose pseudo-code is sketched in Alg. 2 and currently implemented in our tool *crowd*. For a sake of space, we write the core of the algorithm leaving out other aspects about how stricter cardinalities are visualised back in a tool, which is a similar to the code in Alg. 1. The algorithm takes a UML input with *n* binary associations and a specific cardinality value, named *card*. After that, it asserts axioms as (10) and (11) for each cardinality value between 1 and *card*. Finally, when an answer returns from a reasoning system, the algorithm looks for axioms equivalent to domain and range of each association, which represent precise cardinality values. Under this approach, the number of new concepts is lineal to the number of associations in the input model, i.e. *4cardn*, being being *card* the allowed maximum cardinality and *n* the number of associations in the input model. This upper bound is acceptable considering middle-size models, and mainly it relies on the capabilities of the reasoning tool for processing a potentially huge number of concepts and/or roles.

```
Algorithm 2 Reasoning on graphical ontologies in crowd
    Result: A set of stricter cardinalities for each association in the graphical model
   diag \leftarrow Input UML graphical ontology in crowd with n associations;
 1
 2 foreach $class and $attribute in $diag do
 3
        $owlFile \leftarrow encodeOWL($class);
                                                                /* each class is encoded in OWL */
 4
        $owlFile \leftarrow encodeOWL($attribute);
 5 end
    /* $card: threshold of max cardinality to be checked
                                                                                                                  */
    /* the algorithm incorporates 4*\$card*n classes to the initial UML
                                                                                                                  */
 6 foreach $association in $diag do
        domain \leftarrow getDomain(association);
 7
 8
        range \leftarrow getRange(sassociation);
 9
        for i = 1 to $card do
10
             domain_min \leftarrow domain_min \cup domain_min_i \equiv getName(domain) \sqcap (\geq
                                                                                                                 i
               getName($association))};
11
              domain_max \leftarrow domain_max \cup \{domain_max_i \equiv getName(domain) \sqcap (\leq 
                                                                                                                 i
               getName($association))};
12
              \text{srange_min} \leftarrow \text{srange_min} \cup \{range_min_i \equiv \text{getName}(\text{srange}) \sqcap (\geq
                                                                                                                i
               getName($association)<sup>-</sup>)};
13
              \text{srange}_{max} \leftarrow \text{srange}_{max} \cup \{ range_{max_i} \equiv \text{getName}(\text{srange}) \sqcap (\leq 
                                                                                                                 i
               getName($association))<sup>-</sup>};
14
               with Cards \leftarrow 
15
        end
        $owlFile \leftarrow encodeOWL($association, $withCards);
16
17 end
18 $answer \leftarrow send $owlFile to a reasoning system;
19
   while not at end of this $answer document do
20
         foreach $domain_maxi in $domain_min) and ($domain_maxi in $domain_max) do
21
             if domain_{min_i} \equiv domain in answer then
22
                   dmin \leftarrow dmin \cup dmin_min_i
23
              end
24
             if domain_max_i \equiv domain in answer then
25
                  dmax \leftarrow dmax \cup dmain_max_i
26
             end
27
        end
28
        foreach $range and ($range_min<sub>i</sub> in $range_min) and ($range_max<sub>i</sub> in $range_max) do
29
             if \text{srange}_{\min_i} \equiv \text{srange} in \text{sanswer} then
30
                  rmin \leftarrow rmin \cup range_min_i
31
              end
32
             if \text{srange}_{max_i} \equiv \text{srange} in \text{sanswer} then
33
                  \operatorname{srmax} \leftarrow \operatorname{srmax} \cup \operatorname{srange}_{\operatorname{max}_i}
34
             end
35
        end
36
   end
   return \max(\{\$dmin\}) \cup \max(\{\$rmin\}) \cup \min(\{\$dmax\}) \cup \min(\{\$rmax\})
37
```

4. Conclusions

Automated reasoning is a powerful way to help to users in modelling tasks, and simple examples are enough to see that integrating reasoning to ontology engineering processes should be explored in more depth. Along this final discussion, we revisit our starting motivation and extend the findings of this work by detailing about specific dimensions related to the use of automated reasoning in engineering tasks, and the need of defining standard ways of providing them in the context of visual tools. About exploding the popular myth of interoperability Both described scenarios increase the myth of the interoperability. Firstly, equivalent ontologies are not interpreted as equivalents in Protégé but do in other tools querying the ontology differently, however, all of them are built on top of the same OWL API. With reference to the second scenario, the motivating issue is similar to the previous one. Approaches coming from the DL reconstructions of CMDLs in *classic* ontology engineering fail at the moment of visualising more expressive ontologies through standard languages. Non-standard languages could be able to depict complex axioms and implicit properties but not in a user-friendly way [14, 13]. These findings bring to light some not yet solved interoperability issues in current tools, while the proposed algorithms try to undertake these challenging features without being the cure-all for them.

Defining standardised importing and exporting processes in tools The lack of a more robust interoperability support in tools is closely related to the definition and standardisation of tools and some ontology engineering processes. Making focus on tools in the literature and particularly in visual ones, we see that both importing and exporting processes are typical read from and write to OWL documents. This is fine in environments where the content of these documents does not present semantic definitions. In particular, tools are not considering this when importing OWL 2 document, leaving the manipulation of the document to an API, which identify axioms according to OWL definitions. As a conclusion, the already discussed standardisation of ontology languages [11] should go hand in hand not only for the development of tools but also for the standardisation of the same and the ontology engineering processes [3].

About the use of automatic reasoning in visual tools Taking into account that the consequences of bad ontologies become more critical in real applications, we have to provide access to even more sophisticated tools for modelling ontologies in order to assure certain quality parameters. Visual aspects are fundamental in this sense because they help to modellers in detecting anti-patterns and debugging models. However, the increasing complexity of ontologies because of their size requires closely integrating both visual and automated reasoning in comprehensive engineering tasks. In this direction, we are working hard on *crowd*, which is a meta tool making use of graphical languages and reasoning in an environment for ontology engineering. Without being a silver bullet, *crowd* implements the algorithms presented in this work bringing together research on semantic formal and visual approaches for knowledge representation. Lastly, the tool is able to be interfaced to other ones (commonly, reasoners) for handing models to diverse systems that enabling possible different diagnoses and results.

As a conclusion, we present two novel algorithms derived from a number of experiments in related tools, which are implemented in a concrete visual tool, *crowd*, integrating automated reasoning in engineering tasks. These proposals aim to bring together research on two under-explored aspects to bad or good ontology design, particularly, techniques and tools for maintaining and manipulating good ontologies as well as explaining, and repairing bad ones.

References

[1] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

- [2] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 2005.
- [3] G. Braun. *Metodologías y Herramientas Visuals para Ingeniería Ontológica*. PhD thesis, Universidad Nacional del Sur, Argentina, 2019.
- [4] G. Braun, L. Cecchi, and P. Fillottrani. Integrating Graphical Support with Reasoning in a Methodology for Ontology Evolution. In *JOWO@IJCAI*, 2015.
- [5] K. Cerans, J. Ovcinnikova, R. Liepins, and A. Sprogis. Advanced OWL 2.0 Ontology Visualization in OWLGrEd. In DB&IS, 2012.
- [6] P. Fillottrani, E. Franconi, and S. Tessaris. The ICOM 3.0 intelligent conceptual modelling tool and methodology. *Semantic Web Journal*, 2012.
- [7] P. Fillottrani and C. M. Keet. Dimensions Affecting Representation Styles in Ontologies. In *Knowledge Graphs and Semantic Web*, 2019.
- [8] G. Gottlob, A. Pieris, and E.I Sallinger. Vadalog: Recent advances and applications. In *JELIA*. Springer, 2019.
- [9] M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal*, 2011.
- [10] M. Horridge, R. Gonçalves, C. Nyulas, T. Tudorache, and M. Musen. Webprotégé: A cloud-based ontology editor. In *Companion of The 2019 World Wide Web Conference*, 2019.
- [11] I. Horrocks. Tool Support for Ontology Engineering. In Foundations for the Web of Information and Services - A Review of 20 Years of Semantic Web Research., 2011.
- [12] H. Knublauch, R. Fergerson, N. Noy, and M. Musen. The Protégé OWL plugin: An Open Development Environment for Semantic Web Applications. In *The Semantic Web – ISWC 2004*, 2004.
- [13] D. Lembo, D. Pantaleone, V. Santarelli, and D. Fabio Savo. Drawing OWL 2 ontologies with eddy the editor. AI Communication., 2018.
- [14] S. Lohmann, S. Negru, F. Haag, and T. Ertl. Visualizing ontologies with VOWL. *Semantic Web Journal*, 2016.
- [15] J. Moreira, T. Sales, J. Guerson, B. Braga, F. Brasileiro, and V. Sobral. Menthor Editor: An Ontology-Driven Conceptual Modeling Platform. In *JOWO@FOIS*. CEUR-WS.org, 2016.
- [16] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, editors. OWL 2 Web Ontology Language Profiles. W3C, second edition edition, 2012. https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/.
- [17] D. Nardi and R. J. Brachman. The description logic handbook. chapter An Introduction to Description Logics, pages 1–40. Cambridge University Press, 2003.
- [18] E. Sirin and B. Parsia. SPARQL-DL: SPARQL Query for OWL-DL. In OWLED, 2007.
- [19] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A Practical OWL-DL Reasoner. *Journal of Web Semantics*, 2007.
- [20] F. Sportelli. NORMA: A Software for Intelligent Conceptual Modeling. In JOWO@FOIS. CEUR-WS.org, 2016.
- [21] M. Vigo, S. Bail, C. Jay, and R. Stevens. Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design. *Journal of Human Computer Studies*, 2014.
- [22] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-Based Data Access: A Survey. In *IJCAI*, 2018.