Utility and Feasibility of Reasoning beyond Decidability in Semantic Technologies

Sebastian Rudolph and Michael Schneider

Institute AIFB, Karlsruhe Institute of Technology, DE
rudolph@kit.edu
FZI Research Center for Information Technology, Karlsruhe, DE
schneid@fzi.de

Abstract. Semantic Web knowledge representation standards such as RDF and OWL have gained momentum in the last years and are widely applied today. In the course of the standardization process of these and other knowledge representation formalisms, decidability of logical entailment has often been advocated as a central design criterion. On the other hand, restricting to decidable formalisms inevitably comes with constraints in terms of modeling power. Therefore, in this paper, we examine the requirement of decidability and weigh its importance in different scenarios. Subsequently, we discuss a way to establish incomplete – yet useful – reasoning support for undecidable formalisms by deploying machinery from the successful domain of theorem proving in first-order predicate logic. While elaborating on the undecidable variants of the ontology language OWL 2 as our primary examples, we argue that this approach could likewise serve as a role model for knowledge representation formalisms from the Conceptual Structures community.

1 Introduction

Today, the Semantic Web serves as the primary testbed for practical application of knowledge representation. A plethora of formalisms for representing and reasoning with Web knowledge has been designed and standardized under the auspices of the World Wide Web Consortium (W3C). While the early days of this endeavor saw ad-hoc and semantically underspecified approaches, interoperability requirements enforced their evolution into mature logical languages with clearly specified formal semantics. In the process of defining more and more expressive such formalisms, an often-debated requirement is decidability of logical entailment, i.e. the principled existence of an algorithm that decides whether a body of knowledge has a certain proposition as a consequence. While it goes without saying that such an algorithm is clearly useful for all kind of querying or knowledge management tasks, results established back in the 1930s show

that this property does not hold for all types of logics [6, 24]. In particular, in many expressive knowledge representation formalisms (most notably first-order predicate logic), entailment is undecidable.

Hence, whenever a knowledge representation formalism is to be designed, the trade-off between decidability and expressivity has to be taken into account. An examination of the Semantic Web languages hitherto standardized by the W3C yields a mixed picture in that respect: logical entailment in the basic data description language RDF [13] and its lightweight terminological extension RDF Schema [4] is decidable (although already NP-complete in both cases). Within the OWL 2 language family [17], only the most expressive variant OWL 2 Full [21] is undecidable, whereas OWL 2 DL [16, 15] as well as its specified sublanguages (called tractable profiles) OWL 2 EL, OWL 2 QL, OWL 2 RL [14] are decidable (the latter three even in polynomial time). On the other hand, for the rule interchange format RIF [12], only the very elementary core dialect RIF-Core [2] is decidable whereas already the basic logic dialect RIF-BLD [3] – and hence every prospective extension of it – turns out to be undecidable.

This small survey already shows that decidability is far from being a common feature of the standardized Semantic Web languages. However, extensive reasoning and knowledge engineering support is currently only available for the decidable languages in the form of RDF(S) triple stores or OWL 2 DL reasoners hinting at a clear practitioners' focus on these languages.

In this paper, we argue that inferencing support is important and feasible also in formalisms which are undecidable and we provide an outlook how this can be achieved, referring to our recent work on reasoning in undecidable Semantic Web languages as a showcase. We proceed as follows: Section 2 will remind the reader of the important notions from theoretical computer science. Section 3 proposes a schematic classification of inferencing algorithms by their practical usefulness. Section 4 distinguishes cases where decidability is crucial to enable "failsafe" reasoning from cases where it may make sense to trade decidability for expressivity. After these general considerations, we turn to variants of OWL to demonstrate ways to provide reasoning support for undecidable Semantic Web formalisms. To this end, Section 5 gives an overview of OWL syntaxes and the associated semantics. Section 6 shows two different ways of translating OWL reasoning problems into first-order logic and Section 7 briefly reports on our recent work of employing FOL reasoners in that context. In Section 8, we discuss ramifications of our ideas for Common Logic. Section 9 con-

cludes. An extended version of this paper with examples of reasoning in diverse undecidable languages is available as technical report [20].

2 Recap: Decidability and Semidecidability

Let us first recap some basic notions from theoretical computer science which are essential for our considerations. From an abstract viewpoint, a logic is just a (possibly infinite) set of *sentences*. The *syntax* of the logic defines how these sentences look like. The *semantics* of the logic is captured by an *entailment relation* $\models$ between sets of sentences Φ and sentences φ of the logic. $\Phi \models \varphi$ then means that Φ logically entails φ or that φ is a logical consequence of Φ. Usually, the logical entailment relation is defined in a model-theoretic way.

A logic is said to have a *decidable* entailment problem (often this wording is shortened as to calling the logic itself decidable – we will adopt this common practice in the following) if there is an algorithm which, taking as an input a finite set $\Phi = \{\phi_1, \ldots, \phi_n\}$ of sentences of that logic and a further sentence φ, always terminates and provides the output *YES* iff $\Phi \models \varphi$ or *NO* iff $\Phi \not\models \varphi$. As a standard example for a decidable logic, propositional logic is often mentioned, a straightforward decision procedure being based on truth tables. However, there are decidable logics of much higher expressivity, e.g., the guarded fragment of first-order logic [1].

A logic is said to have a *semidecidable* entailment problem (also here, this can be abbreviated by calling the logic itself semidecidable) if there exists an algorithm that, again given Φ and φ as input, terminates and provides the output *YES* iff $\Phi \models \varphi$, but may not terminate otherwise. Consequently, such an algorithm is complete in the following sense: every consequence will eventually be identified as such. However the algorithm cannot be used to get a guarantee that a certain sentence is *not* a consequence. Clearly, every decidable logic is also semidecidable, yet, the converse does not hold. The prototypical example for a logic that is semidecidable but not decidable is first-oder predicate logic (FOL). While undecidability of FOL can be shown by encoding the halting problem of a Turing machine into a FOL entailment problem, its semidecidability is a consequence from the fact that there exists a sound and complete deduction calculus for FOL [7], hence every consequence can be found in finite time by a breadth-first search in the space of all proofs w.r.t. that deduction calculus. Clearly, today's first-order theorem provers use much

more elaborated and goal-directed strategies to find a proof of a given entailment.

Obviously, in semi-decidable logics, the critical task which cannot be completely solved, is to detect the non-entailment $\Phi \not\models \varphi$. In model-theoretically defined logics such as FOL, $\Phi \not\models \varphi$ means that there exists a model $\mathcal{M}$ of Φ that is not a model of φ. In other words, finding such a model means proving the above non-entailment. Indeed, there are rather effective (yet incomplete) FOL model finders available dedicated to this purpose. For straightforward reasons, most of these model finders focus on finite models. In fact, if there is a finite model with the wanted property, it is always possible to find it (due to the reason that the set of finite models is enumerable and first-order model checking is easy). Hence, the case which is intrinsically hard to automatically detect is when $\Phi \not\models \varphi$ but every model of Φ that is not a model of φ has *infinite* size. While seemingly exotic at first sight, such cases exist and are not very hard to construct. An example for such a situation is the question whether $\varphi = \exists x.(p(x,x))$ is a logical consequence of $\Phi = \{\varphi_1, \varphi_2\}$ with $\varphi_1 = \forall x.\exists y.(p(x,y))$ and $\varphi_2 = \forall x \forall y \forall z.(p(x,y) \wedge p(y,z) \rightarrow p(x,z))$. In this example, φ_1 enforces that in every model of Φ every element must be in a p-relationship to something, whereas φ_2 requires that p must be interpreted by a transitive relation. A side effect of p's transitivity is that whenever a model contains a p-cycle, all the elements in that cycle are p-related to themselves which makes φ satisfied. Therefore, any model of Φ that does not satisfy φ must be p-cycle-free which is only possible if the model is infinite. The problem with infinite models is that, even if one has a method to represent and refer to them somehow, the set of all infinite models cannot fully be enumerated. Hence, whatever enumeration strategy is used, it will only cover a strict subset of all possible models.

3 A Classification of Decision Procedures by Usefulness

We will now take a closer look on the question how useful a sound and complete decision algorithm may be in practice. For the sake of better presentation, assume that we consider not all the (infinitely many) possible inputs to the algorithm but only the finitely many (denoted by the set P) below a fixed size s. Let us furthermore make the very simplifying assumption that every entailment problem in P is equally "important" in the sense that it will be posed to our reasoning algorithm with the same probability as the other problems. Now, a decision algorithm $\mathcal{A}$ comes with the guarantee, that it terminates on each of the inputs after finite

time, hence it gives rise to a function $\mathsf{runtime}_{\mathcal{A}} : P \to \mathbb{R}^+$ assigning to each of the inputs the corresponding (finite) runtime of the algorithm. Now, the algorithm can be described by a *characteristic curve* assigning to a time span Δt the fraction of elements from P on which $\mathcal{A}$ terminates after time less or equal to Δt. Formally, this function $\mathsf{char}_{\mathcal{A}} : \mathbb{R}^+ \to [0, 1]$ would be defined by

$$\mathsf{char}_{\mathcal{A}}(\Delta t) = \frac{\{p \in P \mid \mathsf{runtime}_{\mathcal{A}}(p) \leq \Delta t\}}{|P|}.$$

Figure 1 schematically displays characteristic curves which may be encountered for complete decision procedures. As a common feature, note that the curves are always monotonic by definition. Moreover, every such curve will hit the 100% line at some time point due to the facts that we have a complete decision algorithm and P is finite.

We now assume that the decision algorithm is to be employed in a practical scenario, which gives rise to the following specific figures:

- a maximal time span that is worth to be spent on the computation of an answer to an entailment problem of size s, referred to as *acceptable waiting time*;
- a ratio characterizing the probability of getting an answer below which a use of the algorithm will be considered worthless, called the *perceived added value threshold*; and
- a ratio characterizing the probability of getting an answer above which the algorithm can be considered practically reliable, called the *acceptable reliability threshold*.

Figure 1 also depicts these values, according to which the four schematic characteristic curves can now be coarsely distinguished in terms of practical usefulness.

- In the ideal case, the maximal runtime of the algorithm is smaller than the acceptable waiting time. Then every size s problem is guaranteed to be decided within the available time span, which allows for calling the algorithm *failsafe*.
- If this guarantee cannot be given, yet the probability that a solution will be obtained in the available time lies above the acceptable reliability threshold, the algorithm can still be said to be (practically) *reliable* and may be used within regular and automated knowledge management work flows. Then the rare cases not being covered could be dealt with by a kind of controlled exception handling mechanism.

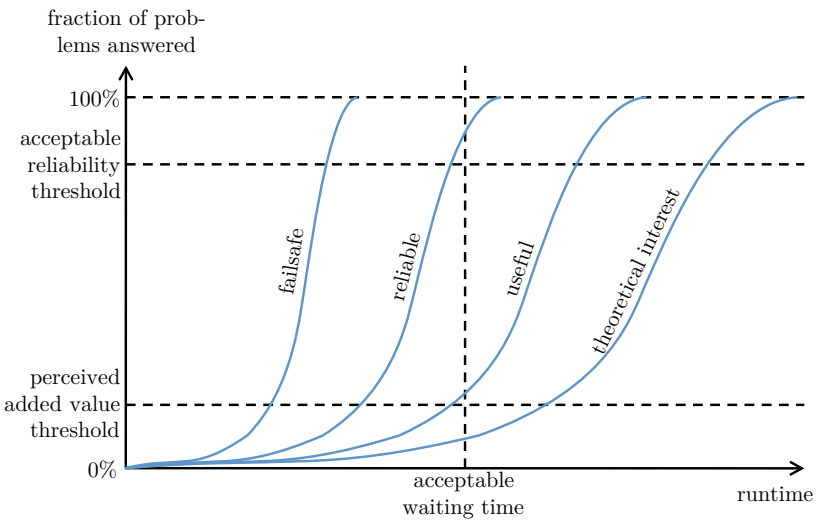

Fig. 1. Schematic representation of different characteristic curves for complete decision algorithms.

- If the expected frequency of termination within the available time span is below the acceptable reliability threshold but above the perceived added value threshold, the algorithm's output should be perceived as nice-to-have additional information which may be taken into account if available. The overall work flow in which it is to be used should not crucially rely on this, yet we can still see that the algorithm may be rightfully called *useful*.
- Finally, if the ratio of the timely obtainable answers is even below the perceived added value threshold, the algorithm is of little or no practical value. Note however, that the existence of a complete decision algorithm – even if it happens to lie within this class – is still a research question worthwhile pursuing since optimizations and hardware improvements may well turn such an algorithm into something practically useful. Conversely, the proven non-existence of a decision algorithm may prevent many ambitious researchers from vainly trying to establish one. This justifies to at least characterize this type of algorithm as of *theoretical interest*. For a prototypical example of this kind see [19].

Now, turning our attention to incomplete decision algorithms, we can assign to them characteristic curves in the same way as to complete ones

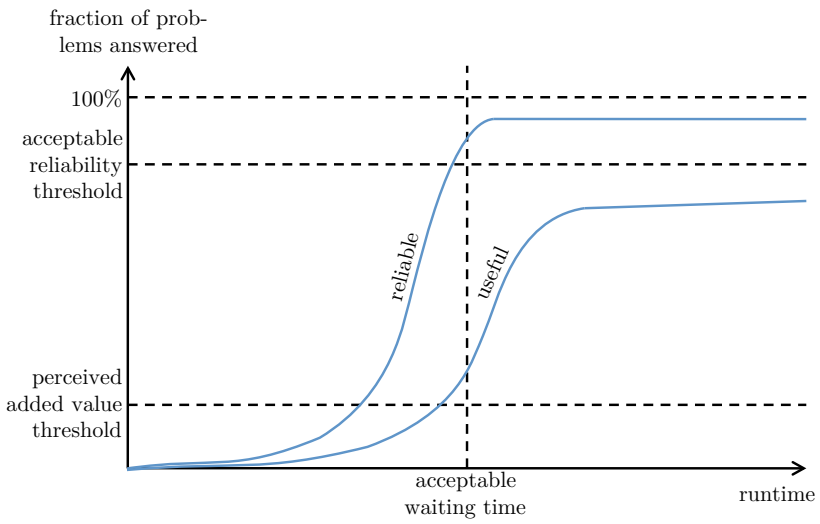

Fig. 2. Sketches of characteristic curves for incomplete decision procedures of practical interest.

(extending the range of $\mathsf{runtime}_{\mathcal{A}}$ to $\mathbb{R}^+ \cup \{\infty\}$). The only difference here is, that the curve will not hit the 100% line. Nevertheless, as Fig. 2 illustrates, there may be incomplete algorithms still satisfying the usefulness or even the reliability criterion defined above leading to the conclusion that there may be cases where the practically relevant behavior of incomplete decision algorithms is just as good as that of complete ones, the notable exception being that no failsafe behavior can be obtained. It should be noted here that inferencing in expressive decidable formalisms comes with high worst-case complexities (normally ExpTime and higher, for instance N2ExpTime for OWL 2 DL) which implies that there are "malicious" inputs of already small size for which the runtime exceeds any reasonable waiting time. Although the runtime on average cases is usually much better, this fact normally vitiates failsafety guarantees. Of course, the case is different for so-called tractable languages which are of time-polynomial or even lower complexity.

4 On the Importance of Failsafe Decision Procedures

Having identified the possible existence of a failsafe decision procedure as the only principled practical advantage that the use of a decidable formalism may have, let us now investigate under which circumstances

such a procedure is strictly needed and in what cases it is dispensable. In the following, we will identify general criteria for this assessment.

4.1 Where Failsafe Decidability is Crucial

Failsafe decidability is important in situations where automated reasoning is a central part of the functionality of a knowledge-based system and the questions posed to the system are normally "yes-or-no" questions of a symmetric form, the answers to which are to be used to trigger different follow-up actions in a highly or purely automated system ("if yes, then do this, otherwise do that").

One important case of this is when entailment checks are to be carried out in "closed-world situations," that is, when one can assume that all necessary information about a state of affairs has been collected in a knowledge base such that the non-entailment of a queried proposition justifies the assumption that the situation expressed by that proposition does indeed not hold. Under these circumstances, both possible answers to an entailment check provide information about the real state of affairs.

In other cases, the entailment check may be aimed at finding out facts about the considered knowledge base itself, instead of the real-world domain that it describes. In fact, this can be seen as a particular closed-world situation. As an example for this, consider the reasoning task of classification, i.e. the computation of a conceptual hierarchy. Here, the typical reasoning task to be performed is to answer the question "are two given classes in a subsumption relationship, or not?" This is a common task in today's ontology management systems, useful both for supporting human ontology engineers and speeding up further reasoning tasks.

4.2 Where Failsafe Decidability is Dispensable

Yet, there are also scenarios, where failsafe decision procedures for logical entailment seem to be of less importance.

First of all, this is the case when the emphasis of the usage of knowledge representation is on modeling rather than on automated inferencing. In fact, there will often be situations where logical languages are just used for noting down specifications in an unambiguous way and probably making use of available tool support for modeling and navigating these specifications. Clearly, in these cases, no reasoning support whatsoever is required, let alone failsafe decision procedures.

In other cases, reasoning may still be required, yet the available knowledge is assumed to be sound but incomplete w.r.t. the described domain,

i.e. we have an "open-world situation." Then, non-entailments cannot be conceived as guarantees for non-validity in the domain, as opposed to entailments, which (given that the knowledge base is sound) ensure validity. As a consequence thereof, non-entailment information is of less immediate value and an entailment checker just notifying the user of established entailments may be sufficient.

A more concrete case where failsafe decidability will often not be a strict requirement is human modeling support that will alarm the knowledge engineer upon the detection of inconsistencies in the knowledge base. (Note that detecting if a knowledge base KB is inconsistent is equivalent to checking the entailment $KB \models false$.) In practical ontology engineering, one certainly wants to make sure that a created ontology is free of semantic errors, and therefore good reasoning support for the detection of inconsistencies should be available. For many application domains, it will then be sufficient if such inconsistency checking does not find an issue after some considerable time of searching, while true consistency confirmation will only be nice to have.

Query answering scenarios represent a further type of setting normally not requiring failsafe decision procedures. When doing query answering, as e.g. in SPARQL, one has, in the simplest case, one or more axioms containing variables where otherwise individuals or class expressions would occur. One asks for all solutions that semantically follow from the queried knowledge base and match the query pattern. In this scenario, one is not interested in non-solutions, but in an *enumeration* of solutions. Therefore, what one needs is an enumeration algorithm, which does not really require a complete decision procedure. Also, from the practical viewpoint, in many applications such as search, completeness of the presented set of solutions is less important than when just one entailment is to be checked. What normally matters is that *enough* solutions are provided and that *relevant* solutions are among them, relevance being a measure that has to be defined for the specific purpose.

5 OWL, Syntactic and Semantic Variants

After these abstract considerations we will turn to OWL 2 as a specific knowledge representation formalism where the aforementioned issues are of particular interest, since both decidable and undecidable versions exist as well as well-investigated reasoning approaches for either. We will particularly focus on approaches for reasoning in undecidable formalisms.

First, let us recap the main aspects of syntax and semantics of OWL 2 to the degree needed for our considerations. When dealing with the Web Ontology Language OWL, one has to distinguish between two representation strategies which are interrelated but not fully interchangeable. The so called *functional syntax* [16] emphasizes the formula structure of what is said in the logic. As an example, consider the fact that an individual characterized as "happy cat owner" indeed owns some cat and everything he or she cares for is healthy. Expressed in OWL functional syntax, this statement would look as follows:

```
SubClassOf(
   ex:HappyCatOwner
   ObjectIntersectionOf(
     ObjectSomeValuesFrom( ex:owns ex:Cat )
     ObjectAllValuesFrom( ex:caresFor ex:Healthy ) ) )
```

On the other hand, there is the RDF syntax [18] which expresses all information in a graph (which in turn is usually encoded as a set of vertice-edge-vertice triples labeled with so-called Uniform Resource Identifiers, short: URIs). The above proposition in RDF representation would

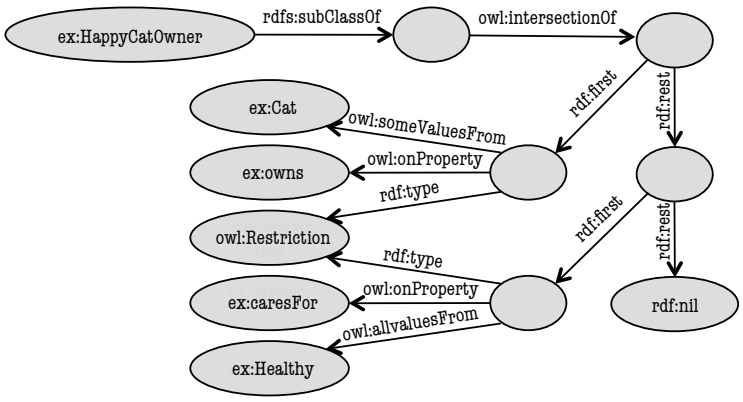

Fig. 3. RDF representation as graph.

look as displayed in Fig. 3. As opposed to the XML-based syntax often used for machine processing, the so-called Turtle syntax better reflects the

triple structure, yet allows for abbreviations for structural bnodes[1] and list structures. The Turtle representation of the RDF graph from Fig. 3 would look as follows:

```
ex:HappyCatOwner rdfs:subClassOf
   [ owl:intersectionOf
     ( [ rdf:type owl:Restriction ;
         owl:onProperty ex:owns ;
         owl:someValuesFrom ex:Cat ]
       [ rdf:type owl:Restriction ;
         owl:onProperty ex:caresFor ;
         owl:allValuesFrom ex:Healthy ] ) ] .
```

Clearly, every functional syntax ontology description can be transformed into RDF representation. The converse is, however, not always the case. For all ontologies expressible in functional syntax, the specification provides the so-called *direct semantics* [15]. This semantics is very close to the extensional semantics commonly used in description logics: an interpretation is defined by choosing a set as domain, URIs denoting individuals are mapped to elements of that domain, class URIs to subsets and property URIs to sets of pairs of domain elements. It is noteworthy that the class of ontologies expressible via the functional syntax is not decidable, but only a subclass of it where further, so-called *global restrictions* apply. It is this decidable class which is referred to as OWL 2 DL and for which comprehensive tool support is available both in terms of ontology management, infrastructure and inferencing (software implementing decision procedures for entailment are typically called *reasoners*).

On the other hand, there is a formal semantics applicable to arbitrary RDF graphs. This so-called *RDF-based semantics* [21] is more in the spirit of the semantics of RDF(S): all URIs are mapped to domain elements in the first place and might be further interpreted through an extension function.

The two different semantics are related. A *correspondence theorem* [21] ensures that, given certain conditions which are easy to establish, the direct semantics on an ontology in functional syntax (taking into account the global restrictions of OWL 2 DL) will only provide conclusions that the RDF-based semantics provides from the ontology's RDF counterpart as well. However, the RDF-based semantics will often provide additional conclusions that are not provided by the direct semantics, for instance

[1] Bnodes, also called blank nodes, are unlabeled vertices in the RDF graph representation and often used as auxiliary elements to encode complex structures.

conclusions based on metamodeling. The difference between the two semantics becomes significant for those ontologies that are still covered by the direct semantics but are not OWL 2 DL ontologies, i.e. ontologies that are beyond the scope of the correspondence theorem. Furthermore, there are ontologies to which only the RDF-based semantics applies but not the direct semantics, since not every RDF graph can be translated into an ontology in functional syntax. Figure 4 summarizes the relationships just stated.

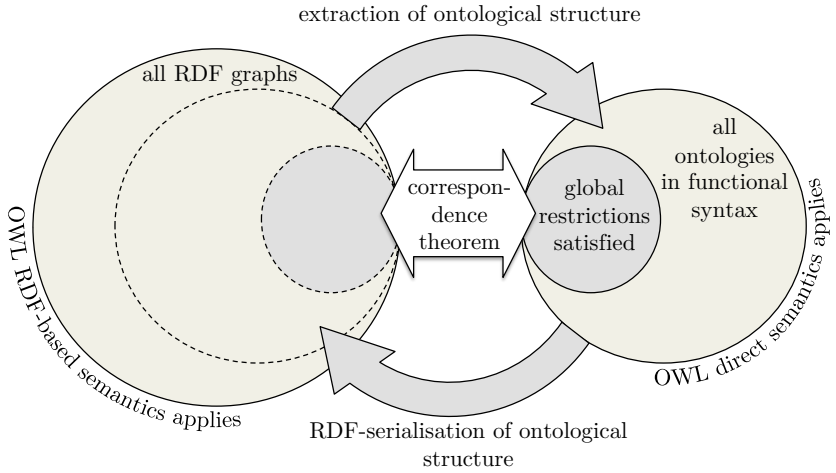

Fig. 4. Syntaxes and semantics for OWL 2.

6 FOL-Empowered OWL Reasoning beyond Decidability

As stated before, for the decidable fraction of OWL 2, decision procedures have been implemented and are widely used today. Here, we will focus on the problem of providing practical inferencing support for the more expressive yet undecidable versions of OWL. Thereby, we want to question an attitude sometimes encountered in the OWL community according to which the quest for automated inferencing beyond decidability is futile.

It seems in place to take a look at the prototypical example of undecidability: first-order predicate logic. Despite the fact that its undecidability was proven way before the wide-spread adoption of computers, automated first-order logic reasoning has always been a focus of interest and is mean-

while well-established also in practical applications such as hard- and software verification [26]. There are many FOL theorem provers which are able to find out when a set of FOL formulas is unsatisfiable or implies another set of formulas. Even model-finding is well supported, although typically restricted to finding finite models.[2]

In view of the success of FOL-based inferencing, which also substantiates our claim that reasoning in undecidable formalisms can be useful, it seems tempting to harness the comprehensive theoretical work and highly optimized implementations originating from that line of research for coping with inferencing problems in the undecidable variants of OWL. As it turns out, OWL inferencing with respect to both syntaxes and according semantics admits translation into FOL entailment problems. In the following, we will describe these embeddings in a bit more detail.

6.1 Translating Direct Semantics Reasoning into FOL

Given an ontology in functional syntax, the translation into FOL can be performed along the lines of the standard translation of description logics into FOL as, e.g., described in [9]. For instance, the ontology axiom presented in Section 5 would be translated into the following FOL sentence:

$$\forall x.\mathrm{HappyCatOwner}(x) \rightarrow \\ \exists y.(\mathrm{owns}(x,y) \wedge \mathrm{Cat}(y)) \wedge \forall z.(\mathrm{caresFor}(x,z) \rightarrow \mathrm{Healthy}(z))$$

We see that URIs referring to classes are translated into unary predicates while those denoting properties are mapped to binary predicates. The existential and the universal property restrictions have been translated into an existentially and a universally quantified subformula, respectively. The two subformulas are connected by a conjunction symbol, which is the result of translating the intersection class expression. Finally, the outermost class subsumption axiom has been translated into a universally quantified implication formula.

After this transformation, reasoning can be performed by means of FOL reasoners: FOL theorem provers can be used to find entailments and to detect inconsistent ontologies, whereas FOL model finders can be used to detect non-entailment and to confirm consistency of an ontology. In fact this approach is not new but has already been demonstrated in [23].

[2] For a comprehensive collection of online-testable state-of-the-art provers and model finders, we refer the reader to `http://www.cs.miami.edu/~tptp/cgi-bin/SystemOnTPTP`.

In general, one can use this approach to embed OWL 2 DL into arbitrary subsets of FOL. This includes reasoning in the OWL 2 direct semantics beyond OWL 2 DL, i.e., reasoning with ontologies that are given in the functional syntax but are not constrained by any of the global restrictions of OWL 2 DL. This relaxation allows, for example, to recognize circular relationships such as cyclic chemical molecules, a feature that has often been asked for but is not available in OWL 2 DL due to its syntactic restrictiveness (see e.g. [25]). This strategy further covers many well-known undecidable extensions of OWL such as the semantic web rules language SWRL [10] as well as the combination of the rule interchange dialect RIF BLD with OWL described in [5].

6.2 Translating RDF-Based Semantics Reasoning into FOL

Under the RDF-based semantics, a translation into FOL is also possible but works differently than for the direct semantics. The RDF graph representation of the example ontology is translated into a conjunction of ternary atomic FOL formulas, which correspond to the different RDF triples in the RDF graph:

$$
\begin{aligned}
&\exists b_0, b_1, b_2, b_3, b_4.(\\
&\quad \text{iext}(\texttt{rdfs:subClassOf}, \texttt{ex:HappyCatOwner}, b_0) \\
&\wedge\ \text{iext}(\texttt{owl:intersectionOf}, b_0, b_1) \\
&\wedge\ \text{iext}(\texttt{rdf:first}, b_1, b_3) \\
&\wedge\ \text{iext}(\texttt{rdf:rest}, b_1, b_2) \\
&\wedge\ \text{iext}(\texttt{rdf:first}, b_2, b_4) \\
&\wedge\ \text{iext}(\texttt{rdf:rest}, b_2, \texttt{rdf:nil}) \\
&\wedge\ \text{iext}(\texttt{rdf:type}, b_3, \texttt{owl:Restriction}) \\
&\wedge\ \text{iext}(\texttt{owl:onProperty}, b_3, \texttt{ex:owns}) \\
&\wedge\ \text{iext}(\texttt{owl:someValuesFrom}, b_3, \texttt{ex:Cat}) \\
&\wedge\ \text{iext}(\texttt{rdf:type}, b_4, \texttt{owl:Restriction}) \\
&\wedge\ \text{iext}(\texttt{owl:onProperty}, b_4, \texttt{ex:caresFor}) \\
&\wedge\ \text{iext}(\texttt{owl:allValuesFrom}, b_4, \texttt{ex:Healthy}))
\end{aligned}
$$

All atoms are built from a single FOL predicate 'iext', which corresponds to the function 'IEXT(.)' used in the RDF-based semantics to represent *property extensions*. Terms within the atoms are either constants or existentially quantified variables, which correspond to the URIs and the blank nodes in the RDF triples, respectively.

The above formula only represents the RDF graph itself without its meaning as an OWL 2 Full ontology. The OWL 2 Full meaning of an

RDF graph is primarily defined by the collection of model-theoretic *semantic conditions* that underly the RDF-based semantics. The semantic conditions add meaning to the terms being used in the graph, such as '`rdfs:subClassOf`', and by this means put constraints on the use of the 'IEXT(.)' function. The semantic conditions have the form of first-order formulas and can therefore be directly translated into FOL formulas. For example, the FOL representation for the semantic condition about the term '`rdfs:subClassOf`' [21, Sec. 5.8] has the form:

$$\begin{aligned}\forall c_1, c_2.(\, &\mathrm{iext}(\texttt{rdfs:subClassOf}, c_1, c_2) \leftrightarrow \\ &\mathrm{iext}(\texttt{rdf:type}, c_1, \texttt{owl:Class}) \\ \wedge \, &\mathrm{iext}(\texttt{rdf:type}, c_2, \texttt{owl:Class}) \\ \wedge \, &\forall x.(\, \mathrm{iext}(\texttt{rdf:type}, x, c_1) \rightarrow \mathrm{iext}(\texttt{rdf:type}, x, c_2) \,) \,)\end{aligned}$$

The RDF-based semantics consists of several hundred semantic conditions. The meaning of the example ontology is given by the FOL translation of the actual RDF graph plus the FOL translations of all the semantic conditions of the RDF-based semantics.

While the semantic conditions of the RDF-based semantics only quantify over elements of the domain, a restricted form of higher-order logic (HOL) is provided by both the syntax and semantics of OWL 2 Full, without however the full semantic expressivity of HOL. Nevertheless, ontologies with flexible metamodeling are supported and some useful HOL-style reasoning results can be obtained.

7 Experimental Results

In previous work, we have translated the OWL 2 RDF-based semantics into a FOL theory and done a series of reasoning experiments using FOL reasoners, which generally indicate that rather complicated reasoning beyond OWL 2 DL is possible. These experiments included reasoning based on metamodeling as well as on syntactic constellations that violate the global restrictions of OWL 2 DL. The used FOL reasoners generally succeeded on most or all of the executed tests, and even often did so considerably fast, while the state-of-the-art OWL 2 DL reasoners that were used for comparison only succeeded on a small fraction of the tests. However, one often observed problem of the FOL reasoners needs to be mentioned: while they were very successful in solving complicated problems, they often showed difficulties on large input sizes. In the future, we will have to further investigate how to cope with this scalability issue. Our results have been described in [22].

In addition, we have recently conducted some initial experiments concerning reasoning in the direct semantics beyond OWL 2 DL, including positive and negative entailment checking based on cyclic relationships. Again, we have found that FOL reasoners can often quickly solve such problems, provided that the input ontologies are of reasonable size.

8 Come on, Logic!

Common Logic (CL) [11] has been proposed as a unifying approach to different knowledge representation formalisms, including several of the formalisms currently deployed on the Web. Semantically, CL is firmly rooted in FOL[3], whereas its syntax has been designed in a way that accounts for the open spirit of the Web by avoiding syntactic constraints typically occurring in FOL, such as fixed arities of predicates, or the strict distinction between predicate symbols and terms.

The syntactic freedom that CL provides allows for flexible modeling in a higher-order logic (HOL) style and it is even possible to receive some useful HOL-style semantic results, similar to metamodeling in OWL 2 Full. In fact, OWL 2 Full can be translated into CL in an even more natural way than into standard FOL, as demonstrated by Hayes [8]. Furthermore, CL allows to represent the OWL 2 Direct Semantics, as well as other Semantic Web formalisms, such as SWRL and RIF. Compared to all these Semantic Web formalisms, CL is significantly more expressive and flexible and, therefore, users of CL should benefit from extended modeling and reasoning capabilities.

By becoming an ISO standard, CL has taken another dissemination path than the commonly adopted Semantic Web languages. While syntax and semantics of CL are well-defined, software support for editing and managing CL knowledge bases has just started to be developed and support for reasoning in CL is close to non-existent to the best of our knowledge. This arguably constitutes the major obstacle for wide deployment – as we have argued in the preceding section, the often criticized undecidability of CL does not qualify as a sufficient reason to dismiss this formalism right away.

While the development of scalable, ergonomic tools is certainly an endeavor that should not be underestimated, the above presented approach provides a clear strategy for accomplishing readily available state-of-the-art reasoning support. In fact, since the translation of CL into FOL is

[3] Strictly speaking, in order to keep within FOL, one has to avoid the use of so called *sequence markers*.

even more direct than for the two considered variants of OWL, creating a reasoning back-end for CL should be even more straight-forward.

We are convinced that a system capable of reading CL knowledge bases and performing inferences with it would lead to a significant breakthrough toward a wider visibility and practical deployment of CL.

9 Conclusion

In our paper, we have discussed the importance of decidability for practical knowledge representation, putting particular emphasis on well-established Semantic Web languages. On a general level, we argued that from a practical perspective, decidability only provides a qualitative advantage if it comes with a decision algorithm the runtime of which can be guaranteed to be below an acceptable time span. We then identified scenarios where such an algorithm is strictly required as well as scenarios where this is not the case.

We therefore conclude that the necessity to constrain to decidable formalisms strongly depends on the typical automated reasoning tasks to be performed in a knowledge-based system dedicated to a concrete purpose.

In order to still provide necessary reasoning services, we proposed to use available highly optimized implementations of first-order theorem provers and model finders. Realistically, specialized reasoners such as existing OWL 2 DL reasoners are likely to be more efficient on their specific language fragment than generic FOL reasoners. But as these reasoners happen to provide (syntactically restricted) FOL reasoning themselves, all considered reasoners are largely interoperable and, therefore, could be applied in parallel to solve complex reasoning tasks conjointly. This approach offers a smooth transition path towards advanced OWL 2 reasoning without loss of efficiency on existing application areas.

Our own experiments, currently using off-the-shelf standard first-order reasoners, have yielded first encouraging results. We are confident that the good results obtained for undecidable OWL variants will carry over to Semantic Web knowledge representation formalisms that even go beyond the OWL 2 specification, such as SWRL or RIF-BLD combined with OWL 2. We will, in principle, only be limited by what first-order logic provides us.

In the light of these promising results, we strongly believe that the described strategy of providing inferencing services via a translation into FOL and the deployment of first-order reasoning machinery can also pave the way to establishing reasoning support for undecidable formalisms

cherished by the conceptual structures community. Endowing conceptual graphs and common logic with ready-to-use inferencing services along the same lines seems a feasible and worthwhile endeavor and will most likely lead to a more wide-spread adoption of these formalisms among knowledge representation practitioners.

References

1. Andréka, H., van Benthem, J.F.A.K., Németi, I.: Modal languages and bounded fragments of predicate logic. Journal of Philosophical Logic 27(3), 217–274 (1998)
2. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (eds.): RIF Core Dialect. W3C Recommendation (22 June 2010), available at http://www.w3.org/TR/rif-core/
3. Boley, H., Kifer, M. (eds.): RIF Basic Logic Dialect. W3C Recommendation (22 June 2010), available at http://www.w3.org/TR/rif-bld/
4. Brickley, D., Guha, R. (eds.): RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation (10 February 2004), available at http://www.w3.org/TR/rdf-schema/
5. de Bruijn, J. (ed.): RIF RDF and OWL Compatibility. W3C Recommendation (22 June 2010), available at http://www.w3.org/TR/rif-rdf-owl/
6. Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. Monatshefte für Mathematik und Physik 38, 173–198 (1931)
7. Gödel, K.: Über die Vollständigkeit des Logikkalküls. Ph.D. thesis, Universität Wien (1929)
8. Hayes, P.: Translating Semantic Web Languages into Common Logic. Tech. rep., IHMC Florida Institute for Human & Machine Cognition, 40 South Alcaniz Street, Pensacola, FL 32502 (18 July 2005), available at http://www.ihmc.us/users/phayes/CL/SW2SCL.html
9. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
10. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B.N., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004), available at http://www.w3.org/Submission/SWRL/
11. ISO/IEC JTC 1: Common Logic (CL): A Framework for a Family of Logic-based Languages. No. ISO/IEC 24707:2007(E), ISO International Standard (1 October 2007), available at http://cl.tamu.edu/
12. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Recommendation (22 June 2010), available at http://www.w3.org/TR/rif-overview/
13. Manola, F., Miller, E. (eds.): Resource Description Framework (RDF). Primer. W3C Recommendation (10 February 2004), available at http://www.w3.org/TR/rdf-primer/
14. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-profiles/
15. Motik, B., Patel-Schneider, P.F., Cuenca Grau, B. (eds.): OWL 2 Web Ontology Language: Direct Semantics. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-direct-semantics/

16. Motik, B., Patel-Schneider, P.F., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-syntax/
17. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-overview/
18. Patel-Schneider, P.F., Motik, B. (eds.): OWL 2 Web Ontology Language: Mapping to RDF Graphs. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-mapping-to-rdf/
19. Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend! J. Artif. Intell. Res. (JAIR) 39, 429–481 (2010)
20. Rudolph, S., Schneider, M.: On the utility and feasibility of reasoning with undecidable semantic web formalisms. Technical Report 3016, Institute AIFB, Karlsruhe Institute of Technology (2011), available via http://www.aifb.kit.edu/web/Techreport3016/en
21. Schneider, M. (ed.): OWL 2 Web Ontology Language: RDF-Based Semantics. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-rdf-based-semantics/
22. Schneider, M., Sutcliffe, G.: Reasoning in the OWL 2 Full Ontology Language using First-Order Automated Theorem Proving. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) Proceedings of the 23rd International Conference on Automated Deduction (CADE 23) (2011), (to appear)
23. Tsarkov, D., Riazanov, A., Bechhofer, S., Horrocks, I.: Using Vampire to Reason with OWL. In: Proceedings of the Third International Semantic Web Conference (ISWC 2004). LNCS, vol. 3298, pp. 471–485. Springer (2004)
24. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 42(2), 230–265 (1937)
25. Villanueva-Rosales, N., Dumontier, M.: Describing Chemical Functional Groups in OWL-DL for the Classification of Chemical Compounds. In: Golbreich, C., Kalyanpur, A., Parsia, B. (eds.) Proceedings of the 3rd International Workshop on OWL: Experiences and Directions (OWLED 2007). CEUR Workshop Proceedings, vol. 258 (2007), available at http://ceur-ws.org/Vol-258/paper28.pdf
26. Voronkov, A.: Automated Reasoning: Past Story and New Trends. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003). pp. 1607–1612. Morgan Kaufmann Publishers Inc. (2003)