

Neuro-Symbolic RDF and Description Logic Reasoners: The State-Of-The-Art and Challenges

Gunjan Singh¹, Sumit Bhatia², and Raghava Mutharaju¹

¹ Knowledgeable Computing and Reasoning Lab, IIIT-Delhi, India
 {gunjans, raghava.mutharaju}@iiitd.ac.in

² Media and Data Science Research Lab, Adobe Inc., Delhi, India
 sumit.bhatia@adobe.com

Abstract. Ontologies are used in various domains, with RDF and OWL being prominent standards for ontology development. RDF is favored for its simplicity and flexibility, while OWL enables detailed domain knowledge representation. However, as ontologies grow larger and more expressive, reasoning complexity increases, and traditional reasoners struggle to perform efficiently. Despite optimization efforts, scalability remains an issue. Additionally, advancements in automated knowledge base construction have led to the creation of large and expressive ontologies that are often noisy and inconsistent, posing further challenges for conventional reasoners. To address these challenges, researchers have explored neuro-symbolic approaches that combine neural networks' learning capabilities with symbolic systems' reasoning abilities. In this chapter, we provide an overview of the existing literature in the field of neuro-symbolic deductive reasoning supported by RDF(S), EL, ALC, and OWL 2 RL, discussing the techniques employed, the tasks they address, and other relevant efforts in this area.

1 Introduction

Ontologies are used to represent knowledge in several domains, such as healthcare, geoscience, IoT, and e-commerce. In recent years, there has been an uptake in expressing ontologies using the Resource Description Framework (RDF)³ [1] and the Web Ontology Language (OWL)⁴ [2], the prominent W3C recommended standards for ontology development. The strength of these knowledge representation languages is automated reasoning, which can be used for various purposes, such as conceptual query answering, data retrieval, and data integration [3]. RDF, while having limited expressivity, is popular because of its simplicity and flexibility to model data in diverse domains [4]. On the other hand, OWL, which

³ <https://www.w3.org/TR/rdf-primer/>

⁴ <https://www.w3.org/TR/owl2-primer/>

This paper is a part of the book titled “Compendium of Neuro-Symbolic Artificial Intelligence” which can be found at the following link: <https://www.iospress.com/catalog/books/compendium-of-neurosymbolic-artificial-intelligence>

is based on Description Logics (DLs)⁵ [5], provides a range of language constructors (Table 1) for capturing the domain knowledge in detail.

Since RDF has limited expressivity, most of the commonly used reasoners⁶ perform quite well on these ontologies. However, as the ontologies grow larger or become more expressive, for example, the ones in OWL 2 DL [2], whose formal underpinning is the highly expressive description logic, *SROIQ* [6], the reasoning complexity also increases. The current reasoners struggle to perform efficiently on such ontologies. Modern reasoning systems typically employ a wide range of optimizations to improve performance. Based on the intended application, these optimizations may fall into different categories (pre-processing, model construction, and reasoning task-dependent optimizations) [7]. However, despite all the research on optimizing the performance of traditional reasoning systems, they still suffer from poor scalability on large and expressive ontologies [8]. Furthermore, with the advancements in automated knowledge base construction⁷, building large and expressive ontologies has become relatively easy. However, such ontologies are often noisy and inconsistent, and the conventional ontology reasoners cannot deal with such ontologies [9].

To come up with a solution to deal with large, expressive, noisy, and inconsistent ontologies, researchers have explored neuro-symbolic approaches [10] that combine the robust learning capabilities of the neural networks and the precise reasoning abilities of the symbolic systems. This chapter presents a survey of existing neuro-symbolic deductive reasoning techniques for RDF and description logics and discusses their benefits, limitations, and scope. Before we do so, let us point out that we exclude works that use machine learning techniques for knowledge graph completion (KGC)⁸, techniques for different ontology completion tasks, such as ontology learning for assisting ontology engineering, enrichment [11,12] and for introducing approximate class definitions that are not part of the standard ontology languages [13] and also predicting assertions in an ontology [14,15]. Specifically, we discuss only those works that focus on deductive reasoning over ontologies. Such techniques infer only logically derivable facts from explicitly stated domain knowledge. The conventional deductive ontology reasoners are based on mathematical logic-based reasoning algorithms such as tableau calculi and inference rule-based methods (Section 2.2).

This chapter is organized as follows. Section 2 gives a brief overview of ontology languages, followed by different traditional and neuro-symbolic reasoning techniques. Although there are several different description logics (*ALC*, *SRI*_F, *SROIQ*, etc.) [6], and several OWL 2 profiles (EL, QL, RL, and DL) [2], the body of work in the neuro-symbolic reasoning category is limited. Hence, we only briefly discuss the ontology languages that are supported by neuro-symbolic reasoning techniques such as RDF(S), *EL*, *ALC*, and OWL 2 RL. Section 3 discusses

⁵ Since there is a one-to-one correspondence between OWL and DLs, we use those two terms interchangeably here.

⁶ <http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/>

⁷ <https://www.akbc.ws/>

⁸ <https://paperswithcode.com/task/knowledge-graph-completion>

the existing literature in the neuro-symbolic deductive reasoning space. In section 3.5, we give a brief overview of the techniques discussed in the chapter in terms of the techniques used, tasks handled by each work, and so on. We also discuss a few complementary efforts in Section 4 that push the research in the ontology reasoning domain, such as reasoner evaluation challenges and ontology benchmarks. We conclude in Section 5.

2 Preliminaries

2.1 Ontology Languages

With the increasing interest in the Semantic Web [16,17] in the early 2000s, ontologies [18] were used in various application domains such as healthcare, geoscience, IoT, and e-commerce to describe knowledge about entities in that particular domain. The basic building blocks of an ontology are entities: **Concepts** (C), **Relations** (R), and **Individuals** (I). **Concepts** denote sets of individuals, **Relations** denote binary relations between individuals, and **individual** names denote a single individual in the domain. For example, an ontology representing the **University** domain might use concepts such as **Student** and **Faculty** to denote the set of all students and faculties in the university, relations such as **teaches** and **hasClassmate** to denote the binary relationships between faculty, course, and students, and individual names such as **alex** and **mary** to denote the individuals alex and mary. Note that, depending on the language, **Concept** may sometimes be referred to as **Class**, **Relation** as **Properties** or **Roles**, and **Individuals** as **Constants**. But, to keep the terminology consistent, we will stick to the terms **Concept**, **Relation**, and **Individual** throughout the survey (irrespective of the language addressed).

Definition 1 (Ontology). An Ontology \mathcal{O} , is a set of statements, called axioms, that is used to describe knowledge about a particular domain. These axioms can be categorized as a triplet = $(TBox, ABox, RBox)$. Let N_C , N_R , and N_I be countable, and pairwise disjoint sets of concept names, relation names, and individual names, respectively. **TBox** is the set of terminological axioms describing the relationships between named concept expressions/concepts, $C \in N_C$. Every terminological axiom is in the form of $A \sqsubseteq B$ (concept inclusion) or $A \equiv B$ (concept equivalence), such that $A, B \in N_C$. **ABox** is the set of assertions describing relationships among individuals $a, b \in N_I$ via relation R (relation assertion) $\in N_R$ as well as instantiation relationships (concept assertion) between elements of N_I and N_C . **RBox** is the set of relational axioms describing different properties of relations.

RDF(S) [1] and OWL 2 (the latest version of OWL) [2] are the two prominent W3C recommended standards for ontology development that provide several language constructors to model the information in a given domain. As mentioned earlier, OWL and DLs have a one-to-one correspondence. OWL 2 is based on DL \mathcal{SROIQ} . Table 1 presents different subsets of constructs offered. Below are a few examples to distinguish between **TBox**, **ABox**, and **RBox** of an ontology.

- **TBox**
 - * Class(UGStudent)
 - UG student is a concept*
 - * UGStudent \equiv Student $\sqcap \exists \text{enrollFor}.\text{Course}$
 - UG student is a student who enrolls in a UG Program*
 - * UGStudent \sqsubseteq Student
 - UG student is a subclass of Student (rdfs:subClassOf)*
- **ABox**
 - * UGStudent(alex)
 - alex is a UG Student (Concept Assertion)*
 - * teaches(mary, alex)
 - mary teaches alex (Relation Assertion)*
- **RBox**
 - * Symmetric(hasClassmate)
 - hasClassmate is a symmetric object property*
 - * Irreflexive(teaches)
 - teaches is an irreflexive object property*

The Resource Description Framework (RDF), which includes RDF Schema (RDFS), is an established and widely used W3C standard for expressing knowledge graphs. An RDF graph, G , is a directed, labeled graph that consists of a set of triples of the form (s, p, o) where the subject s and object o form the graph nodes and the predicate p forms the graph edges, where subject $s \in N_C$ or N_I , and object $o \in N_C, N_I$ or **Literals**, and the predicate $p \in N_R$ denote the binary relation between the nodes s and o .

RDF Schema (RDFS) is a semantic extension of basic RDF that provides several constructs such as **rdfs:subClassOf**, **rdfs:subPropertyOf**, **rdfs:Class**, **rdfs:Resource**, **rdfs:Literal**, and **rdfs:Datatype** to express simple taxonomies and hierarchies among entities. RDF(S) can be used to create lightweight ontologies, and the information in an RDFS ontology can also be represented as a directed, labeled graph. Note that we use RDF(S) to refer to both RDF and RDFS.

OWL, built on top of RDFS, offers more language constructs. The basic building blocks of OWL are concepts (C), relations (R), and individuals (I), which can be put into relationships (called axioms) with each other using different modeling constructs, as shown in Figure 1. The sets C , R , and I must be mutually disjoint. All or a set of these constructs could be utilized depending on the intended application and desired reasoning performance.

OWL 2 [19], the latest version of OWL, is based on DL \mathcal{SROIQ} . Since DLs are a family of first-order logics (FOL) [20], formal semantics are defined for this language. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$, a non-empty set, called the domain of \mathcal{I} (representing all the things existing in the world that \mathcal{I} represent), and $\cdot^{\mathcal{I}}$ is the interpretation function that maps every **individual**, **concept**, and **relation** in N_I , N_C , and N_R , to an element, subset and binary relation in $\Delta^{\mathcal{I}}$ with function $\cdot^{\mathcal{I}}$. For an interpretation I , the function $\cdot^{\mathcal{I}}$ is defined in (Table 1).

OWL 2 has several sublanguages or profiles (EL, QL, RL, DL, and Full) [2], which are overlapping subsets of DL language constructs. In simpler terms, an

DL		Syntax	Semantics	Name
\mathcal{S}	\mathcal{ALC}	\mathcal{EL}	C, R, a	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, $a \in \Delta^{\mathcal{I}}$
			\top	$\Delta^{\mathcal{I}}$
			C(a), R(a,b)	$a^{\mathcal{I}} \in C^{\mathcal{I}}$, $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$
			$C \sqsubseteq D$, $C \equiv D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $C^{\mathcal{I}} = D^{\mathcal{I}}$
			$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
			$\exists R.C$	{ $a \in \Delta^{\mathcal{I}}$ there is a $\langle a, b \rangle \in R^{\mathcal{I}}$ with $b \in C^{\mathcal{I}}$ }
			\perp	ϕ
			$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
			$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
			$\forall R.C$	{ $a \in \Delta^{\mathcal{I}}$ if $\langle a, b \rangle \in R^{\mathcal{I}}$, then $b \in C^{\mathcal{I}}$ }
\mathcal{R}		\mathcal{H}	Trans(R)	if $\langle a, b \rangle \in R^{\mathcal{I}}$ for all $a, b, c \in \Delta^{\mathcal{I}}$ for which $\langle a, b \rangle \in R^{\mathcal{I}}$ and $\langle b, c \rangle \in R^{\mathcal{I}}$
			$R \sqsubseteq S$, $R \equiv S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, $R^{\mathcal{I}} = S^{\mathcal{I}}$
			$R_1 \circ R_2 \sqsubseteq S$	$R^{\mathcal{I}} \circ R_2^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
			U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
			Refl(R), Symm(R),...	if $\langle a, a \rangle \in R^{\mathcal{I}}$ for all $a \in \Delta^{\mathcal{I}}$, if $\langle a, b \rangle \in R^{\mathcal{I}}$ for all $\langle b, a \rangle \in R^{\mathcal{I}}$
			Disj(R, S)	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$
			$\exists R.$ Self	{ $a \in \Delta^{\mathcal{I}}$ $\langle a, b \rangle \in R^{\mathcal{I}}$ }
\mathcal{Q}	\mathcal{N}	\mathcal{F}	\mathcal{O}	{ a }
			\mathcal{I}	$\{a^{\mathcal{I}}\}$
			R^-	$\{\langle b, a \rangle \mid \langle a, b \rangle \in R^{\mathcal{I}}\}$
			Func(R), $\leq 1.$ R	if $a=b$ for all $a, b, c \in \Delta^{\mathcal{I}}$ for which $\langle a, b \rangle \in R^{\mathcal{I}}$ and $\langle a, c \rangle \in R^{\mathcal{I}}$
			$\geq n$ R. C, $\leq n$ R. C	{ $a \in \Delta^{\mathcal{I}}$ { $b \in \Delta^{\mathcal{I}}$ $\langle a, b \rangle \in R^{\mathcal{I}}$ and $b \in C^{\mathcal{I}}$ } $\geq n$ } { $a \in \Delta^{\mathcal{I}}$ { $b \in \Delta^{\mathcal{I}}$ $\langle a, b \rangle \in R^{\mathcal{I}}$ and $b \in C^{\mathcal{I}}$ } $\leq n$ }

Table 1. Syntax and Semantics of the description logic \mathcal{SROTQ}

OWL ontology could be referred to in terms of one of these profiles (for instance, OWL 2 DL ontology) or a (subset) combination of different DL constructs (such as \mathcal{SHIQ} , \mathcal{ALCH} , or \mathcal{SHOIN}).

Different DL constructs (Table 1) have different computational properties. Some are easier to reason over than others. Since the reasoning performance decreases with an increase in expressivity, the primary purpose of the different DLs (formed by various combinations of constructs) is that they enable a balance between ontology expressivity and reasoning complexity depending on the intended application. For example, a lightweight DL, \mathcal{EL} , provides only a few concept constructors, which can be reasoned over in polynomial time w.r.t. the size of the ontology, even in the worst case. Two extensions of \mathcal{EL} , \mathcal{EL}^+ , and \mathcal{EL}^{++} support a few more concept constructors to make the ontology more expressive. \mathcal{EL}^+ additionally supports relation inclusion and composition. On top of that, \mathcal{EL}^{++} further includes concept disjointness, transitive and reflexive relations, along with concept and role assertions. So, for applications where reasoning time is a significant concern, such DLs enable efficient implementations for even large ontologies. More expressive DLs (often termed semi-expressive DLs), such as \mathcal{ALC} , allow for including conjunction, disjunction, negation, existential and universal quantification in the domain description. Finally, highly expressive DL \mathcal{SROIQ} [6] provides an extensive range of language constructs to represent an application domain. Reasoning over such expressive DLs is typically less efficient compared to simpler ones. However, the additional expressiveness can be helpful (and, in some cases, required) in practice to accurately represent the application domain’s constraints.

Each OWL 2 profile [19] consists of different (overlapping) subsets of DL constructs and hence vary in terms of their expressivity and reasoning complexity.

- **OWL 2 EL** (Existential Logic). Based on the DL \mathcal{EL}^{++} and designed to be lightweight and allow polynomial time reasoning.
- **OWL 2 QL** (Query Logic). Based on DL-Lite family of DLs and is designed to give easier access to a large amount of instance data.
- **OWL 2 RL** (Rules Logic). Based on DLP and pD* and designed to interact with rule-based reasoners.
- **OWL 2 DL** (Description Logics). Highly expressive, based on \mathcal{SROIQ} and has the reasoning complexity of N2EXPTIME.
- **OWL 2 Full**. Undecidable.

Each of these profiles has some restrictions in terms of syntax. To illustrate the syntactic limits imposed on the usage of different constructs in each OWL 2 profile, we use existential restriction for the statement, *A faculty is an employee who teaches some course*. The syntax in each OWL 2 profile varies, as shown below.

- In OWL 2 EL, $\text{Faculty} \equiv \text{Employee} \sqcap \exists \text{teaches}.\text{Course}$
- In OWL 2 QL, $\text{Faculty} \sqsubseteq \text{Employee} \sqcap \exists \text{teaches}.\text{Course}$ (existential restrictions are not allowed in the subclass expression)

- In OWL 2 RL, $\text{Employee} \sqcap \exists \text{teaches.Course} \sqsubseteq \text{Faculty}$ (existential restrictions are not allowed in the superclass expression)
- In OWL 2 DL, $\text{Faculty} \equiv \text{Employee} \sqcap =1\text{teaches.Course}$ (since qualified exact cardinalities are supported, we could make the axiom more expressive by writing it using exact cardinality)

2.2 Reasoning

The conventional ontology reasoning methods employ mathematical algorithms, such as tableau and inference rule-based methods, to perform different reasoning tasks. These algorithms, in contrast to learning-based approaches, are capable of deriving every possible conclusion (completeness) with absolute certainty (soundness) in a finite amount of time (termination). However, the conventional methods, despite a range of optimizations, need to scale better for large and expressive ontologies. In addition, these algorithms assume the integrity of the data and can not deal with noisy and inconsistent ontologies. Hence, neuro-symbolic techniques come into the picture because they are often highly scalable and have the added advantage of dealing with noise and providing robust learning abilities.

Before moving on to the existing neuro-symbolic literature, it is crucial to understand the terminologies involved and what exactly (such as conventional reasoning methods or tasks) the neuro-symbolic reasoners try to emulate. This section briefly discusses the most relevant conventional reasoning methods for RDF(S) and DLs, followed by a general idea of neuro-symbolic reasoning in the context of this survey.

Conventional Reasoning Methods The existing reasoning systems support a range of reasoning tasks. Depending on the intended application, these systems could be query-based, which tells whether a given logical expression is a logical consequence of the provided ontology or generates all the deductive inferences in a single run. For the latter case, the tasks that are typically considered standard for ontology reasoning and are supported by most reasoning systems are as follows.

- **Consistency Checking.** Determines whether the given set of axioms is consistent, i.e., no entity violates or contradicts the given ontology definition.
- **Satisfiability.** Determines whether a description of the concept is not contradictory, i.e., checks whether at least one individual satisfies the given concept description.
- **Classification.** Determines the subsumption hierarchies for all the concepts and relations.
- **Realization.** Determines all the concepts to which the individual belongs (often called Concept Membership), especially the most specific ones.
- **Entailment.** Determines all the logical consequences (including classification and realization) that follow from the given ontology, provided the ontology is consistent.

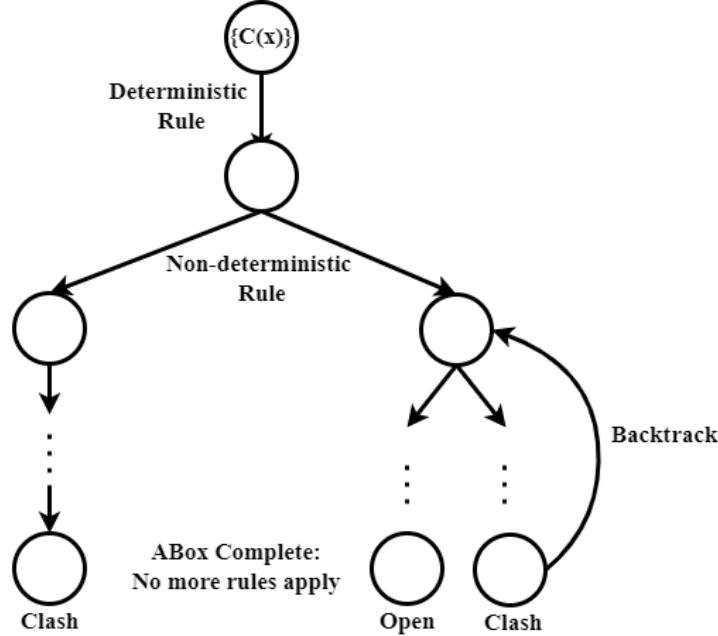
From the point of view of conventional reasoners, the following aspects are typically investigated to measure the effectiveness of the implemented reasoning algorithms.

- **Soundness.** A procedure is called sound if it derives only those consequences that logically follow from the represented axioms.
- **Completeness.** A procedure is called complete if the procedure can derive all the logical consequences that follow from the given ontology.
- **Termination.** A procedure should compute sound and complete answers in a finite amount of time.
- **Scalability.** Indicates how the reasoner performs for varying sizes of ontology (in terms of axiom count and type of language constructs used). The performance is typically measured in terms of reasoning time taken and memory consumed.

Both soundness and completeness are strongly desired features, but in some cases, completeness can be compromised to obtain faster results.

Tableau-Based Methods. For DLs, all the reasoning tasks are reducible to consistency checking. Hence it is sufficient to have a procedure that only determines whether or not ontologies are consistent. Let C be a concept, and I be an interpretation. The aim of the tableau algorithm [6] is to generate a finite interpretation I such that $C^I \neq \emptyset$. Without loss of generality, we assume all concept descriptions are in the negation normal form (NNF) [16]. Tableau expansion starts with an ontology, $A := C(x)$, and the tableau expansion rules are applied sequentially until all constraints are satisfied, or an obvious contradiction is detected. This leads to a form of a search tree (Figure 1), with the root node $C(x)$. Here, the edges represent the applied tableau rules. We call our ontology A consistent if and only if one of the complete ABoxes is open, i.e., does not contain an obvious contradiction of the form $\{A(x), \neg A(x)\}$. These rules may be deterministic (such as conjunction) or non-deterministic (such as disjunction). The second type of rule might lead to the wrong path and require backtracking. Thus, with the increase in the size or expressivity of the ontologies, the tree becomes broader and deeper, leading to a huge search space. Hence, the reasoning complexity increases exponentially.

Inference Rule-Based Methods. Inference rule-based reasoners are widely used and well-known for bottom-up reasoning techniques that derive all the inferences by applying inference rules to the set of axioms in the input ontology iteratively (repeatedly). There are thirteen rules (called entailment rules) for RDFS (Table 2) that can be used to entail new facts in an RDFS ontology. Since RDFS has very low expressivity and axioms are in the form of triples, reasoning algorithms are very straightforward. However, for \mathcal{EL}^+ , the axioms are more complex than RDFS. So, usually, some preprocessing has to be done before applying the rules, such as all the axioms have to be in the standard normal form: $C \sqsubseteq D, C_1 \sqcap C_2 \sqsubseteq D, C \sqsubseteq \exists R.D, \exists R. \sqsubseteq D, R_1 \sqcap R, R_1 \circ R_2 \sqcap R$. The standard reasoning task over \mathcal{EL}^+ is called classification. The inference

**Fig. 1.** Tableau Search Tree

rules (called completion rules) (Table 3) can be applied to \mathcal{EL}^+ axioms in a given \mathcal{EL}^+ ontology to derive all the possible consequences. These approaches have also been extended to DLs with nondeterministic language features, e.g., \mathcal{ALCH} and \mathcal{ALCI} (refer to [21] for more details).

An important point to note is that for the conventional logical reasoning algorithms, the entities in an ontology need not necessarily have an actual (real-world) meaning. The reasoner processes the axioms based on the defined rules and has nothing to do with entity names. For example, in inference-based methods, all the inferences are derived by the application of rules. For the axioms, $\text{Man} \sqsubseteq \text{Mortal}$ and $\text{Man}(\text{socrates})$, imply $\text{Mortal}(\text{socrates})$. If we replace Man , socrates , and Mortal with random names A , x , and B , respectively, then, $A \sqsubseteq B$ and $A(x)$ will imply $B(x)$. Even for the case of tableau-based reasoners, a consistent renaming across the whole ontology would not change the outcome of the reasoning task at hand.

Neuro-symbolic Reasoning Methods Henry Kautz, in his AAAI 2020 Robert S. Engelmore Memorial Award Lecture, discussed five categories of neuro-symbolic AI systems as the “Future of AI”⁹. However, the goal of neuro-symbolic approaches, in the context of this survey, is to devise a model based on deep neural networks that can perform logical reasoning by learning the functionality of basic

⁹ https://www.youtube.com/watch?v=_cQITY0SPiw, starting at minute 29

Rule	Premise	Pattern Type	Conclusion
RDFS1	any IRI aaa in D	-	aaa rdf:type rdfs:Datatype.
RDFS2	aaa rdfs:domain xxx. yyy aaa zzz.	TBox ABox	yyy rdf:type xxx.
RDFS3	aaa rdfs:range xxx. yyy aaa zzz.	TBox ABox	zzz rdf:type xxx.
RDFS4a	xxx aaa yyy.	-	xxx rdf:type rdfs:Resource.
RDFS4b	xxx aaa yyy.	-	yyy rdf:type rdfs:Resource.
RDFS5	xxx rdfs:subPropertyOf yyy. yyy rdfs:subPropertyOf zzz.	TBox TBox	xxx rdfs:subPropertyOf zzz.
RDFS6	xxx rdf:type rdf:Property	TBox	xxx rdfs:subPropertyOf xxx.
RDFS7	aaa rdfs:subPropertyOf bbb. xxx aaa yyy.	TBox ABox	xxx bbb yyy.
RDFS8	xxx rdf:type rdfs:Class	TBox	xxx rdfs:subClassOf rdfs:Resource
RDFS9	xxx rdfs:subClassOf yyy. zzz rdf:type xxx.	TBox ABox	zzz rdf:type yyy.
RDFS10	xxx rdf:type rdfs:Class.	TBox	xxx rdfs:subClassOf xxx.
RDFS11	xxx rdfs:subClassOf yyy. yyy rdfs:subClassOf zzz.	TBox TBox	xxx rdfs:subClassOf zzz.
RDFS12	xxx rdf:type rdfs:ContainerMembershipProperty.	TBox	xxx rdfs:subPropertyOf rdfs:member.
RDFS13	xxx rdf:type rdfs:Datatype.	TBox	xxx rdfs:subClassOf rdfs:Literal.

Table 2. RDF/RDFS Entailment Rules

1	A ⊑ C	C ⊑ D	⊨ A ⊑ D
2	A ⊑ C ₁	A ⊑ C ₂	C ₁ ∩ C ₂ ⊑ D ⊨ A ⊑ D
3	A ⊑ C	C ⊑ ∃R.D	⊨ A ⊑ ∃R.D
4	A ⊑ ∃R.B	B ⊑ C	∃R.C ⊑ D ⊨ A ⊑ D
5	A ⊑ ∃S.D	S ⊑ R	⊨ A ⊑ ∃R.D
6	A ⊑ ∃R ₁ .C	C ⊑ ∃R ₂ .D	R ₁ o R ₂ ⊑ R ⊨ A ⊑ ∃R.D

Table 3. \mathcal{EL}^+ Completion Rules

ontology reasoners. This could be done by learning the embeddings of the entities involved in the ontology or by learning the reasoning steps followed by the conventional tableau or inference rule-based systems. In contrast to conventional reasoning approaches, these approaches aren't sound and complete. They work by finding a balance between approximating the precise reasoning abilities of the symbolic systems and utilizing the robust learning capabilities of learning-based techniques.

From the point of view of evaluating a neuro-symbolic reasoning procedure, the key aspects [22], in addition to the previously discussed conventional reasoning evaluation measures, are as follows.

- **Transferability.** Indicates whether the system can transfer the knowledge learned during training to previously unknown and unseen ontology domains. As in the case of conventional logic-based algorithms, the reasoning systems are independent of the meaning of the entities in the ontology. Although the knowledge is not transferable for conventional reasoners, the same set of rules and reasoning algorithms are applicable across domains. The goal is that the neuro-symbolic reasoner should be able to learn the abstraction irrespective of the textual meaning of the entities, and hence, the learned knowledge can be transferred to new ontologies and would not need retraining.
- **Generative Capability.** Indicates whether the system can generate all inferences in one run (in a finite amount of time) or if it is query-based, i.e., predicts the answer to the specific query or checks whether or not a given logical expression is a logical consequence of the provided ontology.
- **Transparency.** The ability of the neuro-symbolic systems to provide a derivation of the inferred triples and provide an explanation for the predictions. Neural networks are generally black boxes and cannot be inspected. So, this aspect indicates whether the proposed method incorporates the transparency offered by the conventional reasoning methods.
- **Accuracy.** The performance of a neuro-symbolic reasoning system, in contrast to conventional methods that are both sound and complete, is normally measured in terms of the number of correct predictions made for each query or correct inferences drawn (such as precision, recall, or f-measure) from an input ontology.

Logical Embeddings. Recent years have witnessed the successful application of low-dimensional vector space representations of knowledge graphs [23] for the knowledge graph completion tasks, such as predicting missing links and new facts [24,25]. The primary goal is to capture and preserve the underlying properties of the given knowledge base in the vector form. However, it is not yet well understood to what extent ontological knowledge, e.g., given as a set of axioms, can be embedded in a principled way. The goal of logical embeddings for ontologies is two-fold.

1. The learned embedding should disregard the textual meanings of all the entities in the ontology and retain the syntax (type and order of constructs), and capture the interconnection and interplay between the axioms.

2. The learning should be transferable either by learning the embedding for each entity and using those for ontology completion or query answering across domains or by learning the functioning of each construct involved in the ontology. The latter case is preferred for emulating conventional logical reasoning.

Intelligent Decision Making. Despite all the research for optimizing the performance of traditional reasoning systems, they still do not scale well. One of the reasons is that a tableau expansion still has a large degree of freedom in deciding which rule on which node should be applied next, as well as in which order it wants to evaluate the nondeterministic alternatives. The former choice does not make a huge difference because if there is a model, it will be found by any order of rule expansions. However, for the latter case, the choice is relevant because out of available choices, one choice may lead to the successful construction of a model, while another might lead to a clash and will have to be dealt with by a backtracking search. In the worst case, such a search must consider all non-deterministic choices and this can have a tremendous impact on the size of the search tree and, therefore, on the reasoning performance. Figure 2 shows different partial expansions for a single set of axioms [16]. Numbers indicate the order in which the tableau is expanded. Out of these three expansions, partial expansion 3 is the ideal choice for a reasoner because it took the least steps to reach an open branch. So, the ideal heuristic is one that will lead to the conclusion in the least number of steps. But, it is more important not to choose a bad heuristic because a bad decision might lead to a very broad and deep search tree, degrading the reasoning performance tremendously.

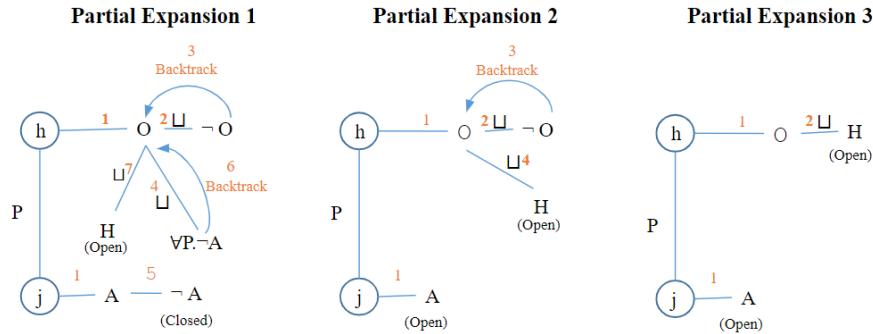


Fig. 2. Partial tableau expansions for the ontology consisting of axioms, $\neg H \sqcap \exists P.H$, $\neg O \sqcap (H \sqcup \forall P . \neg A)$, $O(h)$, $P(h,j)$, $A(j)$

3 Neuro-Symbolic Reasoning: State-of-the-art

In this article, by examining the state-of-the-art, we plan to answer the following research questions.

Q1 *Can neural networks learn to emulate logical reasoning?*

An answer to this question is, in turn, dependent on the following questions.

Q1.1 *What are the different neural network architectures and the corresponding algorithms to learn logical reasoning?*

Q1.2 *How can ontologies and, in general, logical axioms and constructs be represented in a form suitable for different neural network architectures and algorithms?*

Q2 *How capable are the learned models in comparison to conventional RDF and description logic reasoners?*

Q2.1 *Can they handle noisy and inconsistent ontologies?*

An ideal neuro-symbolic reasoner should support the most expressive logic, be transferable to different domains, be able to generate all and only correct inferences in a single run, and explain the generated consequences at a massive scale and with high performance. So the other important question is the following.

Q2.2 *To what extent can neuro-symbolic approaches support these features that were also discussed in Section 2.2?*

In order to evaluate these features, it is important to have benchmarks that can be used to test and compare different neuro-symbolic reasoning systems.

Q3 *What are the metrics that can be used to evaluate the features of the neuro-symbolic reasoning systems?*

3.1 RDF

Since RDF data can be represented as directed, labeled graphs, this information in machine learning scenarios has been utilized in two different forms – general graph kernels [26] and vector space embeddings [27]. With graph-based representations, other than the benefit of improved readability, embeddings can capture the neighborhood information (context) of each entity. So the most intuitive form to use is representing ontology as a graph that can capture the syntax, but this is a complex task. Several graph models (such as bipartite, hypergraph, and metagraph) [28,29,30] have been explored for different purposes ranging from storing and querying RDF graphs to reducing space and time-complexity to solving the reification and provenance problem. [31] Unfortunately, they are not suitable for RDFS reasoning, and these graph forms cannot be provided as input to neural networks because the complex graph forms make it harder for the neural networks to learn from.

Kernel methods refer to machine learning algorithms that learn by comparing pairs of data points using similarity (or dissimilarity) measures and work directly

on graphs without having to do feature extraction to transform them into fixed-length, real-valued feature vectors. The distance between two data instances is computed by counting common substructures in the graphs of the instances, i.e., walks, paths, and trees [32]. However, only a few approaches are general enough to be applied to any given RDF graph, let alone OWL ontologies. In one of the earliest works in this category, Lösch et al. [33] introduced two general RDF graph kernels based on intersection graphs and intersection trees. Later, de Vries et al. [34] simplified the intersection tree path kernel and proposed a better and faster variant for RDF of the general Weisfeiler-Lehman kernel for graphs. In another work, de Vries et al. [35] showed an improvement in the computational time of the kernel when applied to RDF by introducing an approximation of the state-of-the-art Weisfeiler-Lehman graph kernel algorithm.

In the vector space embedding, the initial works adapted neural language models, such as Word2Vec [27], for embedding entities present in the input ontology. Such approaches work on the assumption that closer words in the word sequences are more associated. As shown in Figure 3, the main step is to generate sequences that can be considered similar to sentences in natural language. These sequences could either be obtained directly from the axioms [36,37] or generated from the ontology graphs [38]. Evidently, for the latter case, it is important to have a graphical representation of the ontologies. Once we have the graphs corresponding to the input ontology, several sequences are generated, and these sequences are used to train language models to generate the entity embeddings that capture the likelihood of a sequence of entities. After the training is completed, all the entities are projected into a lower-dimensional feature space, and semantically similar entities are positioned close to each other. These word embeddings can be exploited for tasks related to query answering, ontology completion (such as link prediction and fact prediction), and ontology reasoning (such as predicting concept membership and subsumption).

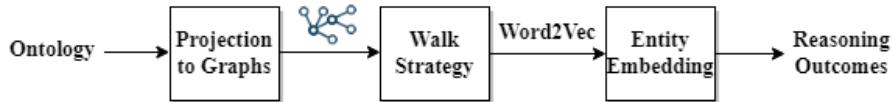


Fig. 3. Ontology to vector representation pipeline

The initial work, RDF2Vec [32] by Ristoski et al., leverages graph walks for transforming the RDF graph to token sequences, and then a natural language embedding algorithm such as Word2Vec [27] is applied to the sequence document that generates embeddings based on token co-occurrences. Word2vec is a computationally-efficient two-layer neural network model for learning word embeddings from raw text. For generating the sequences, two approaches were used; a graph walk based on the breadth-first algorithm and Weisfeiler-Lehman subtree RDF graph kernel, an efficient kernel for graph comparison [39]. The RDF2vec evaluations prove the feasibility and scalability of the graph walk-based methods

over kernel methods. Since the embeddings are created using language models, the language(RDF) semantics are ignored. However, RDF2Vec could be used to study the distributional properties of RDF data, and the pre-trained embeddings can be reused in different deep neural network architectures. Later, intending to capture the most important information for each entity and make the walks less random, Cochez et al. [40] introduced biases to the walks, which led to significant improvements. Along with the edge labels, they augmented each edge with weight using twelve different strategies. Whenever a walk arrives at a vertex, a probability is computed for the possible edges that decide whether to take that edge. Hence, these weights bias the random walks on the graph. Although both the works by Ristoski et al. and Cochez et al. [32,40] were evaluated for several tasks, such as entity relatedness, the approach is suitable for handling link prediction and fact prediction tasks. In OPA2Vec [36], and Onto2Vec [37], a similar approach as RDF2Vec was used. In Onto2Vec, Smaili et al. uses the axioms of an RDFS ontology as the corpus, that is, it treats each axiom as a sentence for training, while in OPA2Vec, Smaili et al. complements the corpus of Onto2Vec with the lexical information provided by ontology metadata, such as `rdfs:comment` and `rdfs:label`. This metadata could potentially provide valuable information about different aspects of the entities. The approaches were evaluated for predicting protein-protein interaction, but the embeddings could be used for graph completion as well. Since these methods treat each axiom as a sentence and do not utilize the interconnections offered by graph structures, it is hard to explore the entity correlations and the logical relation between axioms. Also, considering only the axioms as sentences might lead to a shortage of training corpus for small to medium-scale ontologies. The fundamental method that these walk-based techniques employed were further extended to other types of logic as well, for example, OWL2Vec [41]. The ontology embedding methods based on these walk-based embeddings convert the ontology to a graphical form that can capture the logical structure and inter-entity relationships. Modeling RDF(S) ontologies as graphs is easy. However, OWL ontologies include not only graph structure but also logical constructors, and entities are often augmented with richer lexical information specified using `rdfs:label`, `rdfs:comment`, and many other built-in annotation properties. So modeling OWL axioms as graphs is a complex problem. The general assumption for such techniques is that simpler logic can easily be represented as graphs than more expressive logic, and hence OWL ontologies were projected to RDF graphs [42] to perform walks. To create embeddings, a series of graph walks were conducted, and the sequences were collected in a document. Similar to RDF2Vec, word2vec is used to train the embeddings. But, this approach did not fully capture the logical and lexical information in the graph. Later, Chen et al. introduced OWL2Vec* [38], an extension of OWL2Vec. OWL2Vec* intending to capture the structure, logical constructors involved in each axiom, and the relationship between entities across axioms, generates three documents by walking over the graph. Each document consists of a corpus that captures different aspects of the ontology (i) the graph structure (obtained from the projected RDF graph) and the logic constructors

(obtained from the ontology axioms), (ii) the lexical information (obtained from the annotation axioms, comments, and definitions), and (iii) a combination of corpus from steps (i) and (ii). After the documents are created, they are combined into a single document as one corpus. Finally, OWL2Vec* uses a word embedding model to create embeddings of entities from the generated corpus. The performance was evaluated for concept subsumption and membership tasks.

Later on, the advent of deep learning opened doors for new research directions. Hohenecker and Lukasiewicz [43] proposed Relational Tensor Networks (RTN), an adaptation of Recursive Neural Tensor Networks (RNTN) [44] for relational learning. RNTN was initially designed to support learning from tree-structured data. RTN is an RNTN that makes use of a bilinear tensor layer. The term *relational* in an RTN emphasizes the focus on relational datasets. The intuition is that the most critical information about individuals is hidden in their relations. The authors start by building a Directed Acyclic Graph (DAG) representation of the input RDF ontology where directed edges represent the binary relations, and each of the vertices represents the individuals. Each individual is represented as an incidence vector that indicates the concepts they belong to. The embeddings of the individuals are computed using the RTN model that considers the type or the relation each individual has. They consider two tasks, type prediction (membership of individuals to concepts) and relation prediction. The input for relation prediction is the embeddings of two individuals for which the relations are being classified. The results reported on large datasets for both predictive performance (on concept and relationship prediction) and time consumption (for import and materialization) showed performance improvements compared to RDFox. However, to generate DAG, traditional reasoners (Apache Jena¹⁰, Pellet¹¹) were used. Due to this, RTN is not noise-tolerant. For learning, the datasets (including the inferences) were divided into training, test, and validation sets, meaning the performance is reported on the same ontology, and learning is not transferable to other domains. In order to ensure that there is sufficient data for training, only those relations were considered that appear in at least 5% of the total individuals.

Recently, in an effort to emulate the reasoning capabilities of traditional logical reasoners, deep learning techniques have been utilized to learn the working of entailment rule-based RDF(S) reasoners (see Table 2 for the rules). The first and foremost requirement to work with deep neural networks (DNNs) is that the ontology needs to be converted to a form such that it can focus more on the logical structure and learn the general working of the traditional reasoning algorithms instead of just focusing on the similarity based on the meaning or the textual representation. As shown in Figure 4, given a set of axioms ($A \sqsubseteq B$ and $B \sqsubseteq C$), the model should be able to capture the RDFS rule involved (RDFS11 in this case) in deriving the inference ($A \sqsubseteq C$). In a deep learning context, this setting is similar to sequence-2-sequence machine translation models, also called the encoder-decoder model (similar to the sequential process used in many de-

¹⁰ <https://jena.apache.org>

¹¹ <https://github.com/Complexible/pellet>

ductive reasoning algorithms), that translate text or speech from one language to another. In doing so, it needs to predict the correct substitution of words in one language for words in another. The sentences in the input and target language could be of different lengths and have completely different structures. The difference is that for language translation, the meaning of the words needs to be captured. However, logical reasoning algorithms only learn an abstraction. For example, if we replace A with C_1 , C with C_2 , and D with C_3 , the reasoner only needs to determine that the RDFS11 rule pattern should be used here in order to infer $C_2 \sqsubseteq C_3$. So, the next series of works mostly, though they differ in technical details, apply seq-2-seq based models to capture the functioning of the logical reasoners and generalize them to analogous situations or tasks.

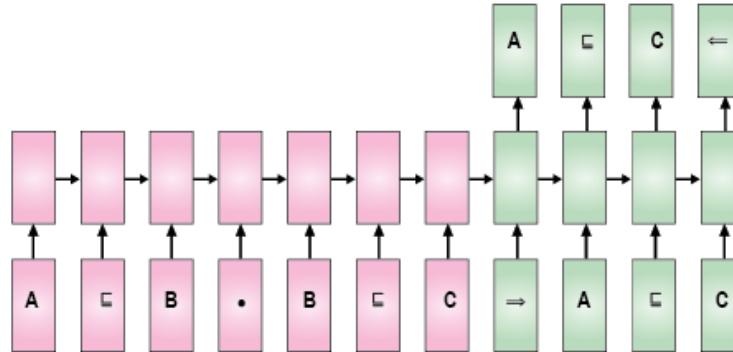


Fig. 4. Seq2seq translator like setting of logical reasoning

With a two-fold goal of learning RDFS rules using deep learning and demonstrating their noise-tolerance capabilities, Makni and Hendler [31] proposed a layered graph model for RDF where the RDF graphs were represented as a sequence of graph words, and the translation-based techniques were used for graph-to-graph learning. The proposed layered graph model is based on simple directed RDF graphs where each layer has its own set of directed edges. The first phase of encoding is to create a 3D adjacency matrix (a tensor) for the input graph. Each layer in the 3D adjacency matrix corresponds to the adjacency matrix between the RDF entities linked by a single relation. These layers are then assigned an ID that represents their layout. The sequence of these IDs represents the RDF graph called “graph words”. The training set becomes a parallel corpus between the sequence of graph words of the input graph and the sequence of graph words of the inference. The algorithm then uses a seq-2-seq encoder-decoder architecture where the input RDF graph and its corresponding inference are encoded. These encoded representations are used for training. After training, the predicted inference graph is obtained for the embedded input test RDF graph. Finally, the inference graph is reconstructed from the translated embedding, and predictions are extracted. However, in this work, evaluation and

training are done on the same RDF graph, that is, there is no learning of the general logical reasoning, and consequently, learning is not transferable to new domains. Hence, re-training will be needed for learning entity embeddings in the new ontology. They evaluated their approach on two different datasets, a synthetic dataset from LUBM and a real-world dataset that has been extracted from DBpedia. This approach can handle only noise in the ABox, and the tolerance to TBox noise is outside the scope of this work. In other words, it is based on the assumption that only assertions can potentially be noisy, and the TBox of an ontology cannot be noisy. Also, for training, they excluded every RDFS rule with only TBox axiom patterns, such as RDFS rule 5. For a similar setting, Makni et al. [45] proposed a method for generating justifications for the derived conclusions. The technique was built upon [31] and consists of a justification model, which is a modified seq2seq model that, while training, takes two sequences (encoded RDF graph and inference graph) as the input and one sequence (justification) as the output. The inferred triples and the justification for each were generated using Apache Jena [46]. Note that the derivations provided by Jena are summarized derivations, so the predicted justifications are also summarized. Although the approach looks promising, they do not consider large and noisy RDFS data. In line with [31], Ebrahimi et al. [47] utilized end-2-end memory networks (MemN2N) [48] to emulate deductive reasoning due to the sequential nature of the memory networks and the attention modeling that captures only relevant information (logical axioms) necessary for the next reasoning step. The proposed approach also addresses the transferability issue by utilizing a preprocessing step consisting of normalization where they consistently rename all the entities to a predefined set of entity names. Note that the URIs which are part of the RDF/RDFS namespace are not renamed, so the learning is based only on the structural information (the type and order of constructs in each axiom) and not the actual names of the entities. The triples (t_i) are treated as text. The model takes a discrete set of normalized triples $t_1, t_2 \dots t_n$ along with a query q and writes them all to the memory, then calculates a continuous embedding for the triples and q through multiple memory accesses. Through multiple hop attention over these continuous representations, the model classifies the query with a 'yes' or 'no', determining whether q can be inferred from the current ontology statements or not. The results demonstrated that a consistent renaming across all the ontologies in the datasets helped perform reasoning over previously unseen RDFS knowledge graphs without retraining. However, there was a limit on the number of sampled triples. Only 1000 triples can be considered at a time. Furthermore, the model is query-based; that is, it does not compute all the inferences that may be derived from a knowledge base but provides predictions for single queries only. They frame the problem as a classification task, i.e., check whether or not the given inference is a valid inference. Later, Ebrahimi et al. explored the capabilities and limitations of neural pointer networks [49] for the same task in a similar setting on different logics (RDF and \mathcal{EL}^+ , discussed later in Section 3.2) and have shown significant improvements in accuracy [22]. The model is an encoder-decoder architecture that uses attention as a pointer

to choose an element of the input knowledge graph at each decoding time step. They claim that since pointer networks can generalize beyond the maximum lengths they were trained on, they are suitable for emulating reasoning behavior and show an improvement in performance by a huge margin. The core of the experiments comprises training a sequence-to-sequence-based model by framing the entailment problem as an input-output mapping task wherein they used two single-layer LSTMs: an LSTM encoder that converts the input sequence to a code that is fed to the generating network and the pointer LSTM that produces a vector that modulates a content-based attention mechanism over symbols in the input. The output of the attention mechanism is a softmax distribution with a dictionary size equal to the length of the input. They used two preprocessing steps: tokenization (splitting the text into meaningful chunks) and normalization (same as [47]). The proposed approach is both transferable and generative, and the results demonstrated improved performance compared to the state-of-the-art. However, they only consider a small dataset for their evaluation, and the scalability aspect hasn't been explored yet.

3.2 Description Logic \mathcal{EL}

The logical constructors supported by the description logic \mathcal{EL} are used by several ontology developers due to their polynomial reasoning time complexity. Intending to further the neuro-symbolic reasoning research for more expressive ontologies, some recent works proposed methods to incorporate the \mathcal{EL} constructors and their extensions, such as existential restriction, conjunction, and the bottom concept (see Table 1) as embeddings.

To incorporate the syntax and the underlying characteristics of the language constructs, Kulmanov et al. [50] used geometric embeddings. Geometric embedding techniques have gained recent attention wherein objective (loss and score) functions for logical axioms are constructed by transforming entities in the ontology into geometric space, as shown in Figure 5. The figure shows the projection of concepts, Parent and Male, of axioms $\text{Parent} \sqsubseteq \text{Human}$, $\text{Male} \sqsubseteq \text{Human}$ and $\text{Parent} \sqcap \text{Male} \equiv \text{Father}$ as spheres [50]. The intersection of the two spheres, Parent and Male is Father.

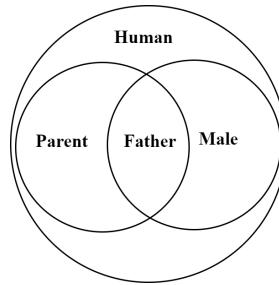


Fig. 5. Entities projected as geometric shapes

The concepts were represented as n-balls (spheres with a fixed radius) and the relations as translation vectors between the centers of each concept ball. TransE [51] assumes that the relationship between words can be computed by their vector difference in the embedding space. Before constructing the embeddings, the individuals were replaced with a single class axiom, and all the axioms were transformed into the standard normal form [16]. For each normal form, a loss function was formulated to capture the semantics of \mathcal{EL}^{++} . Although the results were promising, it did not consider the RBox and certain \mathcal{EL}^{++} axioms that do not translate into the embedding. Further, their use case is predicting protein-protein interactions that are modeled as a traditional link prediction task in knowledge bases. To overcome the limitations, Mondal et al. [52] and Mohapatra et al. [53] extended this work and showed significant improvements. Mondal et al. [52] proposed EmEl++ by complementing the embeddings proposed in [50] by role-inclusions and role chains (additional constructs offered by \mathcal{EL}^{++}). While these methods provided a new technique to model the logical structure of axioms into the geometric space, they only considered one-to-one relations that prevented them from being used in more complex DLs ontologies such as $SROIQ$. To deal with it, Mohapatra et al. [53] further extended the work by Kulmanov et al. by modifying the embeddings to incorporate many-to-many relationships. Although effective, the n-ball embeddings have two major limitations. First, the intersection of two concepts can never be a ball. For example, the intersection of two concepts (balls) **Parent** and **Male**, represents the concept **Father**, which is not a ball. Therefore, the concept equivalence axiom $\text{Parent} \sqcap \text{Male} \equiv \text{Father}$ cannot be captured in the embedding space. Second, simple translation does not allow for scaling the size. So, there are issues with modeling concepts of varying sizes, such as the larger concept **Person** can not be translated into a smaller concept **Parent**. Further, the n-ball approaches regarded the axioms in ABoxes as special cases of TBox axioms and did not treat them separately. This simplification cannot fully express the logical structure of axioms. To overcome these limitations, Xiong et al. proposed BoxEL [54] for embedding \mathcal{EL}^{++} ontologies. They modeled the concepts in the ontology as boxes (axis-aligned hyperrectangles). The intersection of two boxes is also a box. The individuals were encoded as points inside the (concept) boxes they should belong to. The relations were modeled as the affine transformation between boxes and/or points that can capture the cases that are impossible in ball embeddings. Like ELEM, they designed several loss functions for each logical constructor in \mathcal{EL}^{++} and formulated BoxEL as an optimization task. BoxEL preserves the logical structure and provides theoretical soundness guarantees in the sense that if the loss of BoxEL embedding is 0, then the trained model is a (logical) model of the ontology. The evaluations on three ontologies for the subsumption reasoning task and predicting protein-protein interactions prove that BoxEL outperformed previous approaches.

The other series of works by Eberhart et al. [55] and Ebrahimi et al. [22], aims to emulate the behavior of traditional (deductive) reasoners. Similar to RDF entailment rules, \mathcal{EL} involves reasoning rules, i.e., a sequence of applica-

tions of a set of pattern-matching rules that results in the addition of a new set of axioms to the ontology at each step. So the goal of these works is to learn the working of inference-rule-based reasoners, i.e., to learn the structure of reasoning patterns. Although the primary goal by Eberhart et al. [55] is the ontology completion wherein they predict concept inclusions and existential restrictions, this work aligns with the survey because of the shared objective of emulating logical reasoning behavior. In contrast to the other works, their approach also provides transparency of the derived conclusions. The approach works by mapping the inferences (called supports) obtained at each reasoning step in an LSTM learner. For reasoning, they use \mathcal{EL}^+ completion rules. This allows encoding of the input data in terms of ontology axioms. It also provides intermediate answers that might improve results when provided to the system. This logic data is fed into three different LSTM architectures (Deep, Piecewise, and Flat) with identical input and output dimensionalities. The reason for choosing three architectures is to compare the behavior of the systems with and without supports. The number of LSTM cells is determined based on the maximum number of reasoning steps. The approach is able to handle noise and has shown motivating results. Recently, Ebrahimi et al. [22] used pointer networks (explained in Section 3.1 for \mathcal{EL}^+). They claim that since pointer networks can generalize beyond the maximum lengths they were trained on, they are suitable for emulating reasoning behavior and show an improvement in performance by a huge margin. As discussed earlier, the datasets used were very small, and hence the scalability of the system has not been tested.

3.3 Description Logic \mathcal{ALC}

The previously discussed techniques work on RDF data or slightly more expressive ontologies such as \mathcal{EL} , \mathcal{EL}^+ , and \mathcal{EL}^{++} (except [41,38], which in some sense also deal with RDF graphs only wherein the OWL axioms are first projected to RDF graphs, and word2vec is applied over the sequences generated from graph walk). None of the approaches handle constructs such as negation and disjunction. These constructs, along with a few others, make the description logic \mathcal{ALC} more expressive than the ones discussed so far. The neuro-symbolic reasoning work that supports \mathcal{ALC} is limited but diverse in terms of the chosen methods, such as convexity [56], quantum logics [57], and learning heuristics for tableau optimization [58,59].

Özgür et al. [56] proposed to embed concepts as convex regions in vector spaces. The approach works in two parts – first, by constructing concepts on the axes-aligned cones (a class of convex cones) and then placing individuals on these cones. The reason behind choosing axis-aligned cones is two-fold; they are computationally feasible and are preserved under operators (intersection, polarity, and projection) that are sufficient to model constructs in the \mathcal{ALC} description logics. To model unsupported constructs, set-complement, and set-union, they used a scalar product and de Morgan’s laws, respectively. Since an axis-aligned cone is preserved under all these \mathcal{ALC} operators, the idea is that if an ontology can be projected onto an axes-aligned cone, it means that

the ontology is satisfiable. The resulting embeddings can be used in concept membership prediction tasks. An unresolved problem in this work is whether roles can be interpreted by cones (or other feasible structures), and as a side product of defining negations by polarity, they obtain partial models. So there are individuals for which one does not know whether or not they belong to a particular concept. This is different from the approaches considered in classical embedding scenarios. Also, here, they only consider the case where the logic has been specified beforehand, not the case of investigating logic induced by the intersection and polarity operators for the arbitrary cones.

Inspired by quantum space, Garg et al. [57] proposed Embed2Reason (E2R) that embeds an \mathcal{ALC} ontology into a finite-dimensional vector space preserving the logical structure. Such an embedding, referred to as quantum embedding, satisfies the axioms of quantum logic [60] and allows one to perform logical operations (e.g., membership, negation, conjunction, disjunction, inclusion, and implication) directly over the vectors in a manner similar to the boolean logic except that distributive law does not hold true. E2R formulated an unconstrained optimization program that captures all such quantum logical (as well as regularity) constraints, and the program was solved using the Stochastic Gradient Descent (SGD) technique. The resulting embedding captured the logical input structure with good accuracy on complex deductive and predictive reasoning tasks compared to other embedding techniques.

As discussed in Section 2.2, the performance of the tableau-based reasoners depends a lot on the non-deterministic choices made while constructing the tableau. A series of works by Mehri et al. [58,61,62] focuses on selecting the right expansion heuristics for tableau expansion using machine learning techniques. The goal is not to emulate the tableau expansions but to learn the right expansion heuristics for intelligent decision-making at each non-deterministic rule application. In one of these works [58], Mehri et al. focused on improving semantic branching for disjunctions and its effect on backjumping, an optimization technique for backtracking. Even though semantic branching avoids redundant search space by a great deal, machine learning techniques help to decrease redundant search space exploration further by learning new orders for applying rules at each branching level. Their results show that machine learning speeds up JFact by one to two orders of magnitude. Later in [61], they improved the ToDo list optimization technique in JFact using machine learning and applying it to a complex description logic that is a syntactic variant of OWL and contains propositional logic as a very small subset. To the best of our knowledge, this is the first machine-learning approach for improving such a rule-based optimization technique for OWL reasoners. The approach is based on an independent systematic analysis of rule orderings to uncover ontology patterns and their associated features relevant to this specific optimization technique. In continuation with this work, Mehri et al. proposed a variation [62] wherein they learn to choose among the built-in expansion-ordering heuristics. These learning-derived models are built based on features computed and extracted from ontologies. Therefore, the manual fine-tuning of heuristics for these reasoners can be replaced by a new

ML-based approach implementing automatic fine-tuning of heuristics to make the right choice. However, the features were manually computed, and supervised learning techniques were applied to a set of ontologies to learn the best possible expansion. Also, despite a significant amount of research there is no general single sorting strategy that works best for all ontology types. That's because the ontologies vary in features, and hence the interaction and interplay of the axioms also vary. Manually designing heuristics for large and complex ontologies is nearly impossible. Therefore, it is important to develop a methodology that automatically learns to choose the most suitable expansion heuristic for each ontology. Inspired by the success of AlphaGo [63], researchers started exploring reinforcement learning (RL) for intelligent human decision-making for their applications. Based on a similar idea, Singh et al. [59] proposed to learn the non-deterministic expansions from scratch using deep RL such that the agent can learn the interaction and interplay between the axioms and make clever decisions whenever it has to deal with non-deterministic choices.

3.4 OWL 2 RL

So far, there have not been many neuro-symbolic reasoning techniques for expressive ontology profiles, such as OWL 2 RL and OWL 2 DL. Hohenecker and Lukasiewicz [64] developed a deep learning-based model called Recursive Reasoning Networks (RRN) for reasoning on OWL 2 RL ontologies. The approach deals with ABox and TBox separately (called factual information and ontology, respectively, in the paper). The primary goal is to train an RRN to learn the rules specified as part of TBox to answer queries related to the facts in the ABox. Since the training is relative to a particular TBox, it is independent of the ABox axioms it is provided with and can be used to answer queries over new ABox based on the same vocabulary. However, to reason over ontologies from new domains, the RRN needs to be retrained again. Hence the learning is not transferable to new domains. Since they train it on a fixed TBox, the concepts and relations are also fixed from the beginning, and that determines the number of recursive layers needed for training. For generating the embeddings, all the facts are first viewed as triples, where concept assertion, $C(i)$, is denoted as $(i, \text{memberOf}, C)$, and all other relation assertions, $R(i, j)$, as (i, R, j) . For negative samples, such as $\neg C(i)$ and $\neg R(i, j)$, facts are represented as $(i, \neg \text{memberOf}, C)$ and $(i, \neg R, j)$, respectively. Then, a random embedding is generated for each individual present in these triples. After this, the model updates the embeddings of the individuals by iterating over the triples. As mentioned above, the number of iterations required depends on the concepts and relations in the respective datasets. After these updates, each embedding is supposed to store the considered triple and encode all the inferences that are based on the triple involved. So, intuitively, a single update step conducts local reasoning based on the embeddings and the new information gained through the provided fact. Since the model goes through all the facts multiple times in several iterations, it allows for multi-hop reasoning. After training, predictions are made based on the obtained embeddings of all individuals. The reported results show good accuracy.

However, since the updates are made in multiple steps, the method does not have the provision of generating justifications or transparency for the predicted inferences. In RRNs, triples are not treated as text, instead, the individuals that appear in a triple are mapped to their embeddings before providing the same to any layers of the used model. However, this means that RRNs are independent of the individual names used in a database, which makes perfect sense, as it is only the structure of a knowledge base that determines possible inferences. A point of difference from the other approaches [31,47] is that the RRN is trained on a fixed ontology, i.e., the vocabulary of concepts and relations is predetermined. The model does not learn the logical structure of the axioms; hence, the learning is not transferable to different domains. The performance is evaluated based on the query predictions. Hence the approach is not generative.

3.5 Summary

This section gives an overview of the neuro-symbolic reasoning techniques discussed so far from two different perspectives. In Table 4, we categorize the existing works based on the technique. Although the technical details differ, the works can be grouped based on the high-level idea used. For each work, we give a one-line summary along with the reasoning task handled. In Table 5, inspired by Ebrahimi et al. [22], we highlight the key differences between the learned models in terms of the essential features that can potentially provide pointers for further research in this direction. The *transfer* column indicates whether the system can adapt to ontologies in the new domain by making use of its learned model from other domains. The *transparency* column indicates whether the predictions and the drawn inferences are explainable. The *scale* column indicates the size of input ontologies that were used for evaluations, ranging from a few logical statements (up to 1,000 triples) [22,47] to very large (say, up to 1 million). The *performance* column indicates the accuracy of the system on the reasoning task; “high” indicates 70% or more, while “low” indicates values slightly better than random guessing. Although these measures are subjective and there is no common platform where all the systems can be run together and compared on similar datasets, the goal here is just to summarize the existing neuro-symbolic reasoning techniques based on the results reported in terms of their important features.

Technique	Paper	Logic	Summary
Walk-based Embedding	[32]	RDF	Ristoski et al. proposed RDF2Vec. Word2Vec is applied on the sequences generated using graph walks and the sub-tree RDF adaptation of the Weisfeiler-Lehman algorithm. Task: Graph Completion (Link or Fact Prediction)

	[40]	RDF	Cochez et al. extended RDF2Vec [32] by assigning weights to the edges that bias the random graph walks and captures only the most important information. Task: Graph Completion (Link or Fact Prediction)
	[37]	RDFS	Smaili et al. proposed Onto2Vec. Each axiom is treated as a sentence and Word2Vec is applied for training. Task: Graph Completion (Link or Fact Prediction)
	[36]	RDFS	Smaili et al. extended Onto2Vec [37] and proposed OPA2Vec by complementing the corpus with the lexical information provided by ontology metadata, e.g., rdfs:comment and rdfs:label. Task: Graph Completion (Link or Fact Prediction)
	[41]	$\mathcal{SROTQ}(\mathcal{D})$	Holter et al. proposed OWL2Vec. OWL ontologies are mapped to RDF graphs and Word2Vec is applied over generated walks. Task: Concept membership and concept subsumption
	[38]	$\mathcal{SROTQ}(\mathcal{D})$	Chen et al. extended OWL2Vec [41] and proposed OWL2Vec*. The training corpus consists of three documents (structural, lexical, and combination of both) that captures the interrelation between the entities better than previous Word2Vec based methods. Task: Concept membership and concept subsumption
Geometric Embedding	[50]	\mathcal{EL}^{++}	Kulmanov et al. proposed ELEM where the concepts were represented as n-balls and the relations as translation vectors between the centers of each concept ball. Task: Subsumption
	[52]	\mathcal{EL}^{++}	Mondal et al. proposed EmEL ⁺⁺ by extending ELEM [50] with relation inclusion and role chains. Task: Subsumption

	[53]	\mathcal{EL}^{++}	Mohapatra et al. dealt with the limitations of ELEM [50] and EmEL ⁺⁺ [52] by incorporating many-to-many relations in the embeddings. Task: Subsumption
	[54]	\mathcal{EL}^{++}	Xiong et al. mapped concepts as boxes and deals with the limitations of n-ball based embeddings. Task: Subsumption
	[56]	\mathcal{ALC}	Özçep et al. embed concepts as convex regions in vector spaces. Task: Concept Membership
	[57]	\mathcal{ALC}	Garg et al. proposed embeddings in the quantum space. Task: Concept Membership
Emulating Logical Reasoning	[31]	RDFS	Makni and Hendler used an encoder-decoder architecture for translating the embedding of the input RDF graph to the embedding of the corresponding inference graph. Task: Entailment Reasoning
	[47]	RDFS	Ebrahimi et al. explored the capabilities of end-2-end memory networks (MemN2N). Use of normalized embeddings support transfer. Task: Query-based classification
	[22]	RDFS and \mathcal{EL}^+	Ebrahimi et al. utilized pointer networks for learning the sequential application of inference rules used in many deductive reasoning algorithms. Task: Entailment Reasoning
	[64]	OWL 2 RL	Hohenecker and Lukasiewicz developed a deep learning-based model called Recursive Reasoning Networks (RNN). Task: Entailment Reasoning
	[55]	\mathcal{EL}^+	Eberhart et al. utilized LSTMs to learn the mapping of the inferences obtained at each reasoning step with the axioms in the ontology. Task: Ontology completion (predict concept inclusions and existential restrictions)

	[45]	RDFS	Makni et al. built upon the work by Makni and Hendler [31] for generating explanations for the derived conclusions by taking the RDF graph and inferred triples as input and the explanations as the target. Task: Entailment Reasoning with summarized explanations
	[43]	RDF	Hohenecker and Lukasiewicz proposed Relational Tensor Network (RTN). Embeddings of the individuals are computed by applying RTNs on the Directed Acyclic Graph representation of the ontology (including the inferences). Task: Concept Membership and Relation prediction
Learning Heuristics for Tableau	[58]	\mathcal{ALC}	Mehri et al. used machine learning to select the right heuristics for tableau expansion. The learning-derived model is built based on manually designed features that are computed and extracted from ontologies. Task: Satisfiability
	[61] [62]	\mathcal{ALC}	Inspired by AlphaGo [63], Singh et al. propose to use RL for learning the heuristics for non-deterministic tableau expansions from scratch. Task: Consistency Checking

Table 4: Summary of the state-of-the-art neuro-symbolic reasoning techniques over RDF and Description Logic ontologies

4 Other Related Efforts

Although there are several important developments in the neuro-symbolic reasoning space (discussed in the earlier sections) and advancements continue to happen, there still is a lack of availability of

1. **Test Cases.** Ontologies that vary in terms of several parameters, such as ontology size and axiom types.
2. **Common Infrastructure and Experiment Design.** A platform where all the existing systems can be run together and compared against some performance metrics.

Paper	Logic	Transfer	Transparency	Scalability	Performance
[32]	RDF	no	no	low	moderate
[40]	RDF	no	no	low	moderate
[37]	RDFS	no	no	low	moderate
[36]	RDFS	no	no	low	moderate
[41]	$\mathcal{SROIQ}(\mathcal{D})$	no	no	low	moderate
[38]	$\mathcal{SROIQ}(\mathcal{D})$	no	no	low	moderate
[50]	\mathcal{EL}^{++}	no	no	low	moderate
[52]	\mathcal{EL}^{++}	no	no	low	moderate
[53]	\mathcal{EL}^{++}	no	no	low	moderate
[54]	\mathcal{EL}^{++}	no	no	low	high
[56]	\mathcal{ALC}	no	no	low	moderate
[57]	\mathcal{ALC}	no	no	low	moderate
[31]	RDFS	no	no	low	high
[47]	RDFS	yes	no	moderate	high
[22]	RDFS and \mathcal{EL}^+	yes	no	no	high
[64]	OWL 2 RL	no	no	low	high
[55]	\mathcal{EL}^+	yes	yes	moderate	low
[45]	RDFS	no	yes	low	high
[43]	RDF	no	no	high	high
[58]					
[61]	\mathcal{ALC}	yes	yes	moderate	high
[62]					
[59]	\mathcal{ALC}	yes	yes	yes	na

Table 5. Overview of the neuro-symbolic reasoning techniques from the point of view of key evaluation criteria: Transfer (yes or no), Transparency (yes or no), Scalability (low, moderate and high), and Performance (low, moderate and high). na implies not applicable.

One way to bridge this gap and foster a strong research community is to hold challenges. This could further help the new ontology and system developers to check the hardness of the ontologies and find the performance bottlenecks of their systems. With this aim and inspired by ORE [65], two editions of the Semantic Reasoning Evaluation Challenge (SemREC)¹² were held so far. In the first edition of SemREC¹³ [66], co-located with the 20th International Semantic Web Conference (ISWC 2021¹⁴), submissions were invited across three tracks. In the first track, ontologies that challenge the reasoners can be submitted. The second and third track is for RDFS and description logic reasoners that make use of traditional and neuro-symbolic techniques, respectively. The second edition of SemREC was co-located with the 21st International Semantic Web Conference (ISWC 2022¹⁵). Since the research in conventional reasoning optimization has

¹² <https://semrec.github.io/>

¹³ <https://semrec.github.io/SemREC2021.html>

¹⁴ <https://iswc2021.semanticweb.org/>

¹⁵ <https://iswc2022.semanticweb.org/>

stagnated, in the second edition of SemREC, the second track from the first edition was dropped. The track on neuro-symbolic reasoning systems has been expanded to include multi-hop reasoners and inductive reasoning techniques. Across the two editions, there were three submissions related to neuro-symbolic reasoning [67,68,69]. The system submitted by Mohapatra et al. [67] is the same as [53] that was discussed in section 3. They provided an effective way to capture the many-to-many relations between the concepts and built on top of [50,52]. Chowdhury et al. [68] used an end-to-end Memory Network (MemN2N) [48] with attention that captures the most relevant information to conduct logical reasoning. Mehryar et al. [69] submitted aTransE, an extension of TransE and rTransE to make subsumption and instance checking reasoning possible, whereby reasoning can take place in the vector space by leveraging transitive relations. They further improved the quality of embeddings using reinforcement learning, where they generated multi-hop samples using a policy network. The agent is a neural network that takes as input an entity vector embedding (i.e., state) and outputs a relationship (i.e., action). The use of RL helped find more meaningful and longer sequences of translations. All the submissions were evaluated on ontologies from different sources such as the ORE benchmark framework [65], OWL2Bench [8], and CaLiGraph [70]. The ORE benchmark framework¹⁶ is an open-source java-based framework that was a part of the OWL Reasoner Evaluation (ORE) Competition. The competition evaluated the performance of OWL 2 compliant reasoners over several different OWL 2 EL and OWL 2 DL ontologies. But, the performance evaluation in the context of varying sizes of an ontology was not considered. The ORE competition corpus can be used with the framework for reasoner evaluation. The framework evaluates the reasoners on three reasoning tasks – consistency checking, classification, and realization. The framework does not include the evaluation of the SPARQL query engines (with OWL reasoning support) with respect to the coverage of OWL 2 constructs and scalability. OWL2Bench, on the other hand, tests the OWL 2 reasoners in terms of their coverage, scalability, and query performance. It is an extension of the well-known University Ontology Benchmark (UOBM) [71]. OWL2Bench consists of the following – TBox axioms for each of the four OWL 2 profiles (EL, QL, RL, and DL), a synthetic ABox axiom generator that can generate axioms of arbitrary size, and a set of SPARQL queries that involve reasoning over the OWL 2 language constructs. CaLiGraph ontologies were the winners of task 1 (challenging ontology submission) of SemREC 2021. CaLiGraph is a large-scale cross-domain knowledge graph generated from Wikipedia by exploiting the category system, list pages, and other list structures in Wikipedia, containing more than 15 million typed entities and around 10 million relation assertions. Other than the knowledge graphs such as DBpedia [72] and YAGO [73], whose ontologies are comparably simplistic, CaLiGraph has a rich ontology comprising more than 200,000 class restrictions. Those two properties, a large ABox and a rich ontology, make it an interesting challenge for benchmarking reasoners. Since all the submissions to the neuro-symbolic reasoner track in the second edition of

¹⁶ <https://github.com/ykazakov/ore-2015-competition-framework>

SemREC belonged to a different category with respect to the task on which the system was evaluated on, it was not possible to compare them. For instance, Sulogna et al. [68] submitted a system that supports the *transfer* property. The evaluation was performed on a query-based task, where they obtained good results on the CaliGraph dataset without retraining. But their system supports only RDFS axioms, and there was a limit on the number of axioms (1000 triples) used for training. For that, they randomly divided each ontology into multiple datasets of 1000 triples each. The other submission by Shervin et al. [69] was designed for the link prediction task (subclass and type of relations), and the metric used was Hits@n.

The growth of neuro-symbolic and conventional ontology reasoners largely depends on the datasets that are available for evaluating them. Most reasoner evaluations are performed on a subset of several thousands of ontologies that are available across repositories such as the DBpedia [72], YAGO [73], Wikidata [74], Claros¹⁷, NCBO Bioportal¹⁸, and AgroPortal¹⁹. However, such an approach is inflexible as most ontologies involve only a limited set of OWL constructs and arbitrarily large and complex ontologies are seldom available that can be used to test the limits of systems being benchmarked. Similarly, existing synthetic benchmarks, such as LUBM [75], UOBM [71], OntoBench [76], ORE, and OWL2Bench, also do not have the flexibility to generate ontologies that vary in terms of *TBox* size and language constructs. To evaluate the reasoners, it is important to have a benchmark that can generate ontologies depending on the task at hand. For example, if the performance of a neuro-symbolic reasoner needs to be evaluated or compared with another reasoner on the concept subsumption task for \mathcal{EL}^{++} ontologies, we need to have a benchmark that can generate a large number of ontologies that vary in size with respect to different OWL constructs. [77], an extension of OWL2Bench describes an ongoing effort towards building such a customizable ontology benchmark for OWL 2 reasoners.

5 Conclusion

In this chapter, we gave an overview of the neuro-symbolic reasoning techniques over RDF and description logic ontologies. We now answer the research questions posed in Section 3. Although the body of published work (Table 4 and Table 5) is relatively small, it is obvious from the results that deductive reasoning is a very hard task for deep learning with increasing hardness, especially for more complex logic and scalability issues. It can be observed that, under certain provisions, even the best reasoning models based on machine learning are still not in a position to compete with their symbolic counterparts. To an extent, it has been shown that the developed neuro-symbolic techniques are noise-resistant, but they need negative samples to be added to the datasets, which is mostly done manually

¹⁷ <https://www.clarosnet.org>

¹⁸ <https://bioportal.bioontology.org/>

¹⁹ <http://agroportal.lirmm.fr/>

or randomly. Despite all the efforts, existing neuro-symbolic reasoners are still unable to do the following.

- Perform full reasoning over expressive ontologies such as OWL 2 DL. The existing methods have limitations in learning complex ontology axioms. Most techniques used simpler and less expressive ontology profiles. However, interestingly, the body of work indicates the potential and feasibility of deep learning techniques for reasoning over more expressive ontologies.
- Perform reasoning over new ontologies (domain-independent reasoning) with high precision and recall. Although some techniques support the *transfer* property, the method used was based on the consistent renaming of all the entities in the ontologies (each entity is assigned a unique numeric identifier). But this does not solve the out-of-vocabulary issue because the highest number assigned to an entity depends on the size of the largest ontology used for training. The size of the test ontology could never be larger than that. However, the initial results are interesting and highly motivating for further work in that direction.

A major aspect that the researchers in the domain have probably overlooked is the unavailability of the datasets and a common evaluation framework. Clearly, all the techniques discussed indicate the scope for research and improvements in this direction. It would have been great if all the published or yet-to-be-published works could have been trained and tested on the same datasets using the same resources. From the results reported in the summary table, if the scalability and performance are high, without any transfer, then the approach is clearly unsuitable for the deductive reasoning goal. On the other hand, some techniques support the transfer property but fail to consider the dataset variety (varying language constructs and dataset sizes). In order to make progress, firstly, it is very important to incorporate the datasets in evaluations that vary in terms of several parameters, and these datasets should become the standard for the neuro-symbolic reasoning systems. Secondly, automatic learning should be incorporated. Recently, reinforcement learning for FOL [78] has shown tremendous improvements where the systems learn to perform reasoning from scratch. For instance, in the case of reasoners that use inference-rule based methods, instead of providing the inferences (output) beforehand and then mapping the input and the output sequences using seq-2-seq translators, an RL agent can be provided with all the ontology axioms, along with the inference rules. The RL agent can then start randomly applying the inference rules on the given set of axioms. The rewards can be designed so that the agent learns to reach the desired conclusion in the least number of steps. Learning from scratch has an added advantage over labeled/supervised methods, where neural networks learn by exploring several possible moves that could lead to either right or wrong decisions (human-like learning). Something similar can also be explored for the tableau based reasoners. Extending the architectures discussed here with learning from scratch would enable transparency and explanation generation alongside the computed predictions.

References

1. Klyne G, J Carroll J, McBride B. Resource Description Framework (RDF) ; 2015. Available from: <https://www.w3.org/TR/rdf11-concepts/>.
2. Motik B, Grau BC, Horrocks I, et al. OWL 2 Web Ontology Language Profiles (Second Edition) ; 2012. Available from: <https://www.w3.org/TR/owl2-profiles/>.
3. Horrocks I. Ontologies and the Semantic Web. Commun ACM. 2008;51(12):58–67. Available from: <https://doi.org/10.1145/1409360.1409377>.
4. Heist N, Hertling S, Ringler D, et al. Knowledge Graphs on the Web - An Overview. In: Tiddi I, Lécué F, Hitzler P, editors. Knowledge graphs for explainable artificial intelligence: Foundations, applications and challenges. (Studies on the Semantic Web; Vol. 47). IOS Press; 2020. p. 3–22. Available from: <https://doi.org/10.3233/SSW200009>.
5. Krötzsch M, Simancik F, Horrocks I. A Description Logic Primer. CoRR. 2012; abs/1201.4089. Available from: <http://arxiv.org/abs/1201.4089>.
6. Horrocks I, Kutz O, Sattler U. The Even More Irresistible SROIQ. In: Doherty P, Mylopoulos J, Welty CA, editors. Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006. AAAI Press; 2006. p. 57–67. Available from: <http://www.aaai.org/Library/KR/2006/kr06-009.php>.
7. Baader F, Calvanese D, McGuinness DL, et al., editors. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press; 2003.
8. Singh G, Bhatia S, Mutharaju R. OWL2Bench: A Benchmark for OWL 2 Reasoners. In: Pan JZ, Tamam VAM, d'Amato C, et al., editors. The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II; (Lecture Notes in Computer Science; Vol. 12507). Springer; 2020. p. 81–96. Available from: https://doi.org/10.1007/978-3-030-62466-8_6.
9. Sarker MK, Zhou L, Eberhart A, et al. Neuro-Symbolic Artificial Intelligence: Current Trends. CoRR. 2021;abs/2105.05330. Available from: <https://arxiv.org/abs/2105.05330>.
10. d'Avila Garcez AS, Besold TR, Raedt LD, et al. Neural-Symbolic Learning and Reasoning: Contributions and Challenges. In: 2015 AAAI Spring Symposia, Stanford University, Palo Alto, California, USA, March 22-25, 2015. AAAI Press; 2015. Available from: <http://www.aaai.org/ocs/index.php/SSS/SSS15/paper/view/10281>.
11. d'Amato C, Fanizzi N, Esposito F. Query Answering and Ontology Population: An Inductive Approach. In: Bechhofer S, Hauswirth M, Hoffmann J, et al., editors. The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings; (Lecture Notes in Computer Science; Vol. 5021). Springer; 2008. p. 288–302. Available from: https://doi.org/10.1007/978-3-540-68234-9_23.
12. Baader F, Ganter B, Sertkaya B, et al. Completing Description Logic Knowledge Bases Using Formal Concept Analysis. In: Veloso MM, editor. IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007; 2007. p. 230–235. Available from: <http://ijcai.org/Proceedings/07/Papers/035.pdf>.
13. Lehmann J, Hitzler P. Concept Learning in Description Logics using Refinement Operators. Mach Learn. 2010;78(1-2):203–250. Available from: <https://doi.org/10.1007/s10994-009-5146-2>.

14. d'Amato C, Fanizzi N, Esposito F. Distance-Based Classification in OWL Ontologies. In: Lovrek I, Howlett RJ, Jain LC, editors. Knowledge-Based Intelligent Information and Engineering Systems, 12th International Conference, KES 2008, Zagreb, Croatia, September 3-5, 2008, Proceedings, Part II; (Lecture Notes in Computer Science; Vol. 5178). Springer; 2008. p. 656–661. Available from: https://doi.org/10.1007/978-3-540-85565-1_81.
15. Rizzo G, Fanizzi N, d'Amato C, et al. Prediction of class and property assertions on OWL ontologies through evidence combination. In: Akerkar R, editor. Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011, Sogndal, Norway, May 25 - 27, 2011. ACM; 2011. p. 45. Available from: <https://doi.org/10.1145/1988688.1988741>.
16. Hitzler P, Krötzsch M, Rudolph S. Foundations of Semantic Web Technologies. Chapman and Hall/CRC Press; 2010. Available from: <http://www.semantic-web-book.org/>.
17. Matthews B. Semantic Web Technologies. E-learning. 2005 01;6:19. Available from: https://www.researchgate.net/publication/30408878_Semantic_Web_Technologies.
18. Guarino N, Oberle D, Staab S. What Is an Ontology? In: Staab S, Studer R, editors. Handbook on ontologies. Springer; 2009. International Handbooks on Information Systems; p. 1–17. Available from: https://doi.org/10.1007/978-3-540-92673-3_0.
19. Grau BC, Horrocks I, Motik B, et al. OWL 2: The next step for OWL. J Web Semant. 2008;6(4):309–322. Available from: <https://doi.org/10.1016/j.websem.2008.05.001>.
20. Ewald W. The Emergence of First-Order Logic. In: Zalta EN, editor. The Stanford encyclopedia of philosophy. Spring 2019 ed. Metaphysics Research Lab, Stanford University; 2019.
21. Simancik F, Motik B, Horrocks I. Consequence-Based and Fixed-Parameter Tractable Reasoning in Description Logics. Artif Intell. 2014;209:29–77. Available from: <https://doi.org/10.1016/j.artint.2014.01.002>.
22. Ebrahimi M, Eberhart A, Hitzler P. On the Capabilities of Pointer Networks for Deep Deductive Reasoning. CoRR. 2021;abs/2106.09225. Available from: <https://arxiv.org/abs/2106.09225>.
23. Hogan A, Blomqvist E, Cochez M, et al. Knowledge Graphs. ACM Comput Surv. 2022;54(4):71:1–71:37. Available from: <https://doi.org/10.1145/3447772>.
24. Ji S, Pan S, Cambria E, et al. A Survey on Knowledge Graphs: Representation, acquisition and applications. CoRR. 2020;abs/2002.00388. Available from: <https://arxiv.org/abs/2002.00388>.
25. Zhang W, Chen J, Li J, et al. Knowledge Graph Reasoning with Logics and Embeddings: Survey and Perspective. CoRR. 2022;abs/2202.07412. Available from: <https://arxiv.org/abs/2202.07412>.
26. Gärtner T, Flach PA, Wrobel S. On Graph Kernels: Hardness Results and Efficient Alternatives. In: Schölkopf B, Warmuth MK, editors. Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings; (Lecture Notes in Computer Science; Vol. 2777). Springer; 2003. p. 129–143. Available from: https://doi.org/10.1007/978-3-540-45167-9_11.
27. Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space. In: Bengio Y, LeCun Y, editors. 1st International Conference on

- Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings; 2013. Available from: <http://arxiv.org/abs/1301.3781>.
28. Hayes J, Gutierrez C. Bipartite Graphs as Intermediate Model for RDF. In: McIlraith SA, Plexousakis D, van Harmelen F, editors. The Semantic Web - ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings; (Lecture Notes in Computer Science; Vol. 3298). Springer; 2004. p. 47–61. Available from: https://doi.org/10.1007/978-3-540-30475-3_5.
 29. Morales AAM. A Directed Hypergraph Model for RDF. In: Simperl EPPB, Diederich J, Schreiber G, editors. Proceedings of the KWEPSY 2007 Knowledge Web PhD Symposium 2007, Innsbruck, Austria, June 6, 2007; (CEUR Workshop Proceedings; Vol. 275). CEUR-WS.org; 2007. Available from: <http://ceur-ws.org/Vol-275/paper24.pdf>.
 30. Chernenkiy V, Gapanyuk Y, Nardid A, et al. Using the metagraph approach for addressing RDF knowledge representation limitations. 09; 2017. p. 47–52.
 31. Makni B, Handler JA. Deep learning for noise-tolerant RDFS reasoning. Semantic Web. 2019;10(5):823–862. Available from: <https://doi.org/10.3233/SW-190363>.
 32. Ristoski P, Rosati J, Noia TD, et al. RDF2Vec: RDF Graph Embeddings and Their Applications. Semantic Web. 2019;10(4):721–752. Available from: <https://doi.org/10.3233/SW-180317>.
 33. Lösch U, Bloehdorn S, Rettinger A. Graph Kernels for RDF Data. In: Simperl E, Cimiano P, Polleres A, et al., editors. The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings; (Lecture Notes in Computer Science; Vol. 7295). Springer; 2012. p. 134–148. Available from: https://doi.org/10.1007/978-3-642-30284-8_16.
 34. de Vries GKD. A Fast Approximation of the Weisfeiler-Lehman Graph Kernel for RDF Data. In: Blockeel H, Kersting K, Nijssen S, et al., editors. Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I; (Lecture Notes in Computer Science; Vol. 8188). Springer; 2013. p. 606–621. Available from: https://doi.org/10.1007/978-3-642-40988-2_39.
 35. de Vries GKD, de Rooij S. A Fast and Simple Graph Kernel for RDF. In: d'Amato C, Berka P, Svátek V, et al., editors. Proceedings of the International Workshop on Data Mining on Linked Data, with Linked Data Mining Challenge collocated with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2013), Prague, Czech Republic, September 23, 2013; (CEUR Workshop Proceedings; Vol. 1082). CEUR-WS.org; 2013. Available from: <http://ceur-ws.org/Vol-1082/paper2.pdf>.
 36. Smaili FZ, Gao X, Hoehndorf R. OPA2Vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction. Bioinform. 2019; 35(12):2133–2140. Available from: <https://doi.org/10.1093/bioinformatics/bty933>.
 37. Smaili FZ, Gao X, Hoehndorf R. Onto2Vec: joint vector-based representation of biological entities and their ontology-based annotations. Bioinform. 2018;34(13):i52–i60. Available from: <https://doi.org/10.1093/bioinformatics/bty259>.
 38. Chen J, Hu P, Jiménez-Ruiz E, et al. OWL2Vec*: Embedding of OWL ontologies. Machine Learning. 2021;110(7):1813–1845. Available from: <https://doi.org/10.1007/s10994-021-05997-6>.

39. Shervashidze N, Schweitzer P, van Leeuwen EJ, et al. Weisfeiler-Lehman Graph Kernels. *J Mach Learn Res.* 2011;12:2539–2561. Available from: <https://dl.acm.org/doi/10.5555/1953048.2078187>.
40. Cochez M, Ristoski P, Ponzetto SP, et al. Biased Graph Walks for RDF Graph Embeddings. In: Akerkar R, Cuzzocrea A, Cao J, et al., editors. Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017. ACM; 2017. p. 21:1–21:12. Available from: <https://doi.org/10.1145/3102254.3102279>.
41. Holter OM, Myklebust EB, Chen J, et al. Embedding OWL ontologies with OWL2Vec. In: Suárez-Figueroa MC, Cheng G, Gentile AL, et al., editors. Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019; (CEUR Workshop Proceedings; Vol. 2456). CEUR-WS.org; 2019. p. 33–36. Available from: <http://ceur-ws.org/Vol-2456/paper9.pdf>.
42. Soylu A, Kharlamov E, Zheleznyakov D, et al. OptiqueVQS: A Visual Query System over Ontologies for Industry. *Semantic Web.* 2018;9(5):627–660. Available from: <https://doi.org/10.3233/SW-180293>.
43. Hohenecker P, Lukasiewicz T. Deep Learning for Ontology Reasoning. *CoRR.* 2017; abs/1705.10342. Available from: <http://arxiv.org/abs/1705.10342>.
44. Socher R, Perelygin A, Wu J, et al. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL. ACL; 2013. p. 1631–1642. Available from: <https://aclanthology.org/D13-1170/>.
45. Makni B, Abdelaziz I, Hendler JA. Explainable deep RDFS reasoner. *CoRR.* 2020; Available from: <https://arxiv.org/abs/2002.03514>.
46. Carroll JJ, Dickinson I, Dollin C, et al. Jena: Implementing the Semantic Web Recommendations. In: Feldman SI, Uretsky M, Najork M, et al., editors. Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 17-20, 2004. ACM; 2004. p. 74–83. Available from: <https://doi.org/10.1145/1013367.1013381>.
47. Ebrahimi M, Sarker MK, Bianchi F, et al. Reasoning over RDF Knowledge Bases using Deep Learning. *CoRR.* 2018;abs/1811.04132. Available from: <http://arxiv.org/abs/1811.04132>.
48. Sukhbaatar S, Szlam A, Weston J, et al. End-to-End Memory Networks. In: Cortes C, Lawrence ND, Lee DD, et al., editors. Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada; 2015. p. 2440–2448. Available from: <https://proceedings.neurips.cc/paper/2015/hash/8fb21ee7a2207526da55a679f0332de2-Abstract.html>.
49. Vinyals O, Fortunato M, Jaitly N. Pointer Networks. In: Cortes C, Lawrence ND, Lee DD, et al., editors. Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada; 2015. p. 2692–2700. Available from: <https://proceedings.neurips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html>.
50. Kulmanov M, Liu-Wei W, Yan Y, et al. EL Embeddings: Geometric construction of models for the Description Logic \mathcal{EL}^{++} . In: Kraus S, editor. Proceedings of

- the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. ijcai.org; 2019. p. 6103–6109. Available from: <https://doi.org/10.24963/ijcai.2019/845>.
51. Bordes A, Usunier N, García-Durán A, et al. Translating Embeddings for Modeling Multi-relational Data. In: Burges CJC, Bottou L, Ghahramani Z, et al., editors. Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States; 2013. p. 2787–2795. Available from: <https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html>.
 52. Mondal S, Bhatia S, Mutharaju R. EmEL⁺⁺: Embeddings for \mathcal{EL}^{++} Description Logic. In: Martin A, Hinkelmann K, Fill H, et al., editors. Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021; (CEUR Workshop Proceedings; Vol. 2846). CEUR-WS.org; 2021. Available from: <http://ceur-ws.org/Vol-2846/paper19.pdf>.
 53. Mohapatra B, Bhatia S, Mutharaju R, et al. Why Settle for Just One? Extending \mathcal{EL}^{++} Ontology Embeddings with Many-to-Many Relationships. CoRR. 2021; abs/2110.10555. Available from: <https://arxiv.org/abs/2110.10555>.
 54. Xiong B, Potyka N, Tran T, et al. Faithful Embeddings for \mathcal{EL}^{++} Knowledge Bases. In: Sattler U, Hogan A, Keet CM, et al., editors. The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings; (Lecture Notes in Computer Science; Vol. 13489). Springer; 2022. p. 22–38. Available from: https://doi.org/10.1007/978-3-031-19433-7_2.
 55. Eberhart A, Ebrahimi M, Zhou L, et al. Completion Reasoning Emulation for the Description Logic \mathcal{EL}^+ . In: Martin A, Hinkelmann K, Fill H, et al., editors. Proceedings of the AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice, AAAI-MAKE 2020, Palo Alto, CA, USA, March 23-25, 2020, Volume I; (CEUR Workshop Proceedings; Vol. 2600). CEUR-WS.org; 2020. Available from: <http://ceur-ws.org/Vol-2600/paper5.pdf>.
 56. Özçep ÖL, Leemhuis M, Wolter D. Cone Semantics for Logics with Negation. In: Bessiere C, editor. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. ijcai.org; 2020. p. 1820–1826. Available from: <https://doi.org/10.24963/ijcai.2020/252>.
 57. Garg D, Iqbal S, Srivastava SK, et al. Quantum Embedding of Knowledge for Reasoning. In: Wallach HM, Larochelle H, Beygelzimer A, et al., editors. Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada; 2019. p. 5595–5605. Available from: <https://proceedings.neurips.cc/paper/2019/hash/cb12d7f933e7d102c52231bf62b8a678-Abstract.html>.
 58. Mehri R, Haarslev V. Applying machine learning to enhance optimization techniques for OWL reasoning. In: Artale A, Glimm B, Kontchakov R, editors. Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017; (CEUR Workshop Proceedings; Vol. 1879). CEUR-WS.org; 2017. Available from: <http://ceur-ws.org/Vol-1879/paper2.pdf>.
 59. Singh G, Mondal S, Bhatia S, et al. Neuro-Symbolic Techniques for Description Logic Reasoning (Student Abstract). In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9,

2021. AAAI Press; 2021. p. 15891–15892. Available from: <https://ojs.aaai.org/index.php/AAAI/article/view/17942>.
- 60. Birkhoff G, Von Neumann J. The Logic of Quantum Mechanics. Dordrecht: Springer Netherlands; 1975. p. 1–26. Available from: https://doi.org/10.1007/978-94-010-1795-4_1.
 - 61. Mehri R, Haarslev V, Chiniae H. Optimizing Heuristics for Tableau-based OWL Reasoners. CoRR. 2018;abs/1810.06617. Available from: <http://arxiv.org/abs/1810.06617>.
 - 62. Mehri R, Haarslev V, Chiniae H. Learning the Right Expansion-ordering Heuristics for Satisfiability Testing in OWL Reasoners. CoRR. 2019;abs/1904.09443. Available from: <http://arxiv.org/abs/1904.09443>.
 - 63. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. Nat. 2016;529(7587):484–489. Available from: <https://doi.org/10.1038/nature16961>.
 - 64. Hohenecker P, Lukasiewicz T. Ontology Reasoning with Deep Neural Networks. J Artif Intell Res. 2020;68:503–540. Available from: <https://doi.org/10.1613/jair.1.11661>.
 - 65. Parsia B, Matentzoglu N, Gonçalves RS, et al. The OWL Reasoner Evaluation (ORE) 2015 Competition Report. J Autom Reason. 2017;59(4):455–482. Available from: <https://doi.org/10.1007/s10817-017-9406-8>.
 - 66. Singh G, Mutharaju R, Kapanipathi P, editors. Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Event, October 27th, 2021; (CEUR Workshop Proceedings; Vol. 3123). CEUR-WS.org; 2022. Available from: <http://ceur-ws.org/Vol-3123>.
 - 67. Mohapatra B, Bhatia S, Mutharaju R, et al. EmELvar: A NeuroSymbolic Reasoner for the \mathcal{EL}^{++} Description Logic Reasoning. In: Singh G, Mutharaju R, Kapanipathi P, editors. Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Event, October 27th, 2021; (CEUR Workshop Proceedings; Vol. 3123). CEUR-WS.org; 2021. p. 44–51. Available from: <http://ceur-ws.org/Vol-3123/paper6.pdf>.
 - 68. Chowdhury S, Ebrahimi M, Eberhart A, et al. Memory Networks for RDFS reasoning: Experiments. In: Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2022) co-located with the 21st International Semantic Web Conference (ISWC 2021). CEUR-WS.org; 2022. CEUR Workshop Proceedings (to be published).
 - 69. Mehryar S, Celebi R. Improving Transitive Embeddings in Neural Reasoning Tasks via Knowledge-Based Policy Networks. In: Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2022) co-located with the 21st International Semantic Web Conference (ISWC 2021). CEUR-WS.org; 2022. CEUR Workshop Proceedings (to be published).
 - 70. Nicolas Heist HP. The CaLiGraph Ontology as a Challenge for OWL Reasoners. In: Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021); (CEUR Workshop Proceedings; Vol. 3123). CEUR-WS.org; 2021. p. 21–31. Available from: <https://ceur-ws.org/Vol-3123/paper3.pdf>.
 - 71. Ma L, Yang Y, Qiu G Z and Xie, et al. Towards a Complete OWL Ontology Benchmark. In: The Semantic Web: Research and Applications; Berlin, Heidelberg. Springer Berlin Heidelberg; 2006. p. 125–139.

72. Lehmann J, Isele R, Jakob M, et al. Dbpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*. 2014;6:1–6.
73. Suchanek FM, Kasneci G, Weikum G. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In: Proceedings of the 16th International Conference on World Wide Web; New York, NY, USA. Association for Computing Machinery; 2007. p. 697–706; WWW '07. Available from: <https://doi.org/10.1145/1242572.1242667>.
74. Vrandečić D. Wikidata: A new platform for collaborative data collection. In: Proceedings of the 21st International Conference on World Wide Web; New York, NY, USA. Association for Computing Machinery; 2012. p. 1063–1064; WWW '12 Companion. Available from: <https://doi.org/10.1145/2187980.2188242>.
75. Guo Y, Pan Z, Heflin J. LUBM: A Benchmark for OWL Knowledge Base Systems. *Journal of Web Semantics*. 2005;3(2-3):158–182.
76. Link V, Lohmann S, F H. OntoBench: Generating Custom OWL 2 Benchmark Ontologies. In: The Semantic Web – ISWC 2016; Vol. 9982; Cham. Springer International Publishing; 2016. p. 122–130. Available from: https://link.springer.com/chapter/10.1007/978-3-319-46547-0_13.
77. Singh G, Kumar A, Bhagat K, et al. OWL2Bench: Towards a Customizable Benchmark for OWL 2 Reasoners. In: Proceedings of the ISWC 2020 Demos and Industry Tracks: 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC); (CEUR Workshop Proceedings; Vol. 2721). CEUR-WS.org; 2020. p. 344–349. Available from: <http://ceur-ws.org/Vol-2721/paper587.pdf>.
78. Crouse M, Abdelaziz I, Makni B, et al. A Deep Reinforcement Learning Approach to First-Order Logic Theorem Proving. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. AAAI Press; 2021. p. 6279–6287. Available from: <https://ojs.aaai.org/index.php/AAAI/article/view/16780>.