Too Much Information: Can AI Cope With Modern Knowledge Graphs?

Markus Krötzsch[0000−0002−9172−2601]

TU Dresden, Dresden, Germany, markus.kroetzsch@tu-dresden.de

Abstract Knowledge graphs play an important role in artificial intelligence (AI) applications – especially in personal assistants, question answering, and semantic search – and public knowledge bases like Wikidata are widely used in industry and research. However, modern AI includes many different techniques, including machine learning, data mining, natural language processing, which are often not able to use knowledge graphs in their full size and complexity. Feature engineering, sampling, and simplification are needed, and commonly achieved with custom preprocessing code. In this position paper, we argue that a more principled integrated approach to this task is possible using declarative methods from knowledge representation and reasoning. In particular, we suggest that modern rule-based systems are a promising platform for computing customised views on knowledge graphs, and for integrating the results of other AI methods back into the overall knowledge model.

1 Introduction

The modern-day rise of artificial intelligence (AI) in research and applications has also sparked a renewed interest in graph-based knowledge representation. So-called *knowledge graphs* (KGs) have become essential resources for intelligent assistants such as Apple's Siri and Amazon's Alexa [48], for question answering features of modern search engines such as Google[1] and Microsoft Bing, and for the new expert systems in the style of IBM's Watson [21]. As with many recent AI breakthroughs, this has partly been a matter of scale: modern *knowledge graphs* are orders of magnitude larger than knowledge bases conceived in early AI. Public KGs such as Wikidata [52], YAGO2 [29] and Bio2RDF [5] comprise hundreds of millions of statements; KGs in parts of the web search industry are significantly larger still (due to, e.g., crawled schema.org annotation data and other resources, such as maps).

The free availability of large knowledge graphs is also creating new opportunities in research. KGs are used to improve the quality of approaches to natural language processing, machine learning, and data mining [43]. Combining techniques from several fields also leads to advances in knowledge representation, e.g., for improved association rule mining [28]. Machine learning meanwhile can help KG construction and maintenance, e.g., by employing methods of *knowledge graph embedding* to predict missing information or to detect errors [11,54,43].

[1] The company's *Google Knowledge Graph* is the origin of the term.

Nevertheless, the promise of perfect synergy between knowledge graphs and AI often remains unfulfilled. Most KGs available today are the work of human authors, who have contributed either directly or by creating heavily structured online content (HTML tables, Wikipedia templates, schema.org annotations, etc.) from which KGs could be extracted.[2] When Google decided to migrate its former KG project *Freebase* to Wikidata, it largely relied on human labour [49]. Even closely related AI techniques, such as ontology-based knowledge representation, are used in only a few KG projects.

Why don't we see much more use of neighbouring AI methods for building, maintaining, and using KGs? The answer might be quite simple: the "knowledge graphs" addressed in much of the current research are different from the *real* KGs used in applications. Real KGs are predominantly based on annotated directed graph models, where directed, labelled edges are augmented with additional context information (validity time, data source, trustworthiness, etc.). Related data models are used in Wikidata (Section 3), Yago2 [29], and in the widely used *property graph* data model, where graph edges are conceived as complex data objects with additional key-value pairs [46]. Real KGs are also very large.

In contrast, research in other fields of AI often uses small, simplified excerpts of KGs for benchmarking. For example, link prediction through KG embeddings is still frequently evaluated with the *FB15K* dataset, which contains about 600,000 edges for 15,000 Freebase entities [11]. When Google announced the discontinuation of Freebase in 2014, the KG had almost 45M entities and 2.5 billion edges, and it was using annotations, e.g., for time (based on so-called *Compound Value Types*). In fact, KG embeddings often conceive graphs as sets of "triplets" – a simplified form of RDF triples – and n-ary relations, compound structures, quantities, and annotations are poorly supported [43].

These limitations are not specific to KG embeddings, which can actually cope with "triplet" graphs of significant size. Other knowledge-based data mining techniques are also challenged by the scale of modern KGs. For example, Formal Concept Analysis (FCA) requires a projection of graphs to Boolean propositions ("attributes") [22], and studies are typically working with hundreds of attributes at most (see, e.g., [25]). Similar limitations in size and complexity apply to other rule mining approaches [28]. The field of network analysis offers many highly scalable algorithms, e.g., for computing centrality measures such as PageRank, but they need KGs to be preprocessed to obtain directed (and often unlabelled) graphs [38]. Which part of the KG is used to obtain the network has a major impact on the meaning and quality of the results. Ontological knowledge representation and reasoning also has been scaled to hundreds of millions of statements, especially in recent rule engines [7,51], but does not cope well with exceptions and errors, and lacks a general approach for handling annotations [35,33].

Simply put: no single AI approach is able to handle current KGs in their full size and complexity. Indeed, although each approach might be further improved, one should not expect future AI to be based on a single principle. New ideas are needed for reconciling different AI formalisms. IBM Watson has pioneered one way of combining the strengths of formerly incompatible methods in question answering [21], but this integration is

[2] Crawling and extracting such content is a difficult task, and a worthy research area in itself, yet the main work of knowledge gathering remains that of the human author.

limited to selecting the best answer at the output level. To fully exploit KGs in AI, we also need principled ways of defining *input* data that can be used by other techniques (including the task of feature engineering in machine learning). In this paper, we conceive this as the task of defining suitable *views* on KG data, and we propose the use of advanced recursive query mechanisms to solve it in a declarative way. Modern rule engines can compute some useful views already, but mostly lack the features needed for integrating other methods. Moreover, for full integration, the outputs of such methods must be reconciled with the KG contents.

In this position paper, we motivate this new approach, discuss the relevant state of the art, and derive open issues for future research needed to realise this idea. The text includes parts of an earlier publication of KGs in ontological modelling [33].

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. This is apparent not so much from the data structures – graphs can be stored in many formats –, but from the use of query languages. Graph query languages such as SPARQL [45] or Cypher [44] natively support reachability queries and graph search features that are not supported in other paradigms.[3]

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, or it may be 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, we can find the above characteristics in major graph database formats:

– *Property Graph.* The popular graph model that is used in many applications is based on a directed multi-graph with attribute-value annotations that are used to store contextual information on each edge and node [46]. Popular property graph query languages, such as Cypher, support graph search [44].

[3] SQL supports 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.

– *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 [17]. The SPARQL query language for RDF supports regular path queries and named graphs [26].

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

– *Yago* and *Yago2* are prominent knowledge graphs extracted from Wikipedia [29]. 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 [5]. 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 DBs.
– *Wikidata*, the knowledge graph of Wikipedia, is natively using a graph-like data model that supports expressive annotations for capturing context [52]. Details are discussed in the next section.

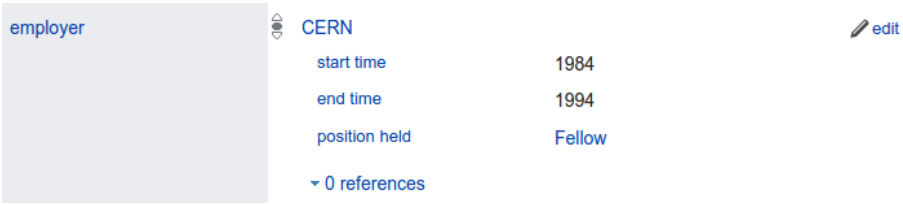
3 Wikidata, the Knowledge Graph of Wikipedia

To gain a better understanding of the problem, it is instructive to take a closer look at a modern KG. We choose Wikidata, the knowledge graph of the Wikimedia Foundation, since it is freely accessible and has become a major resource in many applications. Wikidata is a sister project of Wikipedia that aims to gather and manage factual data used in Wikipedia or any other Wikimedia project [52]. Launched in October 2012, the project has quickly grown to become one of the largest and most active in terms of editing. As of March 2019, Wikidata is one of the largest public collections of general knowledge, consisting of more than 680 million statements about more than 55 million entities. Wikidata is prominently used by intelligent agents such as Alexa and Siri [48], but also in many research activities, e.g., in the life sciences [12], in social science [53], and in Formal Concept Analysis [24,25].

The main content of Wikidata are its *statements*, 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

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

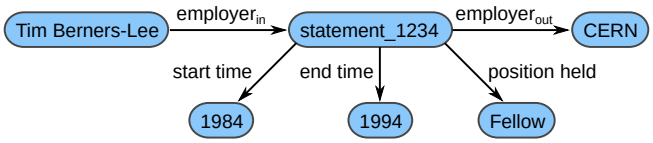


Figure 2. Plain graph encoding of the upper statement in Fig. 1

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 knowledge representation 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 different annotations, e.g., in the case of Elizabeth 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* is the Wikidata property used to annotate the *award received* statement); and (3) the order of statements and statement annotations is not relevant.

Wikidata's rich graph model can of course be encoded in simpler graph structures, which mainly requires some way of encoding statement annotations. To this end, statements are represented by own vertices in the graph, whose incidental edges can indicate their related entities and annotations. Figure 2 illustrates this approach for one encoding (other encodings would be possible [27]). Note that each Wikidata property gives rise to several distinct edge labels, and in particular that the main property *employer* is split into two parts. Wikidata uses a similar encoding to export its contents to RDF. The actual encoding is slightly more complex, e.g., since some data values such as geographic coordinates also need to be encoded using several edges, and since the export already includes several alternative encodings as well as further data [39]. As of March 2019, the RDF export of Wikidata comprises 7.28 billion triples with more than 64,000

RDF properties.[4] It is natural that a more normalised representation like RDF requires significantly more edges to capture the contents of a KG. The magnitude of this increase underlines the challenge that AI methods based on simpler graph models are facing.

The RDF encoding of Wikidata can be downloaded in the form of file exports and queried through a public SPARQL query service, which is based on the current version of this continuously changing KG. Details are described by Malyshev et al. [39]. The public SPARQL service is used as an API in many applications, and serves in the order of 100 million queries per month. Malyshev et al. also published 575 million anonymised queries and provide an initial analysis of their form and content.

4 Ontology-Based Views on Knowledge Graphs

The sheer size of Wikidata creates difficulties for some approaches in data mining and machine learning, but the bigger challenge lies in the complexity of the data, in particular due to the use of complex data types and annotations. Interpreting statements might require background knowledge on Wikidata's modelling conventions or at least a certain amount of common sense.

Example 1. Wikidata specifies many historic facts, and among the most common annotations are those for temporal validity. For example, *London* is said to be located in five administrative regions, including the *Kingdom of Wessex* and two distinct entities called *England*. The only current value is *Greater London*, from which a chain of (temporally current) *located in* statements leads to present-day England. A challenge in this inference is that the times specified for each statement along the chain are different, although the intervals overlap. Population numbers and mayors are also commonly temporalised, whereas other transient statements do not specify a temporal context, e.g., London is stated (without any annotations) to be the *capital of* ten entities, including some that no longer exist. Indeed, many entities also specify times for inception (creation) and dissolution (abolishment), which should be considered when interpreting statements.

Notably, the background knowledge needed to correctly interpret such statements might not be encoded in Wikidata at all. Applying data mining or machine learning techniques to the unfiltered data therefore is likely to produce unintended results, even if the scalability issues can be overcome. This suggests that preprocessing is needed to obtain not only smaller but also more regular, hence *predictable*, slices of the data.

Instead of relying on ad hoc preprocessing, we propose to model the required background knowledge explicitly using declarative knowledge representation languages. Indeed, the "slices of the data" that we seek correspond to *views* (under the meaning common in databases). Logical formalisms have been proposed for specifying such views declaratively, comprising approaches such as *Datalog* queries [2], *inclusion dependencies* [37], *tuple-generating dependencies* [13], and *ontology-based data access* (OBDA) [14,32]. It is worth to consider these approaches for use on KGs.

However, the application of formal logic to KGs is not straightforward either, since the latter may not have a native expression in predicate logic. Statements as in Fig. 1

[4] https://grafana.wikimedia.org/dashboard/db/wikidata-query-service

could be represented as n-ary relations, but the arity would be different for each statement depending on how many annotations are given. Indeed, Wikidata does not impose a limit on the number of annotations, and some statements use a large number of them. For a faithful relational representation with predicates of bounded arity, we therefore need to use an encoding like the one in Fig. 2.

With this encoding, any modelling language that is compatible with predicate logic can be used on knowledge graphs, including description logics (DLs) – today's most common ontology languages and the basis of OBDA.

Example 2. Hanika et al. apply Formal Concept Analysis to Wikidata by defining attributes based on the incidence of Wikidata properties to certain objects [25]. Using relation names as in Fig. 2, we can express, e.g., that every entity that is the source of a $\mathsf{mother}_{\mathsf{in}}$ relation should be in a unary predicate Mother (used as an attribute in FCA). In first-order logic, this is expressed as $\forall x.(\exists y.\mathsf{mother}_{\mathsf{in}}(x, y)) \leftrightarrow \mathsf{Mother}(x)$. An equivalent DL sentence is $\exists \mathsf{mother}_{\mathsf{in}}.\top \equiv \mathsf{Mother}$. Similar uses of DL for defining FCA attributes have been considered previously [47].

Note that the formalisation in Example 2 actually defines a two-way relationship between the view and the KG, such that one could, e.g., interpret association rules computed on this view as logical sentences from which new information can be derived about the KG [10]. The logic-based approach thus is more than just a declarative way of preprocessing KGs: it also is in principle invertible, supporting the interpretation of results obtained on a view with respect to the original data. But Example 2 is still too simple. Unary view predicates will not always be enough (e.g., we need "triplets" in KG embeddings), and significantly more complex view definitions would be needed to address use cases as in Example 1.

Unfortunately, if we try to specify more general views, we encounter severe expressivity limits. DLs, for example, offer very little expressivity for deriving binary relations (which could be interpreted as triplets), and are generally too weak to express even simple relationships on rich KG models [33]. The root cause of this restriction is the close connection of DLs to guarded logics, which syntactically ensure tree-like model structures by requiring *guard atoms* that bind certain variables. Our decomposition of statements into several smaller atoms makes it very hard to find guards, and other kinds of guarded logics are therefore similarly restricted on such KG encodings [35].

A more suitable logical formalism might therefore be Datalog, the simple recursive query language that syntactically corresponds to function-free first-order Horn-logic without existential quantifiers. Indeed, Datalog rules can express arbitrary positive relational patterns (conjunctive queries) in their premise, requiring no guard. Their weakness is that they are restricted to the given set of domain elements, i.e., they cannot infer the existence of new entities. Example 2 shows a case where this power is useful (for the "$\leftarrow$" part of the equivalence). Datalog can be generalised to support this, leading to so-called *existential rules* or *tuple-generating dependencies*, but then other restrictions must be imposed to retain decidability.

Part of this difficulty arises from the loss of structure that our relational encoding has incurred, since we can no longer distinguish the (non-local, possibly incomplete) connection structure of the KG from the (local, complete) annotations of individual

edges. Marx et al. therefore proposed to explicitly include annotations into relational calculus, obtaining what they call *attributed logics* [41]. For example, the upper statement in Fig. 1 could be written in attributed logic as an atom

$$\mathsf{employer}(\mathsf{TimBL}, \mathsf{CERN})@\{\mathsf{start\_time} : 1984, \mathsf{end\_time} : 1994, \mathsf{position} : \mathsf{Fellow}\}.$$

In this syntax, *attributes* (e.g., position) and values (e.g., 1984) are treated like logical terms, and in particular can be variables. In addition, attributed logics allow for quantification over (finite) sets of attributes.

In its full generality, this yields a logic with an undecidable entailment problem, and Marx et al. propose a restricted rule-based language *MARPL* that uses specialised primitives for expressing properties of annotation sets.

Example 3. The following MARPL rule defines a CFellow to be somebody employed by CERN as a fellow, and it copies the respective start and end time (if given):

$$\mathsf{employer}(x, \mathsf{CERN})@Z \wedge \lfloor \mathsf{pos} : \mathsf{Fellow} \rfloor(Z) \rightarrow \mathsf{CFellow}@\{\mathsf{start} : Z.\mathsf{start}, \mathsf{end} : Z.\mathsf{end}\}$$

All variables are implicitly quantified universally. Z is bound to the annotation set of the employer fact, and required to contain pos : Fellow and possibly other attribute-value pairs (denoted by the bracket that is opened to the top). The conclusion of the rule is the required fact, using a shortcut notation from [34] to copy (zero or more) values of the start and end attributes.

Theorem 1 ([41]). *Conjunctive query answering with respect to MARPL ontologies is ExpTime-complete, both in terms of combined and data complexity.*[5]

Combined complexity therefore matches basic Datalog, but the high data complexity might be surprising. It is caused by MARPL's ability of creating an exponential number of different annotation sets, which is already possible if only a fixed set of attributes is used in annotations. It is not expected that reasoning on real KGs leads to such combinatoric explosions in the relevant annotation sets, but it might still be a strength of the proposal that it can also describe view definitions that are not in polynomial time.

5 Rule Engines as Platforms for Artificial Intelligence

New ontology languages such as MARPL might develop into suitable formalism for expressing views and logical relationships over KGs, but they do not provide the required integration with other AI approaches yet. At the time of this writing, there is not even an implemented reasoning engine for MARPL. If we look for existing reasoners that scale to large amounts of data, (existential) rule reasoners stand out [7]. Indeed, these engines have been developed as extensions of the deductive query language *Datalog*, which has a long tradition in databases, where scalability to large datasets is a prime concern. In recent years, there has been significant progress in this area, and many new

[5] Data complexity characterises the worst-case asymptotic complexity of the reasoning problem for a fixed logical theory (i.e., MARPL rule set) with respect to the size of the input data (KG).

rule-based systems have been presented [3,4,8,9,23,42,50]. In this section, we therefore discuss what it would take to develop such systems into a suitable basis for reconciling knowledge graphs with other AI methods.

Existential rule systems, such as RDFox [42] or VLog [50], are often based on bottom-up materialisation of inferences, known as the *chase* in databases. This is the most common reasoning approach for Datalog, and it is naturally extended to existential rules. Common variants of the chase include the *standard* (a.k.a. *restricted*) *chase* [20], the *skolem* (a.k.a. *semi-oblivious*) *chase* [40], and the *core chase* [18]. They differ in the approach taken to determine if the application of a rule can be considered redundant in the sense that it will not contribute to producing distinct new entailments.

However, reasoning with existential rules is undecidable in general, and indeed any chase may fail to terminate in some cases – the algorithm is sound but not complete. Termination of the chase is also undecidable, but many sufficient criteria have been developed for detecting it [16]. Almost all of these criteria ensure that the skolem chase over a set of rules will terminate *universally*, i.e., on all possible sets of input facts. The following result applies in any such case:

Theorem 2 (Marnette, [40]). *Let Σ be a set of existential rules, such that, for every set of input facts $\mathcal{F}$, the skolem chase over $\Sigma \cup \mathcal{F}$ terminates. Then the skolem chase decides fact entailment over $\Sigma \cup \mathcal{F}$ in polynomial time with respect to the size of $\mathcal{F}$.*

In other words, the data complexity of fact entailment over rules with a universally terminating skolem chase is in PTime. This data complexity is strictly smaller than for MARPL rules (Theorem 1), i.e., we cannot hope to encode MARPL theories in skolem-chase terminating rules. This restriction applies to virtually every known chase termination criterion, including recent approaches that are not specific to the skolem chase but also feature PTime data complexity [15].

Surprisingly, however, this restriction is not inherent to other versions of the chase. As has recently been observed, there are universally standard-chase terminating rule sets with non-elementary data complexity [36]. Therefore, there does not seem to be any principled reason why MARPL-style expressive power should not be in reach for existing rule engines. Indeed, we conjecture that MARPL can be translated into existential rules with a suitable encoding. The standard chase is implemented in both RDFox [7] and VLog [51], hence this translation might also allow MARPL reasoning in practice.

A MARPL reasoner would be a first step towards realising the vision described in this paper. It would enable users to replace custom data extraction and projection software with a declarative logical description of a KG view. Further expressive features will be useful to truly cover a significant part of current preprocessing implementations:

(1) Negation: The reasoner needs to be able to check for the absence of data in the KG, e.g., to determine which entities have no time of dissolution.
(2) Quantitative data: Basic range comparisons on input data are important for filtering, e.g., to check if a statement is valid at a specific point in time (within the given validity interval).
(3) Aggregates: especially maximum and minimum values can be of importance, e.g., to find the most recent statement of a certain property.

It would often suffice if these features were supported on input data, which is not affected by recursive computation, and therefore does not raise semantic issues related to, e.g., the use of negation in recursive specifications. Nevertheless, view definitions might benefit from the use of recursion in other places, e.g., to compute excerpts of the KG by computing all entities that are reachable through some property (example: select all monarchs and all of their relatives).

The facts entailed by such recursive view definitions can then be used with other AI methods for data mining or machine learning. This could be done by exporting the entailments, using them as input to other approaches, and re-importing the results into the logical knowledge base. Further logical rules could then be used to draw conclusions that relate to the original data model of the KG.

It would be desirable to achieve a tighter integration that avoids the need for this separate export and import step. A practical way of doing this could be to access the other AI methods through built-in predicates that can be used in rules. This leads to hybrid rule-based systems that can "call" external libraries or sub-systems for optimisation and learning tasks. Some recent rule engines have started to explore this approach, examples being LogicBlox [3] and Vadalog [6].

While built-in functions are not uncommon in rule-based systems, the functions needed here would in fact be aggregates in the sense that they would act on sets of facts (representing views) rather than on single tuples. There are also other conceivable ways of achieving such integration, for example using higher-order built-ins as proposed by Eiter et al. for the case of answer set programming [19]. The main requirement is that the built-in function can receive a complete set of facts, e.g., an unlabelled directed graph that was extracted from the KG.

The use of built-in functions allows us to treat methods from machine learning and data analytics as black boxes. This approach is flexible and versatile, but it does not exploit the characteristics of specific built-in functions. The latter may have significant advantages in terms of overall performance, as has been found by recent works that have explored this idea for (non-recursive) database queries [31]. Similar approaches suggest themselves in the context of rule-based systems.

Accessing data mining and machine learning methods through built-in functions should be contrasted with approaches that attempt to increase the expressivity of the logic in order to be able to implement data analysis algorithms in rules. For example, Aberger et al. present an architecture for computing some recursive network analysis algorithms, such as PageRank centrality, using rule-based specifications [1]. While such iterative algorithms are compatible with the recursive reasoning in rule engines, they require a different control strategy, since they can usually not be iterated until a fixed point is reached but only until some sufficient precision has been obtained. Another approach of combining numeric optimisation with rule reasoning was proposed by Kaminski et al. [30]. It has the potential of integrating linear optimisation with rules in a fully declarative way, but other algorithms such as PageRank seem to be out of scope. We believe that built-in functions can be a middle ground between such a fully integrated approach (which must be carefully restricted to remain within the limits of computability) and today's loose combination of several programs that does not follow any underlying principle.

6 Summary and Outlook

Knowledge graphs are characterised by their heterogeneous and multi-faceted information content, which covers a range of topics at different levels of detail. Significant feature engineering and sampling is required to apply AI techniques from data mining or machine learning to such resources successfully. We have argued that rule-based knowledge representation can be used to achieve this preprocessing in a declarative, principled way.

Besides the immediate benefit of an integrated system that replaces the makeshift processing pipelines in many current works, this approach has the potential of paving the way towards an explainable AI. Indeed, the powerful but opaque methods of statistical optimisation and machine learning would be invoked and interpreted through the knowledge-based framework of logical rules, which is specified by humans and therefore more explicit and easier to verify than learned models.

Looking further ahead, we can see an opportunity for expanding this approach towards new system architectures and design principles that lead to an artificial intelligence that is not just powerful and efficient but also reliable, predictable, and safe.

Acknowledgements This work is partly supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) in project number 389792660 (TRR 248, Center for Perspicuous Systems), CRC 912 (Highly Adaptive Energy-Efficient Computing, HAEC), and Emmy Noether grant KR 4381/1-1.

References

1. Aberger, C.R., Tu, S., Olukotun, K., Ré, C.: EmptyHeaded: A relational engine for graph processing. In: Özcan, F., Koutrika, G., Madden, S. (eds.) Proc. 2016 ACM SIGMOD Int. Conf. on Management of Data. pp. 431–446. ACM (2016)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
3. Aref, M., ten Cate, B., Green, T.J., Kimelfeld, B., Olteanu, D., Pasalic, E., Veldhuizen, T.L., Washburn, G.: Design and implementation of the LogicBlox system. In: Sellis, T.K., Davidson, S.B., Ives, Z.G. (eds.) Proc. 2015 ACM SIGMOD Int. Conf. on Management of Data. pp. 1371–1382. ACM (2015)
4. Baget, J., Leclère, M., Mugnier, M., Rocher, S., Sipieter, C.: Graal: A toolkit for query answering with existential rules. In: Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., Roman, D. (eds.) Proc. 9th Int. Web Rule Symposium (RuleML'15). LNCS, vol. 9202, pp. 328–344. Springer (2015)
5. 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)
6. Bellomarini, L., Sallinger, E., Gottlob, G.: The Vadalog system: Datalog-based reasoning for knowledge graphs. Proc. VLDB Endowment **11**(9), 975–987 (2018)
7. Benedikt, M., Konstantinidis, G., Mecca, G., Motik, B., Papotti, P., Santoro, D., Tsamoura, E.: Benchmarking the chase. In: Sallinger, E., den Bussche, J.V., Geerts, F. (eds.) Proc. 36th Symposium on Principles of Database Systems (PODS'17). pp. 37–52. ACM (2017)
8. Benedikt, M., Leblay, J., Tsamoura, E.: PDQ: proof-driven query answering over web-based data. Proc. VLDB Endowment **7**(13), 1553–1556 (2014)

9. Bonifati, A., Ileana, I., Linardi, M.: Functional dependencies unleashed for scalable data exchange. In: Baumann, P., Manolescu-Goujot, I., Trani, L., Ioannidis, Y.E., Barnaföldi, G.G., Dobos, L., Bányai, E. (eds.) Proc. 28th Int. Conf. on Scientific and Statistical Database Management (SSDBM'16). pp. 2:1–2:12. ACM (2016)
10. Borchmann, D.: Towards an error-tolerant construction of $\mathcal{EL}^{\perp}$-ontologies from data using formal concept analysis. In: Cellier, P., Distel, F., Ganter, B. (eds.) Proc. 11th Int. Conf. on Formal Concept Analysis (ICFCA'13). LNCS, vol. 7880, pp. 60–75. Springer (2013)
11. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (eds.) Proc. 27th Annual Conf. Neural Information Processing Systems (NIPS 2013). pp. 2787–2795 (2013)
12. Burgstaller-Muehlbacher, S., Waagmeester, A., Mitraka, E., Turner, J., Putman, T., Leong, J., Naik, C., Pavlidis, P., Schriml, L., Good, B.M., sSu, A.I.: Wikidata as a semantic framework for the Gene Wiki initiative. Database **2016**, baw015 (2016)
13. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Paredaens, J., Su, J. (eds.) Proc. 28th Symposium on Principles of Database Systems (PODS'09). pp. 77–86. ACM (2009)
14. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning **39**(3), 385–429 (2007)
15. Carral, D., Dragoste, I., Krötzsch, M.: Restricted chase (non)termination for existential rules with disjunctions. In: Sierra, C. (ed.) Proc. 26th Int. Joint Conf. on Artificial Intelligence (IJCAI'17). pp. 922–928. ijcai.org (2017)
16. 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)
17. 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/
18. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Lenzerini, M., Lembo, D. (eds.) Proc. 27th Symposium on Principles of Database Systems (PODS'08). pp. 149–158. ACM (2008)
19. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 90–96. Professional Book Center (2005)
20. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theoretical Computer Science **336**(1), 89–124 (2005)
21. Ferrucci, D.A., Brown, E.W., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J.M., Schlaefer, N., Welty, C.A.: Building Watson: An overview of the DeepQA project. AI Magazine **31**(3), 59–79 (2010)
22. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1997)
23. Geerts, F., Mecca, G., Papotti, P., Santoro, D.: That's all folks! LLUNATIC goes open source. PVLDB **7**(13), 1565–1568 (2014)
24. González, L., Hogan, A.: Modelling dynamics in semantic web knowledge graphs with formal concept analysis. In: Champin, P., Gandon, F.L., Lalmas, M., Ipeirotis, P.G. (eds.) Proc. the 2018 World Wide Web Conference (WWW'18). pp. 1175–1184. ACM (2018)
25. Hanika, T., Marx, M., Stumme, G.: Discovering implicational knowledge in Wikidata. In: Proc. 15th Int. Conf. on Formal Concept Analysis (ICFCA'19). LNCS, Springer (2019)
26. Harris, S., Seaborne, A. (eds.): SPARQL 1.1 Query Language. W3C Recommendation (21 March 2013), available at http://www.w3.org/TR/sparql11-query/

27. Hernández, D., Hogan, A., Krötzsch, M.: Reifying RDF: what works well with wikidata? In: Liebig, T., Fokoue, A. (eds.) Proc. 11th Int. Workshop on Scalable Semantic Web Knowledge Base Systems. CEUR Workshop Proceedings, vol. 1457, pp. 32–47. CEUR-WS.org (2015)
28. Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G.: Learning rules from incomplete KGs using embeddings. In: van Erp, M., Atre, M., López, V., Srinivas, K., Fortuna, C. (eds.) Posters & Demonstrations, Industry and Blue Sky Ideas Tracks of the 17th International Semantic Web Conference (ISWC 2018). CEUR Workshop Proceedings, vol. 2180. CEUR-WS.org (2018)
29. 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)
30. Kaminski, M., Grau, B.C., Kostylev, E.V., Motik, B., Horrocks, I.: Foundations of declarative data analysis using limit datalog programs. In: Sierra, C. (ed.) Proc. 26th Int. Joint Conf. on Artificial Intelligence (IJCAI'17). pp. 1123–1130. ijcai.org (2017)
31. Khamis, M.A., Ngo, H.Q., Nguyen, X., Olteanu, D., Schleich, M.: In-database learning with sparse tensors. In: den Bussche, J.V., Arenas, M. (eds.) Proc. 37th Symposium on Principles of Database Systems (PODS'18). pp. 325–340. ACM (2018)
32. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 2656–2661. AAAI Press/IJCAI (2011)
33. Krötzsch, M.: Ontologies for knowledge graphs? In: Artale, A., Glimm, B., Kontchakov, R. (eds.) Proc. 30th Int. Workshop on Description Logics (DL'17). CEUR Workshop Proceedings, vol. 1879. CEUR-WS.org (2017)
34. Krötzsch, M., Marx, M., Ozaki, A., Thost, V.: Attributed description logics: Reasoning on knowledge graphs. In: Lang, J. (ed.) Proc. 27th Int. Joint Conf. on Artificial Intelligence (IJCAI'18). pp. 5309–5313 (2018). https://doi.org/10.24963/ijcai.2018/743
35. 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)
36. Krötzsch, M., Marx, M., Rudolph, S.: The power of the terminating chase. In: Barceló, P., Calautti, M. (eds.) Proc. 22nd Int. Conf. on Database Theory (ICDT'19). LIPIcs, vol. 127, pp. 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2019)
37. 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)
38. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, 2nd ed edn. (2014)
39. Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the most out of Wikidata: Semantic technology usage in Wikipedia's knowledge graph. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.A., Simperl, E. (eds.) Proc. 17th Int. Semantic Web Conf. (ISWC'18). LNCS, vol. 11137, pp. 376–394 (2016)
40. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: Paredaens, J., Su, J. (eds.) Proc. 28th Symposium on Principles of Database Systems (PODS'09). pp. 13–22. ACM (2009)
41. Marx, M., Krötzsch, M., Thost, V.: Logic on MARS: Ontologies for generalised property graphs. In: Sierra, C. (ed.) Proc. 26th Int. Joint Conf. on Artif. Intell. (IJCAI'17). pp. 1188–1194 (2017)
42. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: RDFox: A highly-scalable RDF store. In: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P.T., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) Proc. 14th Int. Semantic Web Conf. (ISWC'15), Part II. LNCS, vol. 9367, pp. 3–20. Springer (2015)

43. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2016)
44. openCypher community: Cypher Query Language Reference, Version 9. available at http://www.opencypher.org/resources (2019)
45. 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/
46. 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)
47. Rudolph, S.: Exploring relational structures via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) Proc. 12th Int. Conf. on Conceptual Structures, (ICCS'04). LNCS, vol. 3127, pp. 196–212. Springer (2004)
48. Simonite, T.: Inside the Alexa-friendly world of Wikidata. WIRED Magazine **27.03** (2019), available online at https://www.wired.com/story/inside-the-alexa-friendly-world-of-wikidata/, accessed 2019-03-16
49. Tanon, T.P., Vrandecic, D., Schaffert, S., Steiner, T., Pintscher, L.: From Freebase to Wikidata: The great migration. In: Bourdeau, J., Hendler, J., Nkambou, R., Horrocks, I., Zhao, B.Y. (eds.) Proc. 25th Int. Conf. on World Wide Web (WWW'16). pp. 1419–1428. ACM (2016)
50. Urbani, J., Jacobs, C., Krötzsch, M.: Column-oriented Datalog materialization for large knowledge graphs. In: Schuurmans, D., Wellman, M.P. (eds.) Proc. 30th AAAI Conf. on Artificial Intelligence (AAAI'16). pp. 258–264. AAAI Press (2016)
51. Urbani, J., Krötzsch, M., Jacobs, C.J.H., Dragoste, I., Carral, D.: Efficient model construction for Horn logic with VLog: System description. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) Proc. 9th Int. Joint Conf. on Automated Reasoning (IJCAR'18). LNCS, vol. 10900, pp. 680–688. Springer (2018)
52. Vrandečić, D., Krötzsch, M.: Wikidata: A free collaborative knowledgebase. Commun. ACM **57**(10) (2014)
53. Wagner, C., Graells-Garrido, E., Garcia, D., Menczer, F.: Women through the glass ceiling: gender asymmetries in Wikipedia. EPJ Data Science **5**(1), 5 (2016)
54. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Brodley, C.E., Stone, P. (eds.) Proc. 28th AAAI Conf. on Artif. Intell. (AAAI'14). pp. 1112–1119. AAAI Press (2014)