Knowledge Graph Reasoning with Logics and Embeddings: Survey and Perspective

Wen Zhang  
School of Software Technology  
Zhejiang University  
Ningbo, China  
zhang.wen@zju.edu.cn

Jiaoyan Chen  
Department of Computer Science  
The University of Manchester  
Manchester, UK  
jiaoyan.chen@cs.ox.ac.uk

Juan Li*  
School of Software Technology  
Zhejiang University  
Ningbo, China  
lijuan18@zju.edu.cn

Zezhong Xu  
College of Computer Science and Technology  
Zhejiang University  
Hangzhou, China  
xuuzzhong@zju.edu.cn

Jeff Z. Pan  
School of Informatics  
The University of Edinburgh  
Edinburgh, UK  
j.z.pan@ed.ac.uk

Huajun Chen*  
College of Computer Science and Technology  
Zhejiang University  
Hangzhou, China  
huajunsir@zju.edu.cn

*Corresponding Authors.

Abstract—Knowledge graph (KG) reasoning is becoming increasingly popular in both academia and industry. Conventional KG reasoning based on symbolic logic is deterministic, with reasoning results being explainable, while modern embedding-based reasoning can deal with uncertainty and predict plausible knowledge, often with high efficiency via vector computation. A promising direction is to integrate both logic-based and embedding-based methods, with the vision of having advantages for both. It has attracted wide research attention with more and more works published in recent years. In this paper, we comprehensively survey these works, focusing on how logics and embeddings are integrated. We first briefly introduce preliminaries, then systematically categorize and discuss works of logic and embedding-aware KG reasoning from different perspectives, and finally conclude and discuss the challenges and further directions.

Index Terms—Knowledge graph reasoning, embedding-based reasoning, logical reasoning.

I. INTRODUCTION

Knowledge representation and reasoning play a critical role in many domains, especially Artificial Intelligence (AI). As one of the ways for knowledge representation, knowledge graph (KG) has a flourishing development in recent years. It represents facts in the form of triples, for example, (Earth, containsIn, Solar System), and can use a vocabulary defined in a schema (also known as ontology) to leverage more expressiveness logics. KG is a simple yet efficient and increasingly popular way of knowledge representation. Many general-purpose KGs, such as Wikidata [1] and YAGO [2], and domain-specific KGs, such as SNOMED Clinical Terms [3] and Product Knowledge Graph [4], are under fast development and widely used. However, the knowledge stored in KGs is often incomplete or with noise, as large-volume KGs are often extracted from multiple sources automatically or semi-automatically [5]. It has big impacts on the credibility and usability of knowledge graphs, thus research on knowledge graph reasoning, aiming to infer missing knowledge or detect inconsistent knowledge, is widely explored. Knowledge graph reasoning can greatly contribute to both in-KG tasks, such as link prediction and query answering, and out-of-KG tasks, such as recommendation and question answering.

Conventional KG reasoning is often based on symbolic logic and is deductive. For example, HermiT [6] is a classic description logic reasoner for OWL\(^1\) ontologies; RDFox [7] is a famous KG storage supporting Datalog rule reasoning. Recently, with the rapid development of deep learning, KG embedding represents entities and relations as vectors (embeddings), and has shown great success in KG reasoning, especially in inductive reasoning without pre-defined logics. Logic-based reasoning is usually interpretable and transferable, while embedding-based reasoning can deal with uncertainty and data noise, and is able to predict non-determined but plausible knowledge. Thus flourishing research has been conducted to integrate logics and embeddings for more robust KG reasoning.

There are several survey papers about both KG and neural-symbolic integration. [8] focuses on combining symbolic reasoning with statistical reasoning. [9] review methods of knowledge reasoning for knowledge graphs and classify them into rule-based reasoning, distributed representation-based reasoning, and neural network-based reasoning. While [10] reviews papers on symbolic, neural, and hybrid reasoning. Unlike these papers, our survey has a more specific topic, focusing on logics and KG embeddings, and has a more fine-grained categorization of these studies, from two general perspectives: (i) injecting logics, such as logical rules and ontological schemas, into embedding learning, and (ii) utilizing KG embeddings for logic reasoning-relevant tasks, such as query answering, theorem proving and rule mining. More importantly, we analyze and compare methods of each category.

\(^1\)Web Ontology Language. https://www.w3.org/OWL/
from some more fine-grained perspectives, including logic types, the existence of pre-defined logics, integration stages, and integration mechanisms. The survey also introduces the challenges and discusses the potential directions. All these are beneficial for future research on KG reasoning as well as KG construction, KG application, and neural-symbolic integration.

The remainder of the paper is organized as follows. Section II briefly introduces the background on knowledge graph, reasoning with logics, and reasoning with embeddings. Section III reviews methods that utilize logics for augmenting embeddings. Section IV reviews methods of using embeddings for supporting reasoning tasks. Section V analyzes the challenges and discusses the future directions.

II. BACKGROUND

a) Knowledge Graph: The term knowledge graph (KG) was first proposed by Google in 2012 to help machines understand real-world entities and their relationships to one another and make search more intelligent2. A knowledge graph could generally be regarded as a multi-relational graph composed of entities as nodes and relations as different types of edges, pointing from one entity node to another. Figure 1 shows a toy example of a knowledge graph created based on wikidata [1]. During the past decade, although there have been trials for a formal definition of knowledge graphs [11], the standard definition hasn’t been reached [8]. A knowledge graph could be defined from coarse to fine, by distinguishing entities, concepts, and literals, or by distinguishing relations and properties.

In this paper, we follow the simplest definition of knowledge graph represented as a set of triples with each triple in the form of (head entity, relation, tail entity), for example (Earth, part of, Inner Solar System). The knowledge graph that we are interested in could be formally defined as \( G = \{ E, R, T \} \) where \( E \) and \( R \) are the set of entities and relations. \( T = \{(h, r, t)|h \in E, r \in R, t \in E\} \) is the set of triples, where \( h \) and \( t \) are head and tail entities respectively. For logic knowledge that could be understood in ontology schema and logic rules that we are also interested in this paper, we regard them as external knowledge for knowledge graphs in order to make the term ‘knowledge graph’ in this paper simple and consistent.

b) Reasoning with Logics: Given a KG \( G \), logical reasoning can be used to infer new implicit knowledge or to detect inconsistencies. Many might be familiar with logical rules. One rule, which can be simply represented as \( H \leftarrow B_1 \land B_2 \land \ldots \land B_n \), means that the head atom \( H \) can be inferred by the body atoms \( B_1, ..., B_n \). For example, isfatherOf(X, Y) \( \leftarrow \) Male(X) \& isParentOf(X, Y) means that if \( X \) is a male, and \( X \) is a parent of \( Y \), then \( X \) is the father of \( Y \). In addition to rules, the Web Ontology Language OWL 2, which is based on Description Logics (DLs), is a key standard schema language of KGs. It is based on the \( SROIQ \) DL [12]. OWL 2 provides rich expressive power, including strong support for datatypes [13] and rules [14]. OWL 2 schema can be used to define class hierarchies, complex classes and relations, domain and range for relations, and more complex schema axioms. Logic languages allow deductive reasoning, such as consistency checking, materialization, and query answering, as well as inductive and abductive reasoning.

c) Reasoning with Embeddings: KG embedding (KGE), as a kind of representation learning technique, aims to represent entities and relations by vectors with their semantics (e.g., relationships) preserved in the vector space. With the embeddings, implicit and new knowledge can be inferred, often with approximation or prediction. Many successful KGE methods, such as TransE [15], ComplEx [16], and RotatE [17], have been developed in the past decade. A KGE method is usually composed of a score function \( \phi() \) which defines how to compute the truth value based on entity and relation embeddings, and a loss function that maximizes the truth value of explicit positive triples and minimizes values of generated negative triples. Take TransE [15] as an example, it makes \(|h+r-t|\) as score function of triples, where \( h, r, \) and \( t \) are vectors of \( h, r, \) and \( t \) in the Euclidean space, respectively. The embeddings are learned based on a margin loss that tries to maximize the margin between the truth value of positive and negative triples. Finally, KGEs can be used to query triples, infer new triples, and discover inconsistency between triples.

III. LOGICS FOR EMBEDDINGS

Though KGE methods have achieved great success, they still suffer from the following problems: 1) they could fail to embed complex semantics such as relationships by logical rules; 2) they usually ignore the ontological schema of a KG as inputs; 3) most of them build black-box models with a shortage of explanation. Integrating logics into embeddings is expected to solve these problems. Before analyzing concrete methods of such KG embedding, we first introduce the perspectives that are considered in categorization and discussion:

a) Logic Types: We consider two mainstream logic forms used in KG reasoning: (1) logic rules and (2) Ontological schema, such as those supported by OWL 2, which can internalize most logic rules, with some exceptions, such as those with loops in the body [14].

2https://blog.google/products/search/introducing-knowledge-graph-things-not/
b) Pre-defined logics: 1) With: injecting predefined logics or their reasoning results into embeddings, and/or applying ontological axioms as constraints. 2) Without: injecting the concept of logics into embeddings, where logics are assumed to exist in the KG while no specific data of logics are used.

c) Integration Stages: Considering the time when logic is injected in learning embeddings, there are three stages: 1) Pre: conducting symbolic reasoning before learning embeddings. The reasoning often impacts training samples such as positive and negative triples. 2) Joint: injecting the logics during embedding learning. This often extends the loss function with logic constraints, by e.g., adding an additional regularizer on some relation embeddings. 3) Post: conducting symbolic reasoning after embeddings are learned, by e.g., jointly constructing a predictive model with both results from embeddings and logics as inputs, or logical constraints as filters.

d) Mechanisms: 1) Data-based: replacing variables in logic expressions with concrete entities and getting new triples, then adding all or part of the new triples into training. 2) Model-based: adding constraints on the embedding of entities and relations included in logic expressions into training. No additional triples are used.

A. Logic Rules for Embeddings

Logic rules are of the form $H \leftarrow B_1 \land B_2 \land \ldots \land B_n$, where $H$ is the rule head and $B_1 \land B_2 \land \ldots \land B_n$ is the rule body with the conjunction of atoms. A typical kind of logic rule is a path rule where the rule body is a path from a head variable to a tail variable in the rule head; for example, $(X,Y) \leftarrow r_1(X,Z) \land r_2(Z,Y)$ is a path rule with a path from variable $X$ to $Y$ as body. Note that the number of atoms in the body is also known as the rule length. There are a few works that inject path rules into embeddings. PTransE [18] is a typical explicit and joint-training method, where path compositional representations calculated with relation embeddings in the path are encouraged to near the relation embedding in rule head in vector space. PTransE applies all high-quality paths during training. Alternatively, RPJE [19] selects the path with the highest confidence to compose the path for each triple and distinguishes paths with length 1 from other paths, for which confidence of rules is not considered during training. Instead of compositional representation regularization, ComplEx-NNE_AER [20] infers constrains on relation embeddings through making $\phi(h, r_1, t) > \phi(h, r_2, t)$ if $r_1(X,Y) \leftarrow r_2(X,Y)$, and the larger $\phi(h,r,t)$ is, the more possible $(h,r,t)$ to be a positive one. Following ComplEx-NNE_AER, SLRE [21] adapts relation embedding constraints to longer path rules.

Methods mentioned above inject pre-defined rules via regularizing relation embeddings during training. They are specific to KGE methods. One more general solution is using grounding, which replaces variables in each rule with concrete entities, infers implicit triples, and generates additional triples for KGEs’ training. For example, KALE [22] models groundings by t-norm fuzzy logics which gives a truth score for each grounding based on the truth value of all atoms. It trains groundings with negative sampling together with existing triples. KALE conducts grounding before training and injects them one time. Alternatively, multiple times or iterative injection ensures more flexibility for a model. For example, in each training iteration, RUGE [23] and IterE [24] predicts labels of unlabeled triples in groundings based on t-norm fuzzy logics, and uses labeled triples and post-labeled triples for training. The iterative manner enables the model to predict labels for unlabeled triples dynamically based on embeddings. These data-based methods need to materialize each rule, resulting in a massive number of groundings, thus they do not scale well to large KGs. However, the KGE-free property makes them continuously benefit from the development of KGEs. Post-training injection solutions are also KGE-free. For example, [25] proposes to frame an Integer Linear Programming problem to combine rules and KGEs, where rules are translated into conditional constraints during training, and scores from well-trained KGEs are inputs.

Apart from the satisfaction of rules, TARE [26] emphasizes the properties of transitivity and asymmetry of rules which makes the order of relations in rules matter, and it models the order of relation types in logic rules by the component-wise inequality.

B. Ontological Schemas for Embeddings

Ontological schemas, which are often defined by languages such as OWL and RDF Schema, describe high-level semantics (meta information) of KGs. We next survey methods that inject class hierarchies, relation hierarchies, and relation properties.

a) Class Hierarchies: Class hierarchies classify entity types, denoting entities as instantiations of classes. There are two tasks for injecting class hierarchies, encoding the types of entities and encoding hierarchies of entity types.

To encode entity types, with entity types given, one kind of method learns an embedding for each type and adds regularization to these embeddings. For example, TAGAT [30] regularizes entity embeddings to be close to their corresponding type embeddings, and also closes the embeddings of entities that belong to the same type. RETA-Grader [32] uses type embeddings in entity-typed triples $(h_{\text{type}}, r, t)$ for each triple $(h, r, t)$, and concatenate embeddings in these two types of triples as inputs for triple scoring. Without entity types given, a common way is to assume a type for each entity and learn an embedding for it. For example, TypeDM [31] uses the assumed type embedding of an entity and domain(range) embeddings of a relation to calculate the satisfaction of the domain(range) of a relation. Another way is to infer the candidate types of each entity based on ontological information. For example, iterefinE [33] applies the inferred types to refine the KG data and regularize the learning of embeddings.

In order to inject entity type hierarchies into embeddings, there are methods with and without pre-defined hierarchies. With class hierarchies given, one kind of method combines the type hierarchy of each entity with its embedding. For example, TKRL [28] encodes type hierarchies into a projection...
matrix, and injects type hierarchies into entity embeddings by projecting them with projection matrices. Without pre-defined class hierarchies, HAKE [29] proposes to map entities into a polar coordinate system, where concentric circles can naturally reflect hierarchies. After training, implicit entity hierarchies could be decoded from entity embeddings.

b) Relation Hierarchies: Relation hierarchies contain subsumption relationships between relations; for example, hasFather is a sub-relation of hasParents.

Without pre-defined relation hierarchies, HRS [34] assumes a three-layer hierarchical relation structure for each relation, including relation clusters, relations, and sub-relations, discovered by the K-means algorithm on relation embeddings from KGE. To encode relation hierarchies, it learns an embedding for each relation cluster and represents relations as the sum of their cluster embedding, relation embedding, and sub-relations’ embedding. Another method TransRHS [40] which does not explicitly assume a multi-layer relation hierarchy, proposes to model each relation as a vector together with a relation-specific sphere. It assumes lower-level relations are with smaller spheres. If a predicted entity lies in the spheres of a lower level relation such as hasFather, then the model will ensure it also lies in the sphere of its parent relations such as hasParent. This embodies the inherent generalization relationships among relations.

c) Relation Properties: Ontological schemas often define quite a few relation properties (and constraints).

First, we introduce properties constraining only one relation that has been considered. For domain and range of relations, TRECAL [35] leverages them by filtering triples in KGs where entities are not compatible with the domain or range of relations. If not pre-defined, TypeDM [31] learns an assumed domain and range embedding for each relation and uses them for constraining entity type embeddings. To model Asymmetric relations, ComplEx [16] proposes to embed KGs in complex vector space, where the commutative property for multiplication is not satisfied. To further model Composition between relations, RotateE [17] proposes to define each relation as a rotation from the head entity to the tail entity in complex vector space. Furthermore, Rot-Pro [37] proposes to model Transitive relations by defining relations as projection and rotation. dORC [38] enables modeling Reflexive, Symmetric, and Transitive relations by disentangling the embedding of each entity as head and tail entities. [25] propose a post-training injection for Cardinality of relations, by framing a Linear programming problem.

Second, we introduce relation properties constraining multiple relations that have been considered, including Equivalent and Inverse. Given these two pre-defined relation properties, [36] injects them via a single constraint on the embedding of the two relations in schemas, based on the vector space assumption in KGEs. Besides applying axiom-based regularization, TransOWL [41] also proposes to add new triples inferred by these schemas into training, which could also be applied to entity type constraints such as equivalentClass and subClassOf. More generally, SIC [39] proposes to use ontological reasoning within their iterative KG completion approach to inject inferred triples to enrich the input KG for embedding, and to filter out schema-incorrect triples via consistency and constraint checking. ReasonKGE [42] follows SIC [39] by using schema-incorrect triples for negative sampling.

3Composition of relations includes multiple relations, we introduce it in this paragraph because RotatE is based on ComplEx.
C. Summary

We summarize methods injecting logics into embedding methods introduced before in Table I. If logics are pre-defined, there are diverse methods with different integration stages and mechanisms. Moreover, in the scenario without pre-defined logics, model-based and post-training injection methods are used. For pre-training injection methods, database-based models are more common, while model-based models are more common for joint training injection. We separately discussed injecting logical rules and ontology schemas for embeddings in this section, considering they are two types of logic languages. However, they are not entirely distinct, and each ontological schema could be rephrased into a logic rule. For example, the composition of relations could be represented as a path rule.

IV. EMBEDDINGS FOR LOGICS

Pure symbolic reasoning with different kinds of logics has been investigated for years and widely applied, but it still suffers from the following problems: 1) the logics must be given as a priori, but constructing logics often relies on domain experts, costing a lot of time and labor, and the logics in most applications are underspecified, limiting the knowledge that can be inferred; 2) it often cannot cope with noise and uncertainty inherent to real-world data; 3) logic reasoning often cannot scale up since some (complex) logics may lead to high time or space complexity. In contrast, embedding-based reasoning is good at inductive reasoning without pre-defined logics, can well address uncertainty, and can scale up by approximation. Thus utilizing embeddings for augmenting KG reasoning tasks attracts wide research attention. We mainly review two kinds of KG reasoning tasks: deductive logic reasoning, which further includes query answering and theorem proving, and inductive logic learning. Before discussing the methods, as in Section III, we first introduce the perspectives for analyzing and categorization:

a) Logic Types: Many logics are considered in each work, such as path rules, numerical rules, path queries, and logic queries constructed by $\land$, $\lor$, $\neg$, and $\neq$. We present logic types following the expression in the logical forms they originated.

b) Pre-defined Logics: Embeddings could be combined to logics 1)With and 2)Without pre-defined logics.

c) Combination Stage: Considering the time when embeddings are combined to logics, there are two stages: 1) Pre: applying embeddings before logic reasoning or learning, such as for candidate selection. 2) Joint: applying embeddings during logic reasoning or learning.

d) Mechanism: 1) Hybrid: after applying embeddings, methods still infer in a symbolic space. 2) Neural: using embeddings following the process of logic reasoning, and all inferences are conducted in vector space.

A. Embeddings for Logic Reasoning

Query answering and theorem proving are two popular logic reasoning tasks where embeddings could be utilized. They are originally implemented by pure deductive symbolic reasoning with predefined logics. With the embeddings, additional knowledge can be predicted for more robust results.

a) Query Answering: Query answering returns correct entities in a KG as answers of a given structured query, where reasoning is usually considered for hidden answers. Conventional query answering is conducted based on structure query languages such as SPARQL\footnote{https://www.w3.org/TR/rdf-sparql-query/} to retrieve and manipulate knowledge in a KG.

Quite a few studies use embeddings to address KG incomplete and noise issues in query answering. In the beginning, simple queries are considered, for example, path queries proposed in [43]. It interprets TransE as implementing a soft edge traversal operator and recursively applies it to predict compositional path queries and is trained on path samples from random walks and explicit triples. Apart from simple path queries, more complex queries, such as conjunctive logical queries and Existential Positive First-Order (EPO), involving multiple unobserved edges, nodes, and even variables are also widely researched with the help of embeddings. GQE embeds entities as a vector, relations as projection operators on entity embeddings, and makes $\land$ in conjunctive logical queries as intersection operators. Through these embeddings and operators, it encodes each query into a vector and gives answers based on the similarity between query and candidate entity embeddings. BiQE [46] translates conjunctive queries into a sequence and encodes them by Transformer Encoder. Query2box [47] can further support disjunctions ($\lor$) in queries via transforming them into Disjunctive Normal Form (DNF), and it defines vector space operators for each type of quantification. Alternatively, CQD [48] only defines projection operators using ComplIEx [16] while applying other quantifications according to t-norms. To support a complete set of first-order logical operations, including conjunction($\land$), disjunction($\lor$), and negation($\neg$), BetaE [49] and ConE [50] propose to embed entities and queries as Beta distributions and sector-cones respectively, on which projection, intersection and negation operator are defined. NewLook [51] further supports queries including Difference($\neq$) by logical operations as flexible neural networks.

Some studies use embeddings to improve the efficiency of query answering, especially for those queries with complex logics. For example, INS-ES [44], running the data-driven inference algorithm INS on Markov Logic Network (MLN) for symbolic reasoning, uses embeddings from TransE to generate a much smaller candidate set for subsequent fact inference in INS.

b) Theorem Proving: Another task of logic reasoning is theorem proving, automatically inferring triples given a set of facts and predefined logic rules. Conventional theorem proving methods are based on different logic languages, such as Prolog, Datalog, and OWL, which are vulnerable to incomplete and noisy KGs. Differentiable theorem proving using embeddings overcomes the limits of symbolic provers on
generalizing to queries with similar but not identical symbols. With NTP [52] as an example, it enables Prolog to learn embeddings and similarities between entities and relations in a KG. It keeps the variable binding symbolic following the inference process of Prolog but compares symbols using their embeddings rather than identical symbols. It could learn without predefined domain-specific rules and seamlessly reason with them. The core process of NTP follows steps of symbolic logic reasoning that requires enumerating and scoring all bounded-depth proof paths for a given goal, thus NTP is inefficient on large KGs. Thus in some works, embeddings are also used to improve the efficiency of the differentiable prover. For example, GNTP [53] uses fact embeddings to select the top nearest neighbor facts for proving sub-goals, and also uses relation embeddings to select top rules to be expanded. Another method CTP [54] uses a key-value memory network, conditioned on the goal of proving and embeddings of relations and constants, to dynamically generate a minimal set of rules to consider at each reasoning step.

B. Embeddings for Logic Learning

Logic learning is to learn patterns from KGs and discover potential (and probabilistic) logics such as schemas and logic rules. Conventional methods like AMIE [61] and AnyBURL [62] are symbolic-based. They determine structures of rules via random walking and adding atoms based on KGs, and measure the quality of rules by statistical matrices such as Confidence and Head Coverage. While statistical matrices might be misleading due to the incompleteness of and noise in KGs, thus it is difficult to learn high-quality rules from the explicit triples alone. Embeddings are widely used in logic learning to overcome incompleteness and noise issues. RuLES [55] adds confidential triples using embedding models for quality extension of KGs. It iteratively extends rules induced from a KG through feedback from embedding models and evaluates the quality of rules on the origin KG and extended KG. ProPPR+MF [57] reconstructs the transition of proofs in ProPPR [63], a stochastic extension of Prolog, based on embeddings of first-order logics through matrix factorization.

Embeddings have also been utilized to improve the efficiency of rule learning. RLvLR [56] uses embeddings to guide and prune the search during rule mining, where embeddings are learned on the subgraph sampled for target predicates with RESCAL [64]. While DistMult [65] and IterE [24] alternatively use relation embeddings to calculate the confidence of rules. Another type of method is differentiable rule mining, learning rules in an end-to-end differentiable manner in vector space. They are inspired by TensorLog [66], a differentiable probabilistic logic. Tensorlog establishes a connection between inference using first-order rules and sparse matrix multiplication, and enables certain types of logical inference tasks to be compiled into a sequence of differentiable numerical operations on matrices. Differentiable rule mining methods use a module containing relation embeddings to learn the weights for each operation of a rule used in TensorLog, for example, NeuralLP [58] uses an attention-based neural controller system for weight generation, and DRUM [59] applies a low-rank approximation. Besides path rules, Neural-Num-LP [60] further enables mining rules with numerical features in a differentiable framework.

C. Summary

We summarize methods applying embeddings for logic reasoning and learning introduced before in Table II. Embeddings could be combined with pre-defined logics either before(Pre) or during(Joint) the process of logic reasoning and learning. And without pre-defined logics, embeddings usually are applied jointly. If the integration stage of embeddings is Joint, there are both hybrid and neural methods, and only hybrid methods for Pre. Methods that combine embeddings in a Post manner haven’t been proposed yet.

V. CONCLUSION AND DISCUSSION

This paper presents a literature review for KG reasoning with logics and embeddings. We divided the integrated methods into integrating logics (logic rules and ontological
schemas) into embeddings and integrating embeddings to logic-based reasoning tasks (query answering, theorem proving, and rule mining). Moreover, we analyzed those methods from four perspectives: logic types, the existence of pre-defined logics, integration stages, and integration mechanisms.

Next, we discuss challenges on this topic from four perspectives: logic diversity, explainability, benchmark, and application.

a) Logic diversity: One critical challenge of logic and embedding integration lies in the diversity of logics. The majority of current methods only consider specific kinds of logics, such as path rules, entity types (classes), class hierarchies, relation hierarchies, and relation properties (see Section III), or specific logic quantifications such as conjunction, disjunction, and negation (see Section IV). Only a few methods, like SIC, ReasonKGE, and TransOWL, support general ontological schemas. Since a KG is often equipped with different kinds of logics, e.g., rules or ontological schemas, it becomes a significant challenge to support and integrate all or most of them simultaneously.

To this end, we think there are two promising directions. One is to explore more logic forms, such as rules with constants, universal quantification (∀), disjointness, and more. Another is to research general frameworks that can simultaneously support different kinds of logics and is independent of KG embedding methods.

b) Explainability: Another critical challenge is making integrated methods more explainable. Most methods in Section III focus on improving the model’s expressiveness, which does not change the black-box nature of embedding methods. Methods in Section IV enabling logic reasoning in vector space decrease the transparency of logic-based methods because the intermediate results are represented as embeddings that are understandable by humans if and only if the meaning of these embeddings are properly interpreted, which is difficult in most cases. While for further AI applications, systems with higher safety, trust, and fairness are expected. Improving the transparency of integrated methods could ensure their broader applications in the future.

In order to achieve this, potential directions may lie in the following three directions. First, for black-box models such as KGs, inject not only semantics of logics but also reasoning steps into models, which extends them from one-step to multi-step reasoning models with interpretable intermediate results. Second, for multi-step models with embeddings as intermediate results, improve the interpretability of intermediate embeddings, especially generating a set of symbolic representations that is easy for humans to understand, instead of similarity intuitions between embeddings. Third, integrate logics and embeddings through explainable machine learning methods to help preserve the transparency of logic-based methods and the robustness of embedding-based methods.

c) Benchmark: There is a shortage of resources for evaluating KG reasoning with both logics and embeddings. Commonly used benchmarks for KGs’ evaluation, such as WN18RR, FB15k-237, and NELL, are subsets sampled from one or multiple domains in large KGs. When creating these datasets, the primary goal is making them suitable for supervised learning settings, which do not ensure and explore diverse logic patterns contained in the dataset. While many works injecting logics in embeddings or combining embeddings for logic reasoning are evaluated on these datasets, which could not reflect the capability of methods once the logic patterns they are concerned are missing in the dataset. Therefore, benchmarks containing diverse pre-defined logics or logic patterns in triples deserved to be proposed.

Also, SIC [39] argues that experimental results show that the existing correctness notion based on the silver standard is highly questionable. Good results from the silver standard often cannot be transferred to other knowledge graphs beyond the benchmark KGs reported in papers. Instead, in the absence of large-scale human evaluations, schema correctness [39] is more promising.

d) Application: Apart from KG reasoning tasks, there are many KG applications, which have been proven could benefit from logics(embbedings) in KGs, for example, information extraction [67], recommender system [68], and image classification [69], especially low-resource learning [70]. However, these tasks have been rarely explored with both logics and embeddings. Recently, inspired by the concept of pre-training and fine-tuning of language models, which is powerful in many downstream natural language processing tasks, KGs are also extended to pre-trained KG models [4] to be used in many KG applications. Thus in the future, applying both logics and embeddings into more KG applications and pre-trained KG models deserves attention.

VI. ACKNOWLEDGMENTS

This work is founded by National Natural Science Foundation of China (NSFC62306276/NSFCU23B2055/NSF CU19B2027), Zhejiang Provincial Natural Science Foundation of China (No. LQ23F020017), Yongjiang Talent Introduction Programme (2022A-238-G), and Fundamental Research Funds for the Central Universities (226-2023-00138).

REFERENCES

[1] D. Vrandecic and M. Krötzsch, “Wikidata: a free collaborative knowledgebase,” Commun. ACM, vol. 57, no. 10, pp. 78–85, 2014.
[2] T. Rebele, F. M. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, “YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames,” in ISWC (2), ser. Lecture Notes in Computer Science, vol. 9982, 2016, pp. 177–185.
[3] M. Q. Stearns, C. Price, K. A. Stackman, and A. Y. Wang, “SNOMED clinical terms: overview of the development process and project status,” in AMIA. AMIA, 2001.
[4] W. Zhang, C. M. Wong, G. Ye, B. Wen, W. Zhang, and H. Chen, “Billion-scale pre-trained e-commerce product knowledge graph model,” in ICDE, 2021.
[5] B. Xue and L. Zou, “Knowledge graph quality management: a comprehensive survey,” IEEE Transactions on Knowledge and Data Engineering, 2022.
[6] B. Glimm, I. Horrocks, B. Motik, G. Stoiilos, and Z. Wang, “Hermit: An OWL 2 reasoner,” J. Autom. Reason., 2014.
[7] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu, and J. Banerjee, “Rdfox: A highly-scalable RDF store,” in ISWC, 2015.
[8] W. Li, G. Qi, and Q. Ji, “Hybrid reasoning in knowledge graphs: Combining symbolic reasoning and statistical reasoning,” Semantic Web, 2020.
[9] X. Chen, S. Jia, and Y. Xiang, “A review: Knowledge reasoning over knowledge graph,” Expert Syst. Appl., vol. 141, 2020.
[10] J. Zhang, B. Chen, L. Zhang, X. Ke, and H. Ding, “Neural, symbolic and neural-symbolic reasoning on knowledge graphs,” AI Open, 2021.
[11] L. Ehrlinger and W. Wölt, “Towards a definition of knowledge graphs,” in SEMANTiCS (Posters, Demos, SoftCCESS), ser. CEUR Workshop Proceedings, vol. 1695. CEUR-WS.org, 2016.
[12] I. Horrocks, O. Kutz, and U. Sattler, “The even more irresistible sroiq,” in KR2006.
[13] J. Z. Pan and I. Horrocks, “OWL-Eu: Adding Customised Datatypes into OWL,” Journal of Web Semantics, 2006.
[14] M. Krotzsch, S. Rudolph, and P. Hitzler, “Description logic rules,” 2008.
[15] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in NIPS, 2013.
[16] T. Trouillon, J. Welbl, S. Riedel, E., and G. Bouchard, “Complex embeddings for simple link prediction,” in ICML, 2016.
[17] Z. Sun, Z. Deng, J. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” in ICLR, 2019.
[18] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, “Modeling relation paths for representation learning of knowledge bases,” in EMNLP, 2015.
[19] G. Niu, Y. Zhang, B. Li, P. Cui, S. Liu, J. Li, and X. Zhang, “Rule-guided compositional representation learning on knowledge graphs,” in AAAI, 2020.
[20] B. Dng, Q. Wang, B. Wang, and L. Guo, “Improving knowledge graph embedding using simple constraints,” in ACL, 2018.
[21] S. Guo, L. Li, Z. Hui, L. Meng, B. Ma, W. Liu, L. Wang, H. Zhai, and H. Zhang, “Knowledge graph embedding preserving soft logical regularity,” in CIKM, 2020.
[22] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, “Jointly embedding knowledge graphs and logical rules,” in EMNLP, 2016.
[23] —, “Knowledge graph embedding with iterative guidance from soft rules,” in AAAI, 2018.
[24] W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, A. Bernstein, and H. Chen, “Iteratively learning embeddings and rules for knowledge graph reasoning,” in WWW, 2019.
[25] Q. Wang, B. Wang, and L. Guo, “Knowledge base completion using embeddings and rules,” in IJCAI, 2015.
[26] M. Wang, E. Rong, H. Zhuo, and H. Zhu, “Embedding knowledge graphs based on transitivity and asymmetry of rules,” in PAKDD, 2018.
[27] F. Wu, J. Song, Y. Yang, X. Li, Z. M. Zhang, and Y. Zhuang, “Structured embedding via pairwise relations and long-range interactions in knowledge base,” in AAAI, 2015.
[28] R. Xie, Z. Liu, and M. Sun, “Representation learning of knowledge graphs with hierarchical types,” in IJCAI, 2016.
[29] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, “Learning hierarchy-aware knowledge graph embeddings for link prediction,” in AAAI, 2020.
[30] Y. Wang, H. Wang, J. He, W. Lu, and S. Gao, “TAGAT: type-aware graph attention networks for reasoning over knowledge graphs,” Knowl. Based Syst., vol. 233, p. 107500, 2021.
[31] P. Jain, P. Kumar, Mausam, and S. Chakrabarti, “Type-sensitive knowledge graph base inference without explicit type supervision,” in ACL, 2018.
[32] P. Rosso, D. Yang, N. Ostapuk, and P. Cudré-Mauroux, “RETA: A schema-aware, end-to-end solution for instance completion in knowledge graphs,” in WWW, 2021.
[33] S. Aorl, S. Bedathur, M. Ramathan, and D. Sharma, “Itereine: Iterative KG refinement embeddings using symbolic knowledge,” in AKBCC, 2020.
[34] Z. Zhang, F. Zhang, M. Qiu, F. Lin, and Q. He, “Knowledge graph embedding with hierarchical relation structure,” in EMNLP, 2018.
[35] K. Chang, W. Yih, B. Yang, and C. Meek, “Typed tensor decomposition of knowledge bases for relation extraction,” in EMNLP, 2014.
[36] P. Minervini, L. Costabello, E., V. and P. Vandenbussche, “Injecting background knowledge into embedding models for predictive tasks on knowledge graphs,” in ISWC, 2021.
[37] M. Nickel, V. Tresp, and H. Kriegel, “A three-way model for collective learning on multi-relational data,” in IJML, 2011.
[38] N. Jain, T. Tran, M. H. Gad-Elrab, and D. Stepanova, “Improving knowledge graph embeddings with ontological reasoning,” in ISWC, 2021.
[39] K. Guo, J. Miller, and P. Liang, “Traversing knowledge graphs in vector space,” in EMNLP, 2015.
[40] Z. Wei, J. Zhao, K. Liu, Z. Qi, Z. Sun, and G. Tian, “Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances,” in CIKM, 2015.
[41] W. L. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec, “Embedding logical queries on knowledge graphs,” in NeurIPS, 2018.
[42] B. Kotnis, C. Lawrence, and M. Niepert, “Ans. complex queries in KGs with bidirectional sequence encoders,” in AAAI, 2021.
[43] H. Ren, W. Hu, and J. Leskovec, “Query2box: Reasoning over knowledge graphs in vector space using box embeddings,” in ICLR, 2020.
[44] E. Arakelyan, D. Daza, P. Minervini, and M. Cochez, “Complex query answering with neural link predictors,” in ICLR, 2021.
[45] H. Ren and J. Leskovec, “Beta embeddings for multi-hop logical reasoning in knowledge graphs,” in NeurIPS, 2020.
[46] Z. Zhang, J. Wang, J. Chen, S. Ji, and F. Wu, “Cone: Cone embeddings for multi-hop reasoning over knowledge graphs,” in NeurIPS, 2021.
[47] L. Liu, B. Du, H. Ji, C. Zhai, and H. Tong, “Neural-answering logical queries on knowledge graphs,” in KDD, 2021.
[48] T. Rocktäschel and S. Riedel, “End-to-end differentiable proving,” in NIPS, 2017.
[49] P. Minervini, M. Bosnjak, T. Rocktäschel, S. Riedel, and E. Grefenstette, “Differentiable reasoning on large knowledge bases and natural language,” in AAAI, 2020.
[50] P. Minervini, S. Riedel, P. Stenetorp, E. Grefenstette, and T. Rocktäschel, “Learning reasoning strategies in end-to-end differentiable proving,” in ICML, 2020.
[51] T. Ho, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, and G. Weikum, “Rule learning from knowledge graphs guided by embedding models,” in ISWC, 2018.
[52] P. G. Omran, K. Wang, and Z. Wang, “Scalable rule learning via learning representation,” in IJCAI, 2018.
[53] W. Y. Wang and W. W. Cohen, “Learning first-order logic embeddings via matrix factorization,” in IJCAI, 2016.
[54] F. Yang, Z. Yang, and W. W. Cohen, “Differentiable learning of logical rules for knowledge base reasoning,” in NIPS, 2017.
[55] A. Sadeghian, M. Armandpour, P. Ding, and D. Wang, “DRUM: end-to-end differentiable rule mining on knowledge graphs,” in NeurIPS, 2019.
[56] P. Wang, D. Stepanova, C. Domokos, and J. Z. Kolter, “Differentiable learning of numerical rules in knowledge graphs,” in ICLR, 2020.
[57] L. Galarraga, C. Telfsoudi, K. Hose, and F. M. Suchanek, “Fast rule mining in ontological knowledge bases with AMIE+,” VLDB J., 2015.
[58] C. Meilicke, M. Chekol, D. Ruffinelli, and H. Stuckenschmidt, “Anytime-bottom-up rule learning for knowledge graph completion,” in IJCAI, 2019.
[59] W. Y. Wang, K. Mazaitis, and W. W. Cohen, “Programming with personalized pagerank: a locally groundable first-order probabilistic logic,” in CIKM, 2013.