Ontologies for Knowledge Graphs?

Markus Krötzsch

Center for Advancing Electronics Dresden (cfaed), TU Dresden
markus.kroetzsch@tu-dresden.de

Abstract Modern knowledge representation (KR), and description logics (DL) in particular, promises many advantages for information management, based on an unambiguous, implementation-independent semantics for which a range of reasoning services is available. These strengths align well with the needs of an ever growing information industry. Today, giants like Google, Facebook, and Wikimedia consciously deploy ontological models, and store information in graph-like data structures that are more similar to DL ABoxes than to traditional relational databases. Many smaller organisations follow, and "knowledge graphs" appear in numerous places. Clearly, logic-based KR can make significant contributions to this development, yet there is often very little adoption in typical knowledge graph applications. Focusing on Wikidata as a particular use case, this invited contribution asks which technical issues might limit the impact of symbolic KR in this area, and summarises some recent developments towards addressing them in various logics.

1 Introduction

Modern data management has re-discovered the power and flexibility of graph-based representation formats, and so-called *knowledge graphs* are now used in many practical applications. The term had originally been coined by Google's eponymous approach of managing structured knowledge for question answering, advertised as "one of the key breakthroughs behind the future of search" [11]. This activity resonated with a shift towards more flexible, schema-optional data management solutions [9], and in particular with the advent of graph databases. We therefore find a variety of knowledge graphs in industry (e.g., at Google and Facebook); on the Web (e.g., Freebase [4] and Wikidata [30]); and in research (e.g., YAGO2 [13] and Bio2RDF [3]).

Logic-based knowledge representation (KR) is a matching technology for these developments, and indeed seems to be a perfect fit for some of the main needs behind the recent shift towards knowledge graphs, due to various features:

- *Modelling with (hyper)graphs:* KR research customarily adopts a graph-based view on data, and bases most of its semantics on similar concepts.
- *Discovering connections:* reasoning is essentially about making non-local connections, and it can often be used naturally for graph analysis and querying.
- *Declarativity:* Independence of particular implementations, and even of the underlying hardware infrastructure, is a prominent requirement in modern data management. Declarative modelling is essential for achieving this.

– *Unambiguous semantics:* Interchange of knowledge between various groups of humans as well as software tools is critical in many applications. A clearly defined, formal semantics can provide this.
– *Ontology engineering services:* Many KR paradigms support the implementation of services that can verify the correctness of a model for certain constraints (the most basic being consistency), and propose measures to fix possible errors.

These strengths, together with the maturity of many KR fields, and the availability of well-supported standards like OWL [25], might be expected place KR technologies at the heart of current activities in IT companies around the world. However, this is not the practical reality. Instead, companies continue to rely on *ad hoc* graph representations and custom code for working with knowledge graphs. Even established standards such as RDF and SPARQL have only limited traction, while popular graph models such as Neo4j's *Property Graph* rely on reference implementations that might change their semantics from release to release [28].

In what follows, I will discuss this – in my view – unfortunate situation from my particular, naturally subjective viewpoint. My focus will thus be that of a designer of KR formalisms and partly also that of a programmer. Indeed, I do not consider the success of symbolic KR to be tied to the philosophical question of whether we can create a "true" model of "reality." This is hardly the goal of any company or organisation, presumably not even of Google. In practice, technology is used to solve technological problems, and the competition for KR therefore are down-to-earth Java programs or Python scripts that aim to solve the same problems.

Whether justified or not, I will also dismiss any larger scale economic and social factors. The level of technological support, standardisation, commercialisation, and industrial co-operations around modern KR research is significant and should be a sound basis for further adoption. My thesis therefore is that the issue is not a lack of advertisement or motivation, but that there are indeed important technological mismatches that research must address to advance the field. Even if not fully accurate, this view is at least a good basis for an optimistic outlook on the impact of academic research in general.

2 What is a Knowledge Graph?

The knowledge graphs in modern applications are characterised by several properties that together distinguish them from more traditional knowledge management paradigms:

(1) **Normalisation:** Information is decomposed into small units of information, interpreted as edges of some form of graph.
(2) **Connectivity:** Knowledge is represented by the relationships between these units.
(3) **Context:** Data is enriched with contextual information to record aspects such as temporal validity, provenance, trustworthiness, or other side conditions and details.

While many databases are normalised in some way (1), the focus on connections (2) is what distinguishes the graph-based view [28]. This is apparent not so much from the data structures – graphs can be stored in many formats –, but in the use of query languages. Graph query languages such as SPARQL [27] or Cypher [28] natively

support reachability queries and graph search features that are not supported in other paradigms.[1]

Contextual information (3) is introduced for a variety of reasons. It may simply be convenient for storing additional details that would otherwise be hard to represent in normalisation [28], or it may by used for capturing meta-information such as provenance, trust, and temporal validity. Indeed, knowledge graphs are often used in data integration – graphs are well suited for capturing heterogeneous data sources and their relationships –, and it is natural to retain basic information regarding, e.g., the source and trustworthiness of a particular piece of data.

While we can only speculate about the shape and content of Google's original knowledge graph,[2] we can find the above characteristics in major graph database formats:

- *Property Graph.* The popular graph model that is used in many industry applications is based on a directed multi-graph with attribute-value annotations that are used to store contextual information on each edge and node [29]. Popular property graph query languages, such as Cypher, support graph search [28].
- *RDF.* The W3C graph data standard encodes directed graphs with different types of edges. Support for storing contextual information has been added to RDF 1.1 by enabling references to named graphs [7]. The SPARQL query language for RDF supports regular path queries and named graphs [12].

Likewise, individual knowledge graphs exhibit these characteristics, for example:

- *Yago* and *Yago2* are prominent knowledge graphs extracted from Wikipedia [13]. The authors extended RDF (at a time before RDF 1.1) with quadruples to capture important contextual information related to time and space.
- *Bio2RDF* is a graph-based data integration effort in the life sciences [3]. It uses an n-ary object model to capture complex and contextual information in plain RDF graphs, i.e., it introduces new identifiers for tuples extracted from relational databases.
- *Wikidata*, the knowledge graph of Wikipedia, is natively using a graph-like data model that supports expressive annotations for capturing context [30]. Details are discussed in the next section.

3 Wikidata, the Free Knowledge Graph of Wikipedia

Wikidata is a sister project of Wikipedia that aims to gather and manage factual data used in Wikipedia or any other Wikimedia project [30]. Launched in October 2012, the project has quickly grown to become one of the largest and most active in terms of editing. As of July 2017, Wikidata stores information about almost 29 million entities,[3] and has received contributions from over 175,000 registered contributors. Content from

[1] SQL, e.g., does support recursive views that resemble the expressivity of linear Datalog, but the standard forbids the use of duplicate elimination (DISTINCT) in their construction, making them quite useless for breadth-first search on graphs that may contain cycles.

[2] Or even if this widely marketed concept exists as a clearly defined technical entity at all.

[3] This should be compared to the 5.5 million articles found in English Wikipedia.

Statement from the page of *Tim Berners-Lee* (https://www.wikidata.org/wiki/Q80):

Statement from the page of *The Imitation Game* (https://www.wikidata.org/wiki/Q14918344):

Figure 1. Two statements from Wikidata

Wikidata is widely used in other applications and on the Web, ranging from interactive query views on specific subjects (e.g., the Academy Awards portal of the major German newspaper FAZ online[4]) to general purpose question answering tools (e.g., Apple's Siri search engine in iOS 11 beta returns data from Wikidata[5]).

The main content of Wikidata are more than 166 million *statements* (as of July 2017), which describe and interrelate the entities. A Wikidata statement can be viewed as an edge in a directed graph that is further annotated by attribute-value pairs and provenance information. For example, Fig. 1 shows two statements as seen by users of wikidata.org. In both cases, the main part of the statement can be read as a directed edge: Berners-Lee's employer has been CERN, and the film *The Imitation Game* has cast member Benedict Cumberbatch. In addition, both statements include contextual information in the form of additional property-value pairs that refer to the statement (rather than to the subject of the statement). As one can see, this additional information includes classical "meta-data" such as validity time and references (collapsed in the figure), but also other details that are more similar to the modelling of n-ary relations.

Statements consist of *properties* (e.g., *employer*, *start date*, *character role*) that are given specific *values*, which may in turn be Wikidata *items* (e.g., *Alan Turing*, *CERN*) or values of specific datatypes (e.g., *1984*, which denotes a date whose precision is limited to the year). Notably, the properties and values used in the main part of the graph are the same as those used to add contextual information, and this can be exploited in queries (e.g., one can ask for actors who have played computer scientists). Moreover, Wikidata allows users to create new properties, and to make statements about them. From a KR viewpoint, Wikidata properties are therefore a special type of individual rather than a binary predicate.

Some further details of Wikidata's graph model are briefly noted: (1) the same directed relationship may occur with several annotations, e.g., in the case of Elizabeth

[4] http://www.faz.net/aktuell/feuilleton/kino/academy-awards-die-oscar-gewinner-auf-einen-blick-12820119.html

[5] https://lists.wikimedia.org/pipermail/wikidata/2017-July/010919.html

Taylor who was married to Richard Burton several times; (2) the same property can be assigned more than one value in the context of some statement, e.g., this is used when several people win an award together (*together with* being the Wikidata property used to annotate the *award received* statement); (3) the order of statements and statement annotations is not relevant.

The content of Wikidata is freely available in several formats. The native storage format is based on JSON, but for enabling query answering, the data has been mapped to RDF [10]. This mapping creates identifiers for statements (*reification*) to represent contextual data in triples. The Wikimedia Foundation operates an official SPARQL service to enable complex queries over the data, and this facility is heavily used. As of mid 2017, between 60 and more than 100 million SPARQL queries are answered per month.[6] More detailed analysis shows that more than 90% of the queries are issued by automated tools that perform community services such as data integration and quality control. Nevertheless, around 500,000 queries per month appear to be issued directly or indirectly by the browsers of human users. Another striking observation is that a significant part of these queries are using regular path expressions: their share of user queries amounts to over 30% in several months, making this the most frequently used SPARQL feature after basic JOIN and SELECT, well before UNION and OPTIONAL (ranked at about 5-10%). This confirms the graph-based nature of Wikidata.

4 Ontological Modelling with Knowledge Graphs

Projects like Wikidata could certainly benefit from KR technologies, which can help in knowledge organisation, quality control, and redundancy elimination. Indeed, while Wikidata is merely a data management platform without an explicit ontology, the community is consciously using data for capturing schematic information. Properties such as *instance of* or *subclass of* are used for basic hierarchy construction, inspired by the usage in RDFS. Wikidata does not distinguish classes as a special type of object, and arbitrary items therefore might be used as classes. Moreover, Wikidata properties are classified to express further ontological information. The Wikidata property *spouse*, e.g., is an instance of *symmetric property*.

Even if intended to be ontological, such user-generated assertions remain data without any special treatment in the system. It would clearly be desirable to axiomatise their semantics using a suitable ontology language, e.g., to declare that instances of *symmetric property* are indeed symmetric. It is important here to avoid any confusion of this approach with that of the RDF semantics, which interprets some (possibly inferred) data directly as ontological axioms; this may lead to undesired semantic issues [26]. Axiomatisation instead treats all data as data, clearly separated from schematic knowledge. It still allows us to capture the semantics of important ontology languages, e.g., it is possible to view $\mathcal{EL}$ ontologies as data on top of which inferences are computed by rules [14].

Unfortunately, the relational (or hypergraph) data model used by most KR formalisms does not provide a direct way of representing Wikidata statements or Property Graph

[6] These statistics stem from an ongoing research collaboration with the Wikimedia Foundation https://meta.wikimedia.org/wiki/Research:Understanding_Wikidata_Queries.

edges, since neither can be captured with n-ary relations of any fixed arity n. We may, however, represent statements using auxiliary individuals, similar to the W3C's *RDB to RDF Mapping Language* representation of relational tuples in RDF graphs [8]. We may therefore imagine the statement that Taylor was married to Burton starting from 1964 and ending 1974 to be represented in several facts:

$$\mathsf{spouse}_1(s, \mathsf{taylor}) \quad \mathsf{spouse}_2(s, \mathsf{burton}) \quad \mathsf{start}(s, 1964), \quad \mathsf{end}(s, 1974) \tag{1}$$

where s is a constant representing the statement. It turns out that most prominent ontology languages are unable to express even the simplest types of relationships on knowledge graphs. For example, the claim that *spouse* is symmetric implicitly involves the assertion that, in the above example, Burton is also married to Taylor *with the same start and end date*. This asserts the existence of another, inferred statement (represented by a dedicated individual), which could be written as follows:

$$\begin{aligned} \forall x, y_1, y_2, y_3, y_4.\mathsf{spouse}_1(x, y_1) \wedge \mathsf{spouse}_2(x, y_2) \wedge \mathsf{start}(x, y_3) \wedge \mathsf{end}(x, y_4) \\ \rightarrow \exists v.\mathsf{spouse}_2(v, y_1) \wedge \mathsf{spouse}_1(v, y_2) \wedge \mathsf{start}(v, y_3) \wedge \mathsf{end}(v, y_4) \end{aligned} \tag{2}$$

Description logics cannot express this statement, since they allow only exactly one universally quantified variable to occur in both premise and conclusion in axioms that have existential quantifiers. Rule languages have no such restriction on universal quantifiers, but the most basic (and most popular) logical rule language Datalog does not support existentials at all. Rule languages with existential quantification that have been proposed for data integration scenarios include *global-as-view* and *local-as-view* mappings [20]. Neither can express (2), since they support only single atoms on the source and on the target side, respectively.

Indeed, all we can say is that (2) has the form of an *existential rule* (also known as *tuple-generating dependency*), for which reasoning is undecidable in general [1]. Note, moreover, that this is just an individual example, which may not be representative for all modelling needs that arise even in the specific case of Wikidata. For example, we might wish to classify currently married couples as those whose spouse relationship has *no* end date, which would require some form of negation that is not supported by existential rules either. Nevertheless, one can hardly claim that the scenario considered here is in any way unusual, or inherently challenging from a computational viewpoint. It would therefore be quite disappointing if KR research could not offer more concrete solutions to address this case.

5 Decidable Fragments of Existential Rules

Significant recent work has been invested into identifying fragments of existential rules for which reasoning is decidable or even tractable. Good overviews of several main approaches are provided in the literature [2,5,6]. We may therefore hope that rules as in (2) could be covered in one of these fragments.

However, it turns out that this is often not immediate, since known criteria rely on syntactic characteristics that are often not present when modelling rules for knowledge graphs [18]. Two common criteria are as follows:

– *Guardedness* requires certain variables of rules to occur together in a single body atom. It ensures the existence of tree-like models that can be represented finitely.
– *Linearity* requires rule bodies to consist of only one atom. It ensures that reasoning tasks can be rewritten into first-order queries.

Both of these properties are clearly lost when decomposing statements during reification, though their semantic consequences remain [18]. The only class of criteria that is largely robust against statement decomposition is *acyclicity*, which ensures finite models, and indeed it might be hoped that this will be a suitable condition for knowledge graph ontologies.

For the other cases, it is possible to re-discover the fact that rules belong to a known decidable fragment by applying suitable *de-normalisation* algorithms that attempt to correlate the components of decomposed statements so as to treat them like higher-arity predicates (albeit with a possibly unspecified arity). Details on this procedure have been studied in previous work [18].

6 Attributed Logics

The discovery of workable existential rule fragments can be a useful first step towards ontologies for knowledge graphs, but it also bears some limitations:

– Ontological axioms that refer to normalised statement representations may be significantly more complex and less maintainable than their simple use cases would suggest.
– It is very difficult to express rules that are applicable to the variety of possible forms of statements. For example, a *spouse* statement may or may not have a start, an end, or various other annotations, and each combination of these features would require a dedicated rule.
– Normalisation gives up on the distinction between contextual information and the main connections of the graph. This is not desirable, since one would usually consider individual statements (with their annotations) to be complete, whereas the overall set of statements may never be complete (open world semantics).

This motivates the introduction of *annotation sets*, i.e., sets of attribute-value pairs, as a primary data structure in the syntax and semantics of logics, as done in recent work by Marx et al. [21]. To this end, we can simply extend relational structures by moving from relational tuples to relational tuples annotated each with a finite set of attribute-value pairs. The resulting structures have been dubbed *multi-attributed relational structures* (MARS).[7] As in Wikidata, attributes are modelled as domain elements rather than as predicates in this approach.

It is not difficult to extend first-order logic syntactically to refer to the annotation sets of a particular atom. For example, the information encoded in (1) might be represented as follows:

$$\mathsf{spouse}(\mathsf{taylor}, \mathsf{burton})@\{\mathsf{start} : 1964, \mathsf{end} : 1974\} \tag{3}$$

[7] The name is inspired by *attributed graphs*, which MARS generalise to hypergraphs with multiple values per attribute.

and the rule that spouse is symmetric then takes the form

$$\forall Z.\forall x, y.\mathsf{spouse}(x, y)@Z \rightarrow \mathsf{spouse}(y, x)@Z \tag{4}$$

The resulting extension of first-order logic is called *multi-attributed predicate logic* (MAPL). In this unrestricted form, it is highly expressive – in fact it can capture arbitrary *weak second-order logic* (WSO), i.e., second-order logic with quantification over finite predicates only [19]. WSO is strictly harder than first-order logic, since deciding basic reasoning problems is not even semi-decidable.

Hence MAPL is only a conceptual framework, but not a practical KR language yet, and we may look for useful fragments of MAPL with a decidable reasoning problem. As a first approach, Marx et al. introduce *MAPL rules* (MARPL), which restricts to a Horn-logic fragment of MAPL with the following key restrictions:

- all formulae take the form of Horn implications,
- all variables (first-order and second-order) are universally quantified, and
- all second-order variables are bound by some body atom.

While this language can still express (4), it would otherwise be rather limited, since it has hardly any way of defining conditions on annotation sets (as represented by second-order variables). For example, one cannot define the married couples as those with a spouse relationship that has no end date (but is otherwise arbitrarily annotated). To address this, MARPL introduces expressions that constrain the bindings of second-order variables, namely:

- *Open specifiers* that define a minimal set of attribute-value pairs that must be present in an annotation set (lower bound), possibly with wildcard value +. For example, $\lfloor \mathsf{start} : +, \mathsf{position} : \mathsf{fellow} \rfloor(Z)$ requires that Z contains at least one attribute value for start, and the value fellow for attribute position.
- *Closed specifiers* that define exactly which attribute-value pairs are present in an annotation set (upper and lower bound), again with optional wildcards. For example, $[\mathsf{characterRole} : *](Z)$ requires that Z contains zero or more attribute values for characterRole, but no other attributes.
- These specifiers can further be combined with Boolean set operations to define additional expressions, e.g., $(\lfloor \mathsf{start} : + \rfloor \setminus \lfloor \mathsf{end} : + \rfloor)(Z)$ for sets that have a start but no end.

Note that these conditions enable a closed-world assumption on the contents of annotation sets, since we can check for the absence of particular attribute-value sets.

Finally, Marx et al. also propose a mechanism for creating customised annotation sets from given input in a deterministic and declarative fashion. Indeed, it may often be necessary to derive annotated facts where the annotation set is not merely a copy of some existing set, but computed from the premises. To this end, a notion of *function definition* is introduced, which can be viewed as a mini Datalog program that derives a new annotation set from given inputs.

All of these extensions in MARPL – specifiers and function definitions – can be expressed in MAPL using logical sentences, which, however, do not have the restricted rule shape of MARPL. Even with these extensions, MARPL remains decidable:

Theorem 1 ([21]). *Conjunctive query answering with respect to MARPL ontologies is ExpTime-complete, both in terms of combined and data complexity.*

Combined complexity therefore matches basic Datalog, but the high data complexity might be surprising. It is only obtained by applying defined functions to create an exponential number of different annotation sets, which is arguably not expected to be a major problem in practical scenarios. On the other hand, the result also suggests that MARPL is powerful enough to act as a meta-language that interprets parts of the input data as ontological knowledge, and therefore must have a rather high complexity with respect to this input.

7 Description Logics for Knowledge Graphs

DLs are one of the most prominent KR formalisms today, and it is natural to ask whether and how they can be adopted to match this formalism. Both existential rules and MAPL rules are crucially different from DLs, in that their use of variables allows expressing more complex graph structures and richer dependencies between premise and conclusion. It has long been a major focus in DL research to develop extensions that integrate such rule-like capabilities, with diverse approaches such as *description graphs* [22], *hybrid MKNF* [23], *dl-safe rules* [24], or *nominal schemas* [15]. Each of these approaches deviates from traditional, variable-free DL syntax in some way, and indeed this seems unavoidable to achieve the desired goal.

For the definition of *attributed description logics*, a detailed proposal is included in the proceedings where this text is to appear [17], and an extended account of this work can be found in a technical report [16]. I will therefore only give a brief outline here.

Semantically, attributed DLs use the same MARS-based model theory as MAPL in general, so every fact can occur with one or more finite annotation sets. Syntactically, the approach attaches annotation variables to role and concept names within axioms, and adds pre-conditions that further restrict the sets that these variables may represent. For example, the following DL (RBox) axiom states that spouse is a symmetric relation:

$$\mathsf{spouse}@Z \sqsubseteq \mathsf{spouse}^{-}@Z$$

where Z is implicitly universally quantified. A more complex example is

$$Z : \lfloor \mathsf{pos} : \mathsf{fellow} \rfloor \quad \exists \mathsf{employer}@Z.\{\mathsf{cern}\} \sqsubseteq \mathsf{CernFellow}@[\mathsf{start} : Z.\mathsf{start}, \mathsf{end} : Z.\mathsf{end}]$$

stating that everybody employed by CERN as a fellow is in the class CernFellow annotated with the same start and end date as the employment. The dot-notation in the conclusion (e.g., start : Z.start) is used to define new annotations directly. In contrast to MARPL's defined functions, attributed DLs allow conclusions to have partially specified annotation sets that are not fully determined by the axioms, which is more natural in the open world approach of DLs.

Decidable attributed DLs tend to be exponentially more complex than their classic base DL, for example:

Theorem 2 ([16]). *Standard reasoning tasks in attributed $\mathcal{ALCH}$, denoted $\mathcal{ALCH}_{@}$, are 2ExpTime-complete.*

For the highly expressive DL $\mathcal{SROIQ}$, this added complexity is hidden in the (already high, but partly orthogonal) base complexity, so reasoning in $\mathcal{SROIQ}_@$ remains N2ExpTime-complete [16].

8 Outlook

Are the structural limitations of popular KR languages the only major obstacle to further adoption of symbolic KR in applications of knowledge graphs? Probably not, but they are at the very beginning of any path to further adoption. The Wikidata community has developed complex technical solutions for quality control, query answering, and data integration, often based on custom code. A unifying, declarative solution with good tool support would surely be welcome. But any discussion in this direction must start from basic examples such as the symmetry of spouse. Users will naturally ask how KR can help to solve the current problems, and for most KR approaches, including DLs, the plain answer is that they can't. It must be a priority of research to address this, and this does indeed seem quite feasible.

Of course, this will not suffice. To realise their full potential, KR technologies will have to go further. Significant advancements are needed at least in the following areas:

- Cross-system integration: KR-based solutions must become more compatible with existing infrastructures and tools, e.g., to reason over the outputs of machine learning algorithms or remote Web services.
- Smart editing: Current editing tools for Java (for which reasoning is undecidable) offer much more useful assistance than modern ontology editors, although the latter should have a much better understanding of the semantics of the edited artefacts.
- Robustness: KR, and especially practical reasoners, must continue to work towards error tolerance, recovery, and adaptivity.
- Quantitative computation: New approaches are needed for working with quantitative data, supporting relevant computations but avoiding unsolved problems in numerical reasoning.
- Modularisation: There is little design-time support for separating concerns in ontological modelling, and many KR approaches continue to view ontologies as large, amorphic sets of axioms. This is an obstacle for the work of large teams or distributed projects.
- Hardware adaptivity: Reasoning is not taking advantage of the increasing variety of modern compute platforms, and usually support at most some mild form of multi-CPU parallelism. There are hardly any concepts for taking advantages of computer clusters, GPUs, and in-memory computing.

A unifying theme across these areas is that we need to further develop symbolic KR in the context of other paradigms of *computation* rather than viewing it as a static paradigm of mere *representation*. This may at first conjure up Prolog's 1970s idea of "programming in logic," and with it the concern about a creeping loss of declarativity. This relationship, however, is superficial, since our present situation is very different. For many years, our community has recognised data management and integration as an important use case. At the same time, the data management community has embraced

a variety of more declarative processing paradigms, that provide complex analytics in ways that abstract from specific implementations. The convergence of these areas is desirable and realistic.

As we are looking at the wider development, it seems likely that the continued integration and curation of valuable knowledge will unlock application scenarios that are currently out of reach, and become a central component of future AI. No single technology – whether KR, machine learning, databases, or natural language processing – will provide all the necessary components, but KR has a potential to act as an integrating and coordinating layer that declaratively combines many sources of information in a way that is fully understood and controlled by human users. Our ongoing and future research activities will decide whether the field can live up to this expectation.

Acknowledgements. I would like to thank the organisers of DL Workshop for providing me with the opportunity for publishing this invited contribution. Several collaborators have made significant contributions to the research results summarised herein. In particular, I would like to thank Maximilian Marx, Ana Ozaki, Veronika Thost, and Denny Vrandečić. Research on Wikidata queries is enabled by the collaboration of the Wikimedia Foundation. This work is partly supported by the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (cfaed), the Collaborative Research Center SFB 912 (HAEC), and in Emmy Noether grant KR 4381/1-1 (DIAMOND).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
2. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9–10), 1620–1654 (2011)
3. Belleau, F., Nolin, M., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. J. of Biomedical Informatics 41(5), 706–716 (2008)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A collaboratively created graph database for structuring human knowledge. In: Proc. 2008 ACM SIGMOD Int. Conf. on Management of Data. pp. 1247–1250. ACM (2008)
5. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. J. of Artif. Intell. 193, 87–128 (2012)
6. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. J. of Artificial Intelligence Research 47, 741–808 (2013)
7. Cyganiak, R., Wood, D., Lanthaler, M. (eds.): RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation (25 February 2014), available at http://www.w3.org/TR/rdf11-concepts/
8. Das, S., Sundara, S., Cyganiak, R. (eds.): R2RML: RDB to RDF Mapping Language. W3C Recommendation (27 September 2012), available at https://www.w3.org/TR/r2rml/
9. Edlich, S.: NOSQL Databases (2017), http://nosql-database.org, accessed July 2017
10. Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., Vrandečić, D.: Introducing Wikidata to the linked data web. In: Proc. 13th Int. Semantic Web Conf. (ISWC'14). LNCS, vol. 8796, pp. 50–65. Springer (2014)
11. Google Inc.: Knowledge – Inside Search. https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html, retrieved July 2017 (2017)

12. Harris, S., Seaborne, A. (eds.): SPARQL 1.1 Query Language. W3C Recommendation (21 March 2013), available at http://www.w3.org/TR/sparql11-query/
13. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. J. of Artif. Intell. 194, 28–61 (2013)
14. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 2668–2673. AAAI Press/IJCAI (2011)
15. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Proc. 20th Int. Conf. on World Wide Web (WWW'11). pp. 645–654. ACM (2011)
16. Krötzsch, M., Marx, M., Ozaki, A., Thost, V.: Attributed description logics: Ontologies for knowledge graphs. Tech. rep., TU Dresden (2017), https://iccl.inf.tu-dresden.de/web/Techreport3032
17. Krötzsch, M., Marx, M., Ozaki, A., Thost, V.: Reasoning with attributed description logics. In: Proc. 30th Int. Workshop on Description Logics (DL'17). CEUR Workshop Proceedings, CEUR-WS.org (2017)
18. Krötzsch, M., Thost, V.: Ontologies for knowledge graphs: Breaking the rules. In: Groth, P.T., Simperl, E., Gray, A.J.G., Sabou, M., Krötzsch, M., Lécué, F., Flöck, F., Gil, Y. (eds.) Proc. 15th Int. Semantic Web Conf. (ISWC'16). LNCS, vol. 9981, pp. 376–392 (2016)
19. Leivant, D.: Higher order logic. In: Gabbay, D.M., Hogger, C.J., Robinson, J.A., Siekmann, J.H. (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 2, Deduction Methodologies, pp. 229–322. Oxford University Press (1994)
20. Lenzerini, M.: Data integration: A theoretical perspective. In: Popa, L. (ed.) Proc. 21st Symposium on Principles of Database Systems (PODS'02). pp. 233–246. ACM (2002)
21. Marx, M., Krötzsch, M., Thost, V.: Logic on MARS: Ontologies for generalised property graphs. In: Proc. 26th Int. Joint Conf. on Artificial Intelligence (IJCAI'17). AAAI Press (2017), to appear; available at https://iccl.inf.tu-dresden.de/web/Inproceedings3141
22. Motik, B., Cuenca Grau, B., Horrocks, I., Sattler, U.: Representing ontologies using description logics, description graphs, and rules. Artificial Intelligence 173(14), 1275–1309 (2009)
23. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5), 30:1–30:62 (2010)
24. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. J. of Web Semantics 3(1), 41–60 (2005)
25. 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/
26. Patel-Schneider, P.F.: Building the semantic web tower from RDF straw. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 546–551. Professional Book Center (2005)
27. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Recommendation (15 January 2008), available at http://www.w3.org/TR/rdf-sparql-query/
28. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly Media (2013)
29. Rodriguez, M.A., Neubauer, P.: Constructions from dots and lines. Bulletin of the American Society for Information Science and Technology 36(6), 35–41 (2010)
30. Vrandečić, D., Krötzsch, M.: Wikidata: A free collaborative knowledgebase. Commun. ACM 57(10) (2014)