Reasoning in Knowledge Graphs

Ricardo Guimarães
University of Bergen, Norway

Ana Ozaki
University of Bergen, Norway

Abstract

Knowledge Graphs (KGs) are becoming increasingly popular in the industry and academia. They can be represented as labelled graphs conveying structured knowledge in a domain of interest, where nodes and edges are enriched with metaknowledge such as time validity, provenance, language, among others. Once the data is structured as a labelled graph one can apply reasoning techniques to extract relevant and insightful information. We provide an overview of deductive and inductive reasoning approaches for reasoning in KGs.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Knowledge Graphs, Description Logics, Knowledge Graph Embeddings

Digital Object Identifier 10.4230/OASIcs.AIB.2022.2

Category Invited Paper

Funding *Ricardo Guimarães*: The first author is supported by the ERC project “Lossy Preprocessing” (LOPRE), grant number 819416, led by Prof. Saket Saurabh.
Ana Ozaki: The second author is supported by the NFR project “Learning Description Logic Ontologies”, grant number 316022.

Acknowledgements Part of this work has been done in the context of CEDAS (Center for Data Science, University of Bergen, Norway).

1 Introduction

Knowledge Graphs (KGs) [55, 61, 111, 119] are becoming increasingly popular in the industry and academia. They can be represented as labelled graphs conveying structured knowledge in a domain of interest, where nodes and edges are enriched with metaknowledge. Provenance and time validity are the most common kinds of metaknowledge. Since facts in KGs usually come from multiple datasources, it is important to record the provenance information, which is often in the format of an URL (or multiple URLs). Facts may happen multiple times and, therefore, a temporal dimension with time validity is important to record such information. Other kinds of metaknowledge include contextual information and language.

One of the most popular methods to model KGs consists in representing them as directed edge-labelled graphs [55, 61]. In this model, each entity in the domain of interest (people, places) are represented as nodes, while the different relations between pairs of these entities are expressed as as directed edges with a label that specifies the type of the relationship. Figure 1 depicts an example of such graph using entities such as *Artur Ávila* and *Brazil*, and relationships such as *citizenship*.

In addition to data models, there are concrete languages and systems which implement KGs. The Resource Description Framework (RDF) [79] is one of the most prominent of these languages. In fact, RDF is the W3C standard for writing KGs. A KG written in RDF, that is an *RDF graph*, is a collection of triples $(subject, predicate, object)$ that indicate that a relationship (given by the predicate) holds between the entities given as

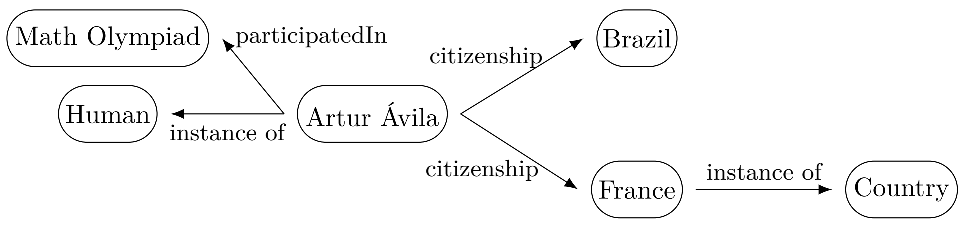


Figure 1 Example of directed edge-labelled graph.

subject and object. Example 1 depicts the RDF graph corresponding to the directed edge-labelled graph from Figure 1. Here, we will also refer to triples using a prefixed notation: $predicate(subject, object)$, which is closer to the logical formalisations discussed in Section 2.

▶ **Example 1.** We present below the RDF graph using Terse RDF Syntax, known as "Turtle", each line corresponds to a triple. The relationships (predicates) with prefix `rdf` and `rdfs` are imported from existing vocabularies that define the intended meaning of the terms.

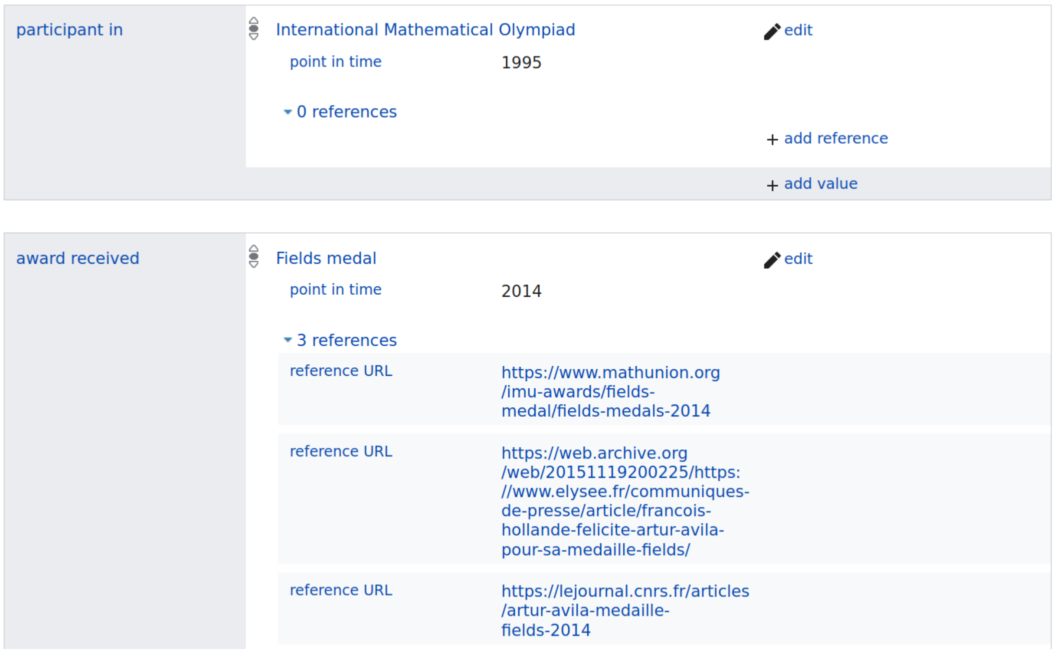
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://www.example.org/> .

# "Artur Avila" "is a" "Human"
:ArturAvila rdf:type :Human .
# "Artur Avila" "has country of citizenship" "France"
:ArturAvila :citizenship  :France .
# "Artur Avila" "has country of citizenship" "Brazil"
:ArturAvila :citizenship :Brazil .
# "France" "is a" "Country"
:France rdf:type :Country .
# "Artur Avila" "participated in" "Math Olympiad"
:ArturAvila :participatedIn :MathOlympiad .
```

As example of well-known KGs, we have Wikidata [112], a KG which contains much of the information displayed on Wikipedia. Example 2 discusses an excerpt of the Wikidata about Artur Avila.

▶ **Example 2.** Figure 2 contains some information about the mathematician Artur Ávila. Binary relations, such as `participantIn`, serve as labels to edges connecting nodes in the graph. In this example, the nodes would be the mathematician, the International Mathematical Olympiad, and the Fields medal. The fact that he participated in this event can be represented with the triple `participantIn`(`ArturAvila`, `MathOlympiad`). There is temporal metaknowledge associated with this fact which is the year of the participation, 1995. As we can see in Figure 2, not every fact is annotated with all the relevant metaknowledge, such as provenance information. The fact that Artur Ávila won the Fields medal is annotated with both temporal information and provenance (in the format of URLs).

Once the data is structured as a labelled graph one can apply reasoning techniques to extract relevant and insightful information. The three main classical types of reasoning are deduction, induction, and abduction:

- in deduction, we assume that a proposition or a formula α holds and, if $\alpha \to \beta$ is valid, we can deduce β;



participant in | International Mathematical Olympiad | edit
point in time | 1995
0 references
+ add reference
+ add value

award received | Fields medal | edit
point in time | 2014
3 references
reference URL | https://www.mathunion.org/imu-awards/fields-medal/fields-medals-2014
reference URL | https://web.archive.org/web/20151119200225/https://www.elysee.fr/communiques-de-presse/article/francois-hollande-felicite-artur-avila-pour-sa-medaille-fields/
reference URL | https://lejournal.cnrs.fr/articles/artur-avila-medaille-fields-2014

Figure 2 Excerpt of the Wikidata page of Artur Ávila (Q715043).

- in induction, we have that α and β holds and attempt to generalize the facts and generate a *rule*, which is an expression of the form $\alpha \to \beta$ (or $\beta \to \alpha$);
- finally, in abduction, we assume that a proposition or a formula β holds and, if $\alpha \to \beta$ is valid, we attempt to find an explanation α for β.

▶ **Example 3.** Consider the rule "if someone is a participant of an event then this person attended the event", which can be expressed in First Order logic (FOL) with the sentence $\forall x, y(\texttt{participantIn}(x, y) \to \texttt{attendant}(x, y))$ (and in Description Logic with the role inclusion $\texttt{participantIn} \sqsubseteq \texttt{attendant}$). Given the triple in Example 2 and the rule, we can deduce that Artur Ávila attended the event, in symbols, $\texttt{attendant}(\texttt{ArturAvila}, \texttt{MathOlympiad})$. If, instead of the rule, we have $\texttt{attendant}(\texttt{ArturAvila}, \texttt{MathOlympiad})$ and the triple in Example 2, then an inductive procedure could attempt to generalize the fact and generate the rule $\forall x, y(\texttt{participantIn}(x, y) \to \texttt{attendant}(x, y))$ (or $\forall x, y(\texttt{attendant}(x, y) \to \texttt{participantIn}(x, y))$, which can happen since inductive procedures can make wrong generalizations). Finally, given the rule and the fact $\texttt{attendant}(\texttt{ArturAvila}, \texttt{MathOlympiad})$, an abductive procedure could attempt to find the triple in Example 2 as an explanation for the fact $\texttt{attendant}(\texttt{ArturAvila}, \texttt{MathOlympiad})$.

While reasoning can often empower information retrieval [14, 63] and reasoners can work as query mechanisms, our focus is on reasoning rather than data retrieval. Pure database-like retrieval is strictly more constrained, as nothing that is not asserted can be derived (except in the case that the missing data is treated immediately as false). Yet, the line between the two is sometimes blurry [61]. Here, we focus on the main approaches commonly regarded as reasoning. We provide an overview of deductive (Section 2) and inductive (Section 3) reasoning approaches for reasoning in KGs. We do not cover query languages such as SPARQL [90], even though one can enrich SPARQL queries with reasoning capabilities. We conclude in Section 5.

2 Deductive Reasoning

Deductive reasoning in KGs is commonly performed by mapping its contents to a logic-based formalism. With this transformation in place, users and designers can apply reasoners for extracting logical consequences from the information specified in a KG. In this section, we discuss three of the most popular formalisms for deductive reasoning with KGs: Description Logics, Datalog and SHACL. Description Logics underpin the Web Ontology Language, OWL 2 [84], which is the current W3C standard for ontologies. Datalog corresponds to a family of languages inspired by the logical programming language Prolog, whose computational properties makes it attractive in many use-cases. Finally, SHACL is a more recent W3C recommendation designed to validate constraints in RDF graphs.

2.1 Description Logics

Description logics (DLs) are a family of knowledge representation formalisms [7]. Each DL has its own language with its own expressivity and, thus, different computational costs for different tasks. As already mentioned, DLs underpin OWL (including the current version OWL 2). Thus, the field of DL flourished together with the popularisation of ontologies as a way of sharing knowledge in disciplines such as Medicine and Biology. While the use-cases of ontologies and KGs diverge, the two are, nevertheless, related. KGs such as DBPedia incorporate ontologies [71] that help users and developers to understand the data.

In the following, whenever we refer to a DL ontology or a knowledge base, we mean a finite set of formulas (or axioms) in a DL. Despite their differences, these languages are very similar in the way which they are used to describe knowledge. Moreover, most DLs have decidable reasoning problems and are tailored for specific applications making DL ontologies valuable tools for deductive reasoning with KGs, even if DLs cannot capture everything that a KG can represent. In all DLs, the main ideas of the domain of interest are described via concept descriptions. These concept descriptions essentially create classes to which one can assign the elements of the domain. Then, a DL ontology will contain statements (formulas) that, among other things, determine how these concept descriptions relate to each other. We will discuss more about representing knowledge with DLs next.

Given a domain of interest, the first step is to determine the key notions to be described, which will then form a set of terms. These terms can be either concept names, role names or individual names. Concept names determine the basic groups to which elements of the domain may belong, role names the basic relationships between these elements and individual names refer to (some) elements of the domain. The set of terms in an ontology is called *signature*, composed of three pairwise disjoint sets: *concept names* ($\mathsf{N_C}$), *role names* ($\mathsf{N_R}$) and *individual names* ($\mathsf{N_I}$).

▶ **Example 4.** If we are modelling knowledge on notable scientists, following Figure 2, we could have among the atomic concepts in our signature `Scientist`, `Award` and `University`. We could also have roles such as `receivedAward` and `participantIn`. Finally, we would need individual names to refer to particular elements of the domain, such as `ArturAvila`, `FieldsMedal` and `MathOlympiad`.

These are just the fundamental building blocks, as it is possible to build complex concept descriptions (and sometimes even complex roles) by using a set of constructors, which varies according to the DL selected. Many DLs allow the ontology engineer to represent the conjunction (intuitively, the intersection) of two concepts, for instance, $\texttt{Scientist} \sqcap \texttt{Brazilian}$ to refer to Brazilian scientists. Table 1 lists the concept constructors allowed in the DL $\mathcal{ALC}$.

Table 1 Complex concepts in $\mathcal{ALC}$. $\mathtt{C}$, $\mathtt{D}$ are concept expressions and $\mathtt{r} \in \mathsf{N_R}$.

Name	Syntax	Semantic
Conjunction	$\mathtt{C} \sqcap \mathtt{D}$	$\mathtt{C}^\mathcal{I} \cap \mathtt{D}^\mathcal{I}$
Disjunction	$\mathtt{C} \sqcup \mathtt{D}$	$\mathtt{C}^\mathcal{I} \cup \mathtt{C}^\mathcal{I}$
Negation	$\neg\mathtt{C}$	$\Delta^\mathcal{I} \setminus \mathtt{C}^\mathcal{I}$
Existential Restriction	$\exists\mathtt{r}.\mathtt{C}$	$\{x \in \Delta^\mathcal{I} \mid \exists y.(x,y) \in \mathtt{r}^\mathcal{I} \wedge y \in \mathtt{C}^\mathcal{I}\}$
Value Restriction	$\forall\mathtt{r}.\mathtt{C}$	$\{x \in \Delta^\mathcal{I} \mid \forall y.(x,y) \in \mathtt{r}^\mathcal{I} \to y \in \mathtt{C}^\mathcal{I}\}$

Using concepts, roles and individuals, one can write formulas which express the constraints about the domain of interest. These formulas, or axioms, also vary according to the DL at hand. Regardless of the DL, ontologies are often split into two parts, the TBox and the ABox. The TBox contains the terminological knowledge, that is, the relationships between concepts and between roles, while the ABox contains the assertions, which concern characteristics of individuals. Table 2 lists the types of axioms allowed in $\mathcal{ALC}$. *Concept inclusions* and *equalities* are TBox axioms, while concept and role assertions are ABox axioms.

Table 2 Axioms in $\mathcal{ALC}$, $\mathtt{a}, \mathtt{b} \in \mathsf{N_I}$.

Name	Syntax	Semantics
Concept inclusion	$\mathtt{C} \sqsubseteq \mathtt{D}$	$\mathtt{C}^\mathcal{I} \subseteq \mathtt{D}^\mathcal{I}$
Concept equality	$\mathtt{C} \equiv \mathtt{D}$	$\mathtt{C}^\mathcal{I} = \mathtt{D}^\mathcal{I}$
Concept assertion	$\mathtt{C}(\mathtt{a})$	$\mathtt{a}^\mathcal{I} \in \mathtt{C}^\mathcal{I}$
Role assertion	$\mathtt{r}(\mathtt{a},\mathtt{b})$	$(\mathtt{a}^\mathcal{I},\mathtt{b}^\mathcal{I}) \in \mathtt{r}^\mathcal{I}$

Now, we make additional notes about the relationship between KGs and DL ontologies. As we mentioned in the beginning of this section, KGs may have ontologies associated to them. In this context, the "ontology" usually refers to the part of KG that corresponds to a TBox, while the remaining part of the KG corresponds to an ABox [61]. There are also components in KGs in general, even when considering RDF alone, that cannot be mapped into DL axioms (such as metadata, as discussed in Section 1). Example 5 continues our running example, with a simple $\mathcal{ALC}$ ontology which describes some constraints about the domain.

▶ **Example 5.**

$$\begin{aligned}
\mathtt{Scientist} &\sqsubseteq \mathtt{Human}\\
\mathtt{Human} &\sqsubseteq \neg\mathtt{University}\\
\mathtt{PhD} &\sqsubseteq \exists\mathtt{educatedAt}.\mathtt{University}\\
&\mathtt{Scientist}(\mathtt{ArturAvila})\\
&\mathtt{participantIn}(\mathtt{ArturAvila},\mathtt{MathOlympiad})
\end{aligned}$$

In the ontology above, the first axiom states that every scientist is a human, and the second says that no human is a university. The third axiom says that everyone with a PhD title should have been educated at some university. The last two are assertions: one states that Artur Ávila is a scientist and the other that he participated in the International Mathematical Olympiad.

In DLs, interpretations act as the most popular form of semantics. An interpretation is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is an arbitrary set of elements and $\cdot^{\mathcal{I}}$ is a function which maps each concept to a subset of $\Delta^{\mathcal{I}}$, each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name to an element of $\Delta^{\mathcal{I}}$. Each axiom in an ontology places a new constraint on interpretations that satisfy the ontology. Tables 1 and 2 describe how each axiom constrains the possible interpretations of an ontology. If an interpretation $\mathcal{I}$ complies with every requirement specified by an ontology $\mathcal{O}$, we say that $\mathcal{I}$ satisfies $\mathcal{O}$, in symbols $\mathcal{I} \models \mathcal{O}$. Example 6 clarifies this notion with an interpretation for the ontology in Example 5.

▶ **Example 6.** Consider the following interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ such that

$$\begin{aligned}
\Delta^{\mathcal{J}} &= \{\texttt{ArturAvila}, \texttt{Bob}, \texttt{IMO1995}, \texttt{UFRJ}\};\\
\texttt{Human}^{\mathcal{J}} &= \{\texttt{ArturAvila}, \texttt{Bob}\};\\
\texttt{Scientist}^{\mathcal{J}} &= \{\texttt{ArturAvila}\};\\
\texttt{PhD}^{\mathcal{J}} &= \{\texttt{ArturAvila}\};\\
\texttt{University}^{\mathcal{J}} &= \{\texttt{UFRJ}\};\\
\texttt{educatedAt}^{\mathcal{J}} &= \{(\texttt{ArturAvila}, \texttt{UFRJ})\};\\
\texttt{participantIn}^{\mathcal{J}} &= \{(\texttt{ArturAvila}, \texttt{IMO1995})\};\\
\texttt{ArturAvila}^{\mathcal{J}} &= \texttt{ArturAvila}; \text{ and}\\
\texttt{MathOlympiad}^{\mathcal{J}} &= \texttt{IMO1995}.
\end{aligned}$$

The interpretation $\mathcal{J}$ satisfies the ontology in Example 5.

2.1.1 Attributed DLs

While DLs provide an important advantage when reasoning with OWL ontologies, there are important features seen in real-world KGs that cannot be captured in a useful way in traditional DLs. OWL and RDF were designed using a directed edge-labelled model, which enforces every piece of data to be either a node (an entity or literal) or a relation. However, not only there are RDF graphs (and even OWL features) that cannot be expressed in DLs, but there are other graph data models which allow nodes and edges to be enriched with annotations. In a property graph (used for instance in Neo4J), each node and relation may have a map from keys to values as annotations. The Wikidata model is even more flexible, as whole statements (already more complex than assertions) may have different annotations (called qualifiers), with multiple values for the same type of qualification. For example, in Figure 2 the year 1995 is the value for the qualifier "point in time" for the statement "Artur Ávila participated in the International Mathematical Olympiad".

Attributed DLs [21, 22, 65, 68, 85] add non-functional attribute-value pairs to DL axioms (not only assertions) with the goal of representing annotations. This extension presupposes a set of variables $\mathsf{N_V}$ which can be used to build *specifiers*. These are expressions representing sets of annotations.

▶ **Definition 7.** *The set of specifiers* $\mathbf{S}$ *is the smallest set containing the following expressions:*
variables: X*;*
closed specifiers: $[a_1 : v_1, \ldots, a_n : v_n]$*; and*
open specifiers: $\lfloor a_1 : v_1, \ldots, a_n : v_n \rfloor$*;*
where $X \in \mathsf{N_V}$ *is a variable,* $n \in \mathbb{N}$*,* $a_i \in \mathsf{N_I}$ *and* $v_i \in \mathsf{N_I} \cup \{+\} \cup X.c$ *with* $c \in \mathsf{N_I}$.

The symbol "+" has the meaning "at least one". A closed specifier is satisfied iff the set of annotations matches exactly the specification, while an open specifier requires only that the annotations appear (others might occur as well).

▶ **Example 8.** Consider the statements in Figure 2. Using attributed DLs, we can express specifically the statement about Ávila's participation as follows:

$$\texttt{participantIn}(\texttt{ArturAvila}, \texttt{mathOlympiad})@[\texttt{pointInTime} : 1995].$$

However, to express the Fields medal award, an assertion with an open specifier would be preferred, to account for other annotations (e.g. provenance):

$$\texttt{awardReceived}(\texttt{ArturAvila}, \texttt{FieldsMedal})@\lfloor\texttt{pointInTime} : 2014\rfloor.$$

Given a classical, non-attributed, DL $\mathcal{L}$ (e.g. $\mathcal{ALC}$), its attributed version, $\mathcal{L}_{@+}$ (e.g. $\mathcal{ALC}_{@+}$), has essentially the same concept expressions as $\mathcal{L}$ except that concept and role names are associated with a specifier. For instance, since in $\mathcal{ALC}$ $A \sqcap B$ is a valid concept expression, we have that $A@S \sqcap B@S'$ is a valid concept expression in $\mathcal{ALC}_{@+}$, in which S, S' are specifiers as in Definition 7. Note that we can use an empty open specifier $\lfloor\rfloor$ if we do not want to constrain annotations.

▶ **Example 9.** The following are valid concept and role expressions in $\mathcal{ALC}_{@+}$:

- Brazilians: $\texttt{Brazilian}@\lfloor\rfloor$
- Doctors of Philosophy whose only annotation is start time 1995: $\texttt{PhD}@[\texttt{startTime} : 1995]$
- The "membership" relation with some starting time $\texttt{memberOf}@\lfloor\texttt{startTime} : +\rfloor$

In Example 9, we deliberately omitted specifiers of the form `X.c` as they are more intricate. We will clarify their meaning later when we discuss axioms in attributed DLs. Regarding concept and role expressions, given a standard DL $\mathcal{L}$, its attributed extension $\mathcal{L}_{@+}$ allows the following axioms:

- $\mathcal{L}_{@+}$ concept assertions are $A(a)@S$;
- $\mathcal{L}_{@+}$ role assertions are $r(a, b)@S$; and
- $\mathcal{L}_{@+}$ concept inclusions are $X_1 : S_1, \ldots, X_n : S_n(C \sqsubseteq D)$;

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, $a, b \in \mathsf{N_I}$, C and D are $\mathcal{L}_{@+}$ concept expressions, S is a specifier that is not a set variable, $X_1, \ldots, X_n \in \mathsf{N_V}$ and $S_1, \ldots, S_n$ are specifiers.

▶ **Example 10.** If we consider the DL $\mathcal{ALC}$, and its attributed version $\mathcal{ALC}_{@+}$ we can express the following statements:

- $\texttt{PhD}(\texttt{ArturAvila})@\lfloor\texttt{reference} : +\rfloor$
- $\texttt{memberOf}(\texttt{ArturAvila}, \texttt{USNAS})@[\texttt{startTime} : 2019]$
- $\texttt{X} : \lfloor\texttt{reference} : +\rfloor(\texttt{PhD}@[\texttt{reference} : \texttt{X.reference}] \sqsubseteq$
  $\exists\texttt{educatedAt}@\lfloor\texttt{reference} : \texttt{X.reference}\rfloor.\texttt{University})$

The semantics is also similar to standard DLs and it is given by interpretations, but modified to include annotation sets. Hence, terms are interpreted as follows:

- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathcal{P}_f(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$;
- $r^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \times \mathcal{P}_f(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$;
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$;

where $\mathcal{P}_f(\mathtt{S})$ is the set of all finite subsets of a set $\mathtt{S}$. Example 11 clarifies the meaning of interpreting terms alone (that is, without considering complex concepts nor specifiers).

▶ **Example 11.**

$\texttt{ArturAvila}^{\mathcal{J}} = \texttt{ArturAvila}$

$\texttt{CarlosChagas}^{\mathcal{J}} = \texttt{CarlosChagas}$

$\texttt{PhD}^{\mathcal{J}} = \{(\texttt{ArturAvila}, \{(\texttt{pointInTime}, 2001)\}), (\texttt{CarlosChagas}, \emptyset)\}$

$\texttt{memberOf}^{\mathcal{J}} = \{((\texttt{ArturAvila}, \texttt{USNAS}), \{(\texttt{startTime}, 2019), (\texttt{subjectRole}, \texttt{ForeignAssociate})\})\}$

$\texttt{Brazilian}^{\mathcal{J}} = \{(\texttt{ArturAvila}, \{(\texttt{reference}, \texttt{cv})\}), (\texttt{CarlosChagas}, \emptyset)\}$

$\texttt{French}^{\mathcal{J}} = \{(\texttt{ArturAvila}, \{(\texttt{reference}, \texttt{cv})\})\}$

The semantics of specifiers and other expressions containing free variables is defined using a *variable assignment*. A variable assignment is a function, $\mathcal{Z} : \mathsf{N_V} \mapsto \mathcal{P}_f(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$

$$\begin{aligned}
X^{\mathcal{I},\mathcal{Z}} &:= \{\mathcal{Z}(X)\};\\
[a : b]^{\mathcal{I},\mathcal{Z}} &:= \{\{(a^{\mathcal{I}}, b^{\mathcal{I}})\}\};\\
[a : X.b]^{\mathcal{I},\mathcal{Z}} &:= \{\{(a^{\mathcal{I}}, \delta^{\mathcal{I}}) \mid \exists \delta \in \Delta^{\mathcal{I}} \,:\, (b^{\mathcal{I}}, \delta) \in \mathcal{Z}(X)\}\};\\
[a : +]^{\mathcal{I},\mathcal{Z}} &:= \{\{(a^{\mathcal{I}}, \delta_1), \dots, (a^{\mathcal{I}}, \delta_\ell)\} \mid \ell \leq 1 \text{ and } \delta_i \in \Delta^{\mathcal{I}}\}\};\\
[a_1 : v_1, \dots, a_n : v_n]^{\mathcal{I},\mathcal{Z}} &:= \{\cup_{i=1}^{n} \psi_i \mid \psi_i \in [a_i : v_i]^{\mathcal{I},\mathcal{Z}}, 1 \leq i \leq n\};\\
\lfloor a_1 : v_1, \dots, a_n : v_n \rfloor^{\mathcal{I},\mathcal{Z}} &:= \{\psi \in \mathcal{P}_f(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \mid \psi \supseteq \phi \text{ for some } \phi \in [a_i : v_i]^{\mathcal{I},\mathcal{Z}}\}.
\end{aligned}$$

Given the semantics of terms and specifiers, concept names and role names with non-empty specifiers are interpreted as follows:

$$\begin{aligned}
A@S^{\mathcal{I},\mathcal{Z}} &:= \{\delta \in \Delta^{\mathcal{I}} \mid (\delta, \psi) \in A^{\mathcal{I}} \text{ for some } \psi \in S^{\mathcal{I},\mathcal{Z}}\};\\
r@S^{\mathcal{I},\mathcal{Z}} &:= \{(\delta_1, \delta_2) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (\delta_1, \delta_2, \psi) \in \mathtt{r}^{\mathcal{I}} \text{ for some } \psi \in S^{\mathcal{I},\mathcal{Z}}\}.
\end{aligned}$$

The concept constructors in an attributed DL are inherited from its standard counterpart, with the semantics extended to accommodate the annotations. Intuitively, a specifier will reduce the extension of a concept or role, that is, if $\mathtt{t} \in \mathsf{N_C} \cup \mathsf{N_R}$ and S is a specifier, then, for any interpretation $\mathcal{I}$: $\mathtt{t@S}^{\mathcal{I}} \subseteq \mathtt{t}@\lfloor\rfloor^{\mathcal{I}}$.

▶ **Example 12.** Consider the interpretation $\mathcal{J}$ from Example 11. Below we list the interpretation of the concept and role expressions from Example 9 in $\mathcal{J}$ using $\mathcal{Z}(X) = \{(\texttt{startTime} : 2000, \texttt{startTime} : 2019, \texttt{reference} : \texttt{cv})\}$.

$$\begin{aligned}
\texttt{Brazilian}@\lfloor\rfloor^{\mathcal{J},\mathcal{Z}} &= \{\texttt{ArturAvila}\}\\
\texttt{PhD}@[\texttt{startTime} : 1995]^{\mathcal{J},\mathcal{Z}} &= \emptyset\\
\texttt{memberOf}@\lfloor\texttt{startTime} : +\rfloor^{\mathcal{J},\mathcal{Z}} &= \{(\texttt{ArturAvila}, \texttt{USNAS})\}\\
\texttt{French}@[\texttt{reference} : \texttt{X.reference}]^{\mathcal{J},\mathcal{Z}} &= \{\texttt{ArturAvila}\}
\end{aligned}$$

The main drawback of attributed DLs is that even $\mathcal{ALC}_{@+}$ is undecidable, preventing the development of sound and complete reasoning mechanisms. Krötzsch et al. [68] proved that $\mathcal{ALCH}_{@}$ is still decidable, although its satisfiability problem lies in 2ExpTime. Bourgaux and Ozaki [21] proved that reasoning in a more restricted DL, called DL-Lite$^{\mathcal{R}}_{@}$, is in PSpace.

2.1.2 Description Logic Reasoners

Reasoning in DLs refers to obtaining implicit knowledge from explicit data. The formal semantics discussed before allows us to derive new information which respects the constraints defined in a DL ontology. For example, if we specify that "PhDs are humans" by adding to an ontology the axiom $\texttt{PhD} \sqsubseteq \texttt{Human}$, then from the assertion $\texttt{PhD}(\texttt{ArturAvila})$ we can conclude $\texttt{Human}(\texttt{ArturAvila})$, that is, that since Artur Ávila has a PhD title, he is a human.

Reasoning with DL ontologies is a mature and active field of study. There are already many well-known reasoners such as ELK [59], HermiT [100], Pellet [101], FaCT++ [107], Konclude [102] and RDFox [82] used in academia and in the industry. Some of these reasoners are benchmarked in standard reasoning tasks so that both users and developers can have an overview of the empirical state-of-art [89, 97]. There are also many prototypical implementations, which are tailored to specific extensions of DLs which still do not have a widespread adoption.

The DL reasoners differ in a number of factors, for example, ELK can only handle OWL 2 EL ontologies, which are restricted to the DL $\mathcal{EL}$++. Besides differing according to their target language (e.g. $\mathcal{EL}$++), they may also be based on different reasoning techniques (e.g. tableaux, hyper-tableaux, consequence-based reasoning) [7]. Other reasoners specialise in performance, for instance, by giving approximate results such as the case of TrOWL [105].

2.2 Datalog

Datalog is a query language derived from the logical programming language Prolog [30]. It was actively studied in the late 80s an early 90s, in the database community as an implementation of recursive queries. More recently, its interest has been revived for applications for querying graph-like data structures in ontology-based data access [14] and query answering [3, 27]. Besides the ability to express recursive queries, Datalog has polynomial complexity for many reasoning tasks which relate to queries and motivate its use today for querying databases, ontologies, and KGs. There are many extensions of Datalog [26, 27, 44, 94] but here we will focus on "pure" Datalog and its relevant modifications for handling KGs and ontologies.

2.2.1 Syntax

We begin the formalisation of Datalog with the signature. We assume three pairwise disjoint sets of symbols: a set of variables $\mathbf{V}$, a set of constants $\mathbf{C}$, and a set of predicates $\mathbf{P}$. Each predicate is represented as P/n where P is its name and n its arity. A *term* in Datalog is either a variable or a constant. An *atom* is an expression of the form $P(x_1, \ldots, x_n)$ where $P/n \in \mathbf{P}$ and $x_i \in \mathbf{C} \cup \mathbf{V}$ for $1 \leq i \leq n$. Using atoms we can write *rules*. A rule r is an expression of the form $H \leftarrow B_1, \ldots, B_m$ where H is the *head* atom and $B_1, \ldots, B_n$ are *body* atoms or *subgoals* [2, 108]. In Datalog, these rules must be *safe*, that is, each variable that appears in the head of the rule appears in its body. A finite collection of safe rules constitutes a *Datalog program*.

▶ **Example 13.** The following rule says that if someone teaches at a university then he/she is a lecturer:

$$\texttt{Lecturer}(\texttt{x}) \leftarrow \texttt{TeachesAt}(\texttt{x}, \texttt{y}), \texttt{University}(\texttt{y}).$$

The rule above is safe because the variable x appears both in the head and in the body. As an example of an unsafe rule, consider a rule which says that a worker is someone who works at some place:

$$\texttt{WorksAt}(\texttt{x}, \texttt{y}) \leftarrow \texttt{Worker}(\texttt{x}).$$

If an atom has no variables we call it a *fact*, for instance, we can state that Artur Ávila is a PhD with `PhD(ArturAvila)`. A Datalog *database instance* is a finite collection of facts. The safety constraint for rules and the requirement that every database contains only facts ensures that the logical derivations in a Datalog system are finite. Meaning that, in a finite number of applications of the rules over the database instance, one can derive every possible fact that is entailed.

▶ **Example 14.** In this example, we want to use Datalog to know if Artur Ávila has an Erdős number. The following rules are useful for knowing if someone has an Erdős number

$$\begin{aligned}\texttt{HasErdosNumber(x)} &\leftarrow \texttt{HasErdosNumber(y)}, \texttt{CoAuthor(x,y)}.\\ \texttt{CoAuthor(x,y)} &\leftarrow \texttt{CoAuthor(y,x)}.\end{aligned}$$

And then, we can state some basic facts

$$\begin{aligned}&\texttt{HasErdosNumber(PaulErdos)}.\\ &\texttt{CoAuthor(ArturAvila,BarrySimon)}.\\ &\texttt{CoAuthor(VilmosTotik,BarrySimon)}.\\ &\texttt{CoAuthor(VilmosTotik,PaulErdos)}.\\ &\texttt{CoAuthor(ArturAvila,WellingtonDeMelo)}.\end{aligned}$$

Example 14 shows one interesting feature of Datalog: recursion. In the first rule, for example, the predicate `HasErdosNumber/1` appears both in the head and in the body of the same rule. In pure Datalog, recursion does not complicate the semantics much, and in many cases a recursive program can be rewritten without it [2]. However, some complications occur in Datalog extensions (as some mentioned later in this section).

2.2.2 Semantics

Semantics in Datalog can be defined in many ways: with a model-theoretic approach, via fixpoint semantics, or based on proofs. Here, we introduce the first of these, as it relates more closely to the semantics of Description Logics introduced in Section 2.1. To understand the semantics, we must first map each rule to a sentence in FOL. Given a Datalog rule $H \leftarrow B_1, \ldots, B_n$, its corresponding FOL sentence is:

$$\forall x_1, \ldots, x_m \, ((B_1(\mathbf{u}_1) \wedge \cdots \wedge B_n(\mathbf{u}_n)) \rightarrow H(\mathbf{u}_h))$$

where $\mathbf{u}_h$ is the sequence of variables that appear in H, $\mathbf{u}_i$ is the sequence of variables that appear in B_i, and $x_1, \ldots, x_m$ are the variables occurring in the rule. Moreover, each fact corresponds to an atomic formula in FOL.

Then, we will need to consider only *Herbrand interpretations.* A Herbrand interpretation maps each constant to its own name and each n-ary predicate to a subset of $\mathbf{C}^n$. In this way, a Herbrand interpretation can be identified with the set of facts that it satisfies. An interpretation $\mathcal{I}$ satisfies a database instance I, if $I \subseteq \mathcal{I}$. An interpretation $\mathcal{I}$ satisfies a rule $H \leftarrow B_1, \ldots, B_n$ if for every variable substitution by constants θ, $\{B_1\theta, \ldots, B_n\theta\} \subseteq \mathcal{I}$ implies $H\theta \in \mathcal{I}$. Finally, an interpretation satisfies a program P if it satisfies every rule. We denote the satisfaction relation with an infix operator $\models$, where $\mathcal{I} \models X$ indicates that the interpretation $\mathcal{I}$ satisfies X, where X can be a fact, a database instance, a rule, a program, or a set combining those.

▶ **Example 15.** Consider the two following Herbrand interpretations:

$$\begin{aligned}\mathcal{I}_1 = \{\, &\texttt{PhD}(\texttt{ArturAvila}),\\ &\texttt{Scientist}(\texttt{ArturAvila}),\\ &\texttt{educatedAt}(\texttt{ArturAvila}, \texttt{UFRJ}),\\ &\texttt{participantIn}(\texttt{ArturAvila}, \texttt{MathOlympiad})\,\}\end{aligned}$$

$$\mathcal{I}_2 = (\mathcal{I}_1 \cup \{\texttt{University}(\texttt{UFRJ})\}) \setminus \{\texttt{Scientist}(\texttt{ArturAvila})\}.$$

The interpretation $\mathcal{I}_1$ satisfies the fact $\texttt{Scientist}(\texttt{ArturAvila})$, but it does not satisfy $\texttt{University}(\texttt{UFRJ})$, which is only satisfied by $\mathcal{I}_2$. Moreover, the rule

$$\texttt{University}(\texttt{y}) \leftarrow \texttt{educatedAt}(\texttt{x}, \texttt{y}) \wedge \texttt{PhD}(\texttt{x}),$$

is only satisfied by $\mathcal{I}_2$ as $\texttt{University}(\texttt{UFRJ}) \notin \mathcal{I}_1$. The rule $\texttt{PhD}(\texttt{x}) \leftarrow \texttt{Scientist}(\texttt{x})$ is satisfied by both interpretations. $\mathcal{I}_1$ satisfies both $\texttt{PhD}(\texttt{ArturAvila})$ and $\texttt{Scientist}(\texttt{ArturAvila})$, and $\mathcal{I}_2$ satisfies the rule vacuously because $\texttt{Scientist}(\texttt{ArturAvila}) \notin \mathcal{I}_2$.

Example 15 shows that we can have multiple interpretations for a given Datalog program and database instance. We can associate each pair of Datalog program and database instance (P, I) with the set of all Herbrand interpretations that satisfy them $HI(P, I)$. Interestingly, we can take the intersection of these models (even if there are infinitely many) and obtain the minimal Herbrand interpretation for (P, I) [30]. This allows to assign the following meaning to the consequences ($cons$) of a program P with database instance I in terms of all possible Herbrand models as follows

$$cons(P, I) = \bigcap_{\mathcal{I} \in HI(P,I)} \mathcal{I}$$

The semantics as presented here confer Datalog an interesting property: the unique name assumption (UNA). The name of each constant identifies it uniquely. In DLs, the convention is that there is no UNA, which can cause confusion regarding the possible interpretations of an ontology. Arguably, UNA matches the intended meaning in database applications [2].

2.2.3 Datalog Extensions

Pure Datalog, as we have presented, often lacks expressive power to represent more complex rules (or queries in a database point of view). To cover these gaps, many extensions of Datalog have been developed [2, 26, 44].

Many Datalog extensions allow rule atoms to appear negated, which gives the possibility to express interesting relationships. However, these Datalog variants have to deal with two issues that arise from negation. The first concerns groundings, replacements of variables by constants, on infinite domains. The second regards the actual semantics of Datalog programs and happens because the intersection of Herbrand interpretations ceases to characterise the consequences of Datalog with negation in a meaningful way. There are many different approaches to circumvent both issues [2], sometimes using semantics that differ considerably from the one presented here for pure Datalog. In other cases, the issues are solved by employing a similar semantics but placing syntactic restrictions on the usage of negation [2, 30]. A complete categorisation of the different possibilities (and thus, of Datalog variants) is beyond the scope of this work, and we refer the reader to classical references on Datalog for the details [2, 108].

Many Datalog implementations also include built-in predicates which represent, for example, arithmetic operations (e.g. sum, subtraction) and numeric comparisons (such as less than, $<$). These elements may also induce infinite groundings, so their syntactic use is usually constrained in a similar way as it happens with negation.

There are also Datalog extensions which focus on temporal reasoning [23,34,51]. Nowadays, temporal extensions of Datalog, and associated reasoners, often address the *Stream Reasoning* paradigm [38, 113]. In Stream Reasoning, the goal is to perform inferences of a (usually) rapid stream of data. In [11], for example, the authors employ DatalogMTL to formalise reasoning tasks that must account for time. Such tasks involve calculating or counting values over a period of time, for instance, to compute the revenue per year of a company [11].

2.2.4 Datalog and Description Logics

Datalog and Description Logics are not completely disjoint fields. There have been many studies on how these two formalisms relate [43, 67, 94, 95], in particular because Datalog materialisation, a form of inferring new facts, is very efficient and can be applied to reasoning problems involving ontologies [27, 28] and KGs [110].

Datalog$^{\pm}$, for instance, is a family of Datalog variants designed to capture the expressivity of Description Logics in the *DL-Lite* family. *DL-Lite* is composed mostly of lightweight DLs tailored for querying large collections of assertional data, while the concept inclusions are kept relatively simple. In fact, Datalog$^{\pm}$ is strictly more expressive than *DL-Lite* [27]. Krötzsch, Rudolph and Schmitt [67] investigated fragments of Description Logics (such as $\mathcal{ALC}$ and $\mathcal{SROIQ}$) that can be captured in Datalog, while Rosati [94] proposes a framework for integrating DLs with Datalog$^{\neg\vee}$, that is, Datalog extended with negation in the body and boolean disjunction in the head.

The *Semantic Web Rule Language (SWRL)* [57] closely relates DLs and Datalog. More specifically, this language is based both on OWL DL (a fragment of the first OWL proposal) and RuleML (which is essentially a fragment of Datalog. The resulting language is very expressive, so much that reasoning with SWRL with very expressive DLs such as $\mathcal{SROIQ}$ is undecidable. There are, however, subsets of SWRL that retain decidability, even when combined with $\mathcal{SROIQ}$, the so called *DL rules* [66].

2.2.5 Reasoning with Datalog

There are numerous implementations of Datalog embedded in database management systems and inference engines (e.g. RDFox [82], Apache JENA[1]) or available as libraries (e.g. pyDatalog[2], Datalog in Racket[3]). The implementations vary regarding the extensions implemented, the semantics adopted, and the actual syntax of the Datalog rules. Next, we discuss some of the most recent reasoning systems to perform inferences in Datalog rules and KGs.

RDFox [82] is a column store which employs a parallel materialisation method for Datalog reasoning. Materialisation is a popular strategy to save time in reasoning systems. Every entailment computed is stored instead of being discarded after each query, saving costs in subsequent calls. RDFox is designed to manage large volumes of RDF data and provide reasoning services using Datalog extensions tailored for this use case.

[1] https://jena.apache.org/
[2] https://sites.google.com/site/pydatalog/
[3] https://docs.racket-lang.org/datalog/

Another notable Datalog system built with KGs as its primary use case is Vadalog [12]. It focusses on Warded Datalog$^{\pm}$, a variant of Datalog with the three highly desirable properties: (1) ability to express recursive patterns, (2) enough expressivity to capture queries over KGs in fragments of *DL-Lite* (more precisely, SPARQL queries with OWL 2 QL entailment), and (3) polynomial data complexity.

Carral et al. [29] devised a rule engine with reasoning services based on Datalog materialisation. The engine is designed for an extension of Datalog similar to Datalog$^{\pm}$ and targets KGs as its main use-cases. One of the aspects that distinguishes this tool from other approaches mentioned here is the ability to easily integrate with data sources in different formats such as OWL ontologies, SPARQL endpoints, and RDF stores.

As we mentioned in the previous sections, KGs often include uncertain information and time-sensitive data. Chekol et al. [32] devised a framework combining Markov Logic Networks (MLNs) [92] and Datalog extended with inequalities to facilitate time-aware maintenance of KGs. MLNs are a Statistical Relational Learning approach which combines First-Order Logic and probabilities to represent dependencies between events, while they still remain uncertain. In a similar vein, Bellomarini et al. [10] extend the Vadalog system to express probabilistic rules. Recently, Wang et al. [114] developed a DatalogMTL reasoner by combining traditional reasoning methods for Datalog (materialisation) and automata-based methods.

Leone et al. [73] adapted the DLV2 answer set programming system, which already had Datalog capabilities, to handle large KGs (in the paper, the authors focus on DBPedia). These improvements concern mostly scalability, reducing processing time and memory consumption.

2.3 SHACL

The Shape Constraint Language (SHACL) is a W3C recommendation [62] whose purpose is to validate RDF graphs. Each SHACL constraint is called a *shape* which specifies to which nodes it applies and what conditions such nodes must satisfy. Given a KG in RDF, one can use different tools to check that the KG complies with a set of shapes. KGs generally lack a schema when compared with a database. Therefore, constraints are crucial for quality assessment and maintenance of large KGs [88]. SHACL and similar languages, such as ShEx [17], are tailored for the particular task of constraint validation, whereas OWL is tailored for modelling domains of knowledge and for reasoning. However, as we discuss later in this section, one can define reasoning problems and perform inference using SHACL. Next, we illustrate the capabilities of SHACL with an example.

▶ **Example 16.** The RDF graph below, written in terse triple notation, describes a single shape using SHACL (we omit the prefixes' declarations here).

```
:HumanShape rdf:type sh:NodeShape;
    sh:targetClass  :Human; # applies to all humans
    sh:property [
        sh:path :birthDate ; #  predicate for the date of birth
        sh:maxCount 1 ; # must have at most 1
        sh:minCount 1 ; # must have at least 1
        sh:datatype xsd:date ; # must be a date
    ];
    sh:property [
        sh:path :citizenship ; # citizenship predicate
        sh:minCount 1 ; # must have at least 1
        sh:node [ a sh:NodeShape;
                  sh:class :Country];  # must be a country
    ] .
```

The first defines the name of the shape (`:HumanShape`). The shape *targets* the nodes whose type (`rdf:type`) is the node `:Human` (Line 2). It also specifies that each such node must have exactly one birth date and it must be a valid date (lines 3 to 8). Lines 9 to 14 state that the target nodes must (1) be related via `:citizenship` to at least one node and (2) be related by `:citizenship` only to objects of type `:Country` (this specification is done by using an anonymous shape in lines 12 and 13).

Programs called *SHACL processors* check whether an RDF graph complies with a set of shapes. These programs generate a validation report for a given graph and set of shapes, indicating which nodes and constraints are violated, if any. Example 17 depicts an example of validation report.

► **Example 17.** A SPARQL processor receiving shapes from Example 16 and the RDF graph from Example 1 could produce the validation report below[4].

```
1   [
2       a sh:ValidationResult;
3       sh:resultSeverity sh:Violation;
4       sh:sourceConstraintComponent sh:MinCountConstraintComponent;
5       sh:sourceShape _:n51;
6       sh:focusNode <http://www.example.org/ArturAvila>;
7       sh:resultPath <http://www.example.org/birthDate>;
8       sh:resultMessage "Less than 1 values";
9   ] .
10  [
11      a sh:ValidationResult;
12      sh:resultSeverity sh:Violation;
13      sh:sourceConstraintComponent sh:NodeConstraintComponent;
14      sh:sourceShape _:n52;
15      sh:focusNode <http://www.example.org/ArturAvila>;
16      sh:value <http://www.example.org/Brazil>;
17      sh:resultPath <http://www.example.org/citizenship>;
18      sh:resultMessage "Value does not have shape Blank node _:n53";
19  ] .
```

The report specifies two violations: one says that the RDF graph does not include a birth date for `:ArturAvila` (lines 1 to 9) and the other (lines 10 to 19) indicates that the graph does not guarantee that `:Brazil` is a country.

2.3.1 DL-like Syntax for SHACL

Since we are interested in SHACL from the point of view of reasoning, we will look at the SHACL formalisation by Jakubowski and Van den Bussche [16] which adapts the one by Corman, Reutter and Savković [35]. This alternative syntax is much closer to that presented for Description Logics in Section 2.1, and it will aid us in describing SHACL's semantics and its connections to the DLs.

First, we need three pairwise disjoint sets of node names ($\mathsf{N_N}$), shape names ($\mathsf{N_S}$) and property names ($\mathsf{N_P}$). A *signature* Σ is be subset of $\mathsf{N_N} \cup \mathsf{N_S} \cup \mathsf{N_P}$. Using these terms, one can define path expressions which are either property names or built as in Table 3 (these path expressions essentially come from SPARQL [35, 62]). Finally, we can consider shape expressions: terms in $\mathsf{N_S}$ or expressions using the constructors in Table 4.

[4] The validation report was generated at `https://shacl.org/playground/`.

Table 3 Syntax and semantics of abstract SHACL path expressions [16].

Name	Syntax	Semantics
Inverse	$\mathtt{p}^-$	$\{(a,b) \mid (b,a) \in \mathtt{p}^\mathcal{I}\}$
Union	$\mathtt{E}_1 \cup \mathtt{E}_2$	${\mathtt{E}_1}^\mathcal{I} \cup {\mathtt{E}_2}^\mathcal{I}$
Composition	$\mathtt{E}_1 \circ \mathtt{E}_2$	$\{(a,b) \mid \exists c.(a,c) \in {\mathtt{E}_1}^\mathcal{I} \wedge (c,b) \in {\mathtt{E}_2}^\mathcal{I}\}$
Reflexive closure	$\mathtt{E}?$	$\mathtt{E}^\mathcal{I} \cup \{(a,a) \mid a \in \Delta^\mathcal{I}\}$
Reflexive-transitive Closure	$\mathtt{E}^*$	The $\subseteq$-minimal S with $\mathtt{E}?^\mathcal{I} \subseteq S$ and $(a,b),(b,c) \in S$ implies $(a,c) \in S$

Table 4 Syntax and semantics of abstract SHACL shape expressions [16]. $\mathcal{I}$ is an interpretation defined over the signature Σ, $\mathtt{E}$ is a path expression, $\mathtt{p} \in \mathsf{N_P}$, $\mathtt{s} \in \mathsf{N_S}$, $\mathtt{c} \in \mathsf{N_N}$ $n \in \mathbb{N}$, $Q \subseteq \mathsf{N_P}$ and ϕ_1, ϕ_2 are shape expressions. Also, $R(a)$ denotes the set $\{b \mid (a,b) \in R\}$, where R is a binary relation.

Name	Syntax	Semantics
Top	$\top$	$\Delta^\mathcal{I}$
Atomic Shape	$\mathtt{s}$	$\mathtt{s}^\mathcal{I}$
Nominal	$\{\mathtt{c}\}$	$\mathtt{c}^\mathcal{I}$
Conjunction	$\phi_1 \wedge \phi_2$	$\phi_1^\mathcal{I} \cap \phi_2^\mathcal{I}$
Disjunction	$\phi_1 \vee \phi_2$	$\phi_1^\mathcal{I} \cup \phi_2^\mathcal{I}$
Complement	$\neg\phi_1$	$\Delta^\mathcal{I} \setminus \phi_1^\mathcal{I}$
Minimum cardinality	$\geq n\mathtt{E}.\phi_1$	$\{a \in \Delta^\mathcal{I} \mid (\#(\phi_1^\mathcal{I} \cap \mathtt{E}^\mathcal{I}(a))) \geq n\}$
Equality	$eq(\mathtt{p}, \mathtt{E})$	$\{a \in \Delta^\mathcal{I} \mid \mathtt{p}^\mathcal{I}(a) = \mathtt{E}^\mathcal{I}(a)\}$
Disjointness	$disj(\mathtt{p}, \mathtt{E})$	$\{a \in \Delta^\mathcal{I} \mid \mathtt{p}^\mathcal{I}(a) \cap \mathtt{E}^\mathcal{I}(a) = \emptyset\}$
Closed shape	$closed(Q)$	$\{a \in \Delta^\mathcal{I} \mid \mathtt{p}^\mathcal{I}(a) = \emptyset$ for every $\mathtt{p} \in \Sigma \setminus Q\}$

We can regard a set of shapes, as the one in Example 16, as a set of statements relating shape expressions. These statements can be of two types *shape definitions* and *shape constraints*. Shape definitions are expressions of the form $\mathtt{s} \equiv \phi_\mathtt{s}$ with $\mathtt{s} \in \mathsf{N_S}$ and $\phi_\mathtt{s}$ a shape expression. Shape constraints are statements of the form $\phi_1 \sqsubseteq \phi_2$. A set of shapes can be represented as a pair of sets (D, T) called *shape schema* [6], in which D is a set of shape definitions and T a set of shape constraints. The shape schema must satisfy some syntactical requirements to ensure that it represents a valid SHACL document, we refer the reader to [6, 16] for more details.

▶ **Example 18.** This is a translation of the SHACL constraints over the citizenship relation in Example 16.

$$\neg(\geq 1.\mathtt{citizenship}.\neg(\{\mathtt{country}\})) \wedge (\geq 1.\mathtt{citizenship}.\top)$$

The first conjunct states that the target can only be related via `citizenship` to countries `Country`. The second specifies that there must be at least one object to which the target relates to via citizenship.

2.3.2 Semantics

When considering SHACL semantics, we have to make distinctions similar to the Datalog case. However, in this case the distinction concerns recursion. A shape is recursive if it refers to itself directly or indirectly, otherwise, it is non-recursive. The W3C recommendation [62] describes the language precisely enough to specify the meaning of validation against *non-recursive* shapes, however, the semantics of validation containing recursive shapes is open. This gap in SHACL's semantics sparked a number of distinct approaches attempting to clarify the validation of recursive SHACL [6, 15, 35].

For the purposes of this work, it will be enough to consider the semantics for non-recursive SHACL alone. Here, we will also focus on the formalisation proposed by Bogaerts, Jakubowski and Van den Bussche [16]. Thus, we will present the semantics using interpretations, which is also similar to the ones seen in Section 2.1. However, before we proceed, we remark two key differences between standard DL (and OWL) reasoning and SHACL reasoning. The first is the presence of the UNA in SHACL [62]. This means that every term in $\mathsf{N_N} \cup \mathsf{N_P}$ must be mapped to a different element of the domain in SHACL [16], similarly to the case of many Datalog semantics [2]. Furthermore, even blank nodes in a RDF graph (i.e. those who do not have an IRI) must be mapped into distinct elements of the domain [35].

The second main difference is that relations in SHACL are closed, meaning that if the graph does not have a triple (subject, predicate, object), then it is assumed to be false. More precisely, each triple can be seen as a fact (similar to the meaning in Datalog) or an assertion (as in DL ABoxes). For example, the triple (Artur Ávila, memberOf, USNAS) represents the fact that Artur Ávila is member of the United States' National Academy of Science (USNAS). If such triple would not be present in the Wikidata's KG, then SHACL would assume that Artur Ávila is not a member of the USNAS when validating the KG against a set of shapes. This is not the case in standard OWL and DL reasoning which follows the Open World Assumption (OWA). With the OWA, unless there is a statement or its negation (either asserted or inferred), the status of any triple is assumed to be unknown. As Bogaerts, Jakubowski and Van den Bussche [16] remark, this does not mean that SHACL presupposes the Closed World Assumption (CWA) completely. Triples about unknown entities could still be true or false. For instance, if the KG did not include any reference to Artur Ávila, then the status of the triple (Artur Ávila, memberOf, USNAS) would be "unknown", thus there could be an interpretation for that KG in which the triple is true, while another interpretation would consider it false.

Now, for the actual semantics, we consider that a KG G is a finite set of triples (subject, predicate, object), or equivalently, facts of the form $\mathit{predicate}(\mathit{subject}, \mathit{object})$. Given a KG G, we write N_G to denote the set of nodes (entities that appear as subject or object) that occur in G, and $\mathtt{p}^G$ to represent the set $\{(\mathtt{a}, \mathtt{b}) \mid (\mathtt{a}, \mathtt{p}, \mathtt{b}) \in G\}$. One can build an interpretation $\mathcal{I}(G) = (\Delta^{\mathcal{I}(G)}, \cdot^{\mathcal{I}(G)})$ over $\mathsf{N_N} \cup \mathsf{N_P}$ based on the KG G that preserves the intended meaning of the triples and shapes as follows [16]

- $\Delta^{\mathcal{I}(G)} = \mathsf{N_N}$;
- $\mathtt{c}^{\mathcal{I}(G)} = \mathtt{c}$, for all $\mathtt{c} \in \mathsf{N_N}$; and
- $\mathtt{p}^{\mathcal{I}(G)} = \mathtt{p}^G$, for all $\mathtt{p} \in \mathsf{N_P}$.

The semantics for complex path expressions follows the rules stated in Table 3. Shapes are mapped to subsets of $\Delta^{\mathcal{I}(G)}$ using the rules described in Table 4.

2.3.3 Reasoning with SHACL

The main reasoning task in SHACL is validation of constraints, that is, given an RDF graph which acts as an interpretation, one must decide whether it satisfies a set of constraints defined by shapes. As mentioned earlier, there are programs called SHACL processors which are either standalone programs [47], or embedded in larger systems for managing KGs, such as RDFox [82]. We list the most prominent of these reasoning tasks below (see also [5, 72, 87, 88]).

Shape containement: given two shapes ϕ_1 and ϕ_2, decide whether every target node that complies with ϕ_1 will also comply with ϕ_2. If so then we say that ϕ_1 is contained in ϕ_2.

SHACL satisfiability: given a set of shapes S, decide whether there exists G that complies with all shapes in S.

Explanation for Violations: Given a set of shapes S and a KG G, decide whether (B, A) with $B \subseteq G$ and $A \cap G = \emptyset$ is such that $(G \setminus B) \cup A$ complies with S.

Leinberger et al. [72] studies the decision problem of shape containment through DLs using the abstract formalisation of SHACL proposed by Corman, Reutter and Savković [35] that we presented here. More precisely, they map the problem of shape containment to concept subsumption, a well known decision problem in DLs [7]. The problem of shape containment was later generalised by Pareti et al. [88]. The authors studied the containment problem between sets of shapes and also proposed and investigated SHACL satisfiability and a more restricted version, constraint satisfiability. The tasks studied were restricted to non-recursive SHACL, but they are also being extended to consider recursive shapes [87].

While explanation finding and associated tasks are classified as abductive reasoning, we find it convenient to mention here the work due to Ahmetaj et al. [5]. The authors define and investigate the computational complexity of many reasoning problems related to explanations in SHACL, as the one we mentioned before. Their theoretical framework rely on a primarily deductive approach which is also underpinned by the DL abstraction of SHACL.

3 Inductive Reasoning

Inductive reasoning in KGs can be performed with different approaches. One of the main goals of this task is to "complete" the KGs. KG completion is the task of inferring new facts which are plausible based on patterns already present in a given KG. In this section, we discuss classical approaches for KG completion, namely, knowledge graph embeddings [20] (see [55] for some approaches using graph neural networks).

3.1 Knowledge Graph Embeddings

An embedding in Representation Learning (a subarea of Machine Learning) corresponds to a mapping from a collection of objects to a vector space model, often low-dimensional ones. The first approaches for knowledge graph embeddings mapped both objects (corresponding to nodes in the KG) and relations to vectors into a low-dimensional latent space encoding regularities in the KG [20]. These methods also have a score function that determines if a given triple is likely to be true or not, given the embeddings learned. Here we will assume, without loss of generality, that the higher the score, the more likely the triple is to be true. The score function is also the main component of the loss function that should be minimised during the training phase.

▶ **Example 19.** Consider a KG G that contains the entities $ArturAvila$, $CarlosChagas$, $NiedeGuidon$, $archelogist$, $biologist$ and $mathematician$; and the relation $occupation$. Ideally, an embedding method should learn vector representations (embeddings) for entity and relations that allows us to infer new and correct triples. For instance, suppose that G contains $(ArturAvila, occupation, mathematician)$ and $(CarlosChagas, occupation, biologist)$, but does not include $(NiedeGuidon, occupation, archeologist)$. If the KG also includes $NiedeGuidon$ in other triples, then we would expect that we could derive the missing triple by looking at the embeddings for $NiedeGuidon$, $occupation$ and $archeologist$. For instance, if we use TransE [20] and it generates the embeddings $(0.5, 0.2)$ for $NiedeGuidon$, $(0.25, 0.5)$

for ($archeologist$), and $(-0.3, -0.2)$ for the relation *occupation*, then we could expect to infer the triple $(NiedeGuidon, occupation, archeologist)$, as TransE gives high scores for a triple (s, p, o) if $||(\mathbf{s} + \mathbf{p}) - \mathbf{o}||_\ell \approx 0$, where $\mathbf{s}$, $\mathbf{p}$ and $\mathbf{o}$ are, respectively, the embeddings learned for s, p and o, and $||.||_\ell$ denotes the ℓ-norm.

Even today, this is the standard approach when designing KG embedding models, despite the variety of strategies: some rely on geometric-based mappings [20,103], others on expressing the graph as products of tensors [106], and more recent methods rely on neural networks [40].

This representation is shown to be not suitable for encoding rules, in particular, concept and role inclusions present in ontologies [53, 58, 118]. For example, we could have a concept inclusion that states that every mathematician is a scientist, but traditional embedding methods would not be able to use this information when learning embeddings, nor would guarantee that such constraint would be respected. They also do not take into account the time dimension, which may be central in some KGs that contain inconsistencies if we disregard the temporal annotations. Since there is already a fairly recent survey on KG embedding methods in general due to Dai et al. [36], we will focus on the approaches which propose embedding models which focus on compliance with ontological constraints or on representing the time dimension.

3.1.1 KG Embeddings and Ontological Constraints

There are different approaches to incorporate ontological constraints in KG embeddings. The goal is to improve the accuracy of these methods in KG completion tasks and also reduce the amount of data required to achieve a reasonable performance during training, particularly when the KG has sparse relations [46]. Moreover, compliance with ontological constraints might also lead to models that capture the semantics of the relations and entities included.

Fatemi, Ravanbakhsh and Poole [46] modify the SimplE embedding model so that it complies with additional constraints which express concept and role inclusions by enforcing inequality constraints on the score function of affected triples. More precisely, to express the role inclusion $r \sqsubseteq s$, the model enforces that the score function f is such that $\mathrm{f}(x, r, y) \leq \mathrm{f}(x, s, y)$ while also requiring every coordinate of every embedding to be non-negative. The strategy for concept inclusions is analogous, but involves transforming the pertinence relation to each class (i.e. the relation `rdf:type`) into a new relation. The resulting variant, named Simple$^+$ performs similarly to SimplE, and with faster convergence when taxonomic information is available. It also retains the same theoretical qualities of the original model.

In [37], the authors introduce two KG embedding models, TransOWL and TransROWL, which employs a *knowledge injection* approach. The two models proposed consist in adaptions of TransE [20] and TransR [77] in which the loss function is enriched with terms expressing ontological constraints expressing class and property hierarchies and equivalences, and inverse properties. The empirical evaluation shown reductions on the false positive rates, but also that there is still room for improvement.

BoxE [1] is a geometric KG embedding model designed to cope with patterns between relations such as symmetry, inversion and composition, as well as one-to-many and many-to-many relations. The ability to expresses those complex relationship patterns is a common short-coming in translation-based embedding models [1, 103], which can damper the benefits of their comparatively more explainable formal framework. In addition, BoxE employs knowledge injection to guarantee that the outcome always satisfies a rule. The authors prove that BoxE can be injected with rules in a language that combines the union, symmetry, hierarchy and intersection constraints. There were previous approaches to rule injection, but with more limited languages [41, 93].

There are also other approaches which rely directly on existing reasoners. Wiharja et al. [118] investigate iterative strategies for KG completion using a combination of methods that involves deductive reasoning (classical reasoners) and inductive reasoning (KG embeddings). In addition to showing that traditional KG embeddings do not comply with ontological constraints specified in OWL or in SHACL. In their iterative approach they employ an incomplete, but highly efficient reasoning procedure to detect triples that violate the constraints. The wrong triples are deleted, and the correct triples are added to the current KG, and the process is repeated until a fixpoint or other convergence condition is reached. ReasonKGE is a similar iterative approach proposed by Jain et al. [58] that relies on the use of deductive reasoning as an intermediate step. The main difference is that the wrong triples (that is, the ones that violate the ontological constraints) are not discarded, they are added to the set of negative samples for training the internal embedding model.

Gutiérrez-Basulto and Schockaert [53] introduce geometric models, where relations are mapped to convex regions, rather than vectors. This principled solution is suitable for a large fragment of the first-order Horn rule language, called quasi-chained. This language also corresponds to the chain-Datalog variant, which also places constraints on the negation (in particular, in combination with recursion) [109]. In this approach using convex regions [53], any triple predicted by that embedding model will not only be consistent with the quasi-chained ontology at hand, but it will also be a logical consequence of that same ontology. However, the geometric model based on convex-region, as introduced in the mentioned work, does not capture negation beyond the ability of expressing disjointness [86]. This is investigated in a new semantics for representing relations which allows full negation, called cone semantics [86]. In cone semantics, an interpretation maps each concept name to an axis-aligned, convex, cone (henceforth, simply cone). The authors then focus on the DL $\mathcal{ALC}$ and extend the semantics to all of its concept expressions (i.e. those built using the concept constructors in Table 1) so that those are also mapped to cones.

To conclude this discussion on KG embeddings and ontological constraints, we remark that there are still open questions regarding the ability of embedding models in capturing the semantics of entities and relations in a way that is logically consistent. The main argument for focussing on these questions remains to be able to verify that the embeddings capture connections between entities, concepts and relations. Therefore, there is still work to be done if we consider different languages and other forms of building defining interpretations using embeddings, as even the most principled approaches in this direction still have drawbacks [53, 58, 86].

3.1.2 KG Embeddings and Temporal Information

There are two prominent aspects in KGs that invoke time-awareness. First, as discussed earlier, KGs often include facts annotated with time validity [39]. For instance, the fact that Artur Ávila is member of the USNAS is only valid from 2019 onwards. The second aspect is the dynamic nature of KGs [60, 99]. KGs can be edited both by humans and programs, receiving constant updates, which also require constant repair. Hence, the KG itself changes with time. When performing reasoning, for instance via KG completion with embeddings, it might be crucial to take into account time validity and changes over time. These needs fostered KG embedding proposals that give a particular attention to temporal information. This is still a very recent and active area of research, and as such, we only briefly summarise some approaches in this field while presenting some of the characteristics that differentiate these methods from usual (atemporal) KG embedding strategies. We refer the reader to the survey due to Kazemi et al. [60] for a more detailed and thorough overview of temporal KG embedding approaches (and other time-aware Representation Learning methods for KGs).

Many approaches for temporal embeddings depart from atemporal KG embedding methods and modify the models to accommodate time [98] or devise frameworks in which traditional KG embedding methods are seen as components [120, 125]. Moreover, the strategies for creating this models often fall into the same three categories as traditional embedding models: geometric models [96, 115, 116, 122], tensor factorisation models [31, 50, 76, 98] and models based on neural networks [74]. HyTE [39], for example, was one of the first embedding models to encode time into the embeddings. HyTE represents the dynamic KG as a series of *snapshots*, each snapshot being a static KG at a specific point in time. In this model, translation-based embedding models, such as TransE, are modified with the specification of hyperplanes that indicate the time component of a fact. Wang and Li [116] employed the same hyperplane-based approach and extended TransD.

SEDE [127] also departs from TransE and uses the same snapshot-modelling intuition as HyTE. However the adaption for accommodating temporal information is inspired by strategies used in semantic word evolution. QCHyTE is another model based on HyTE which view the changes over time as gradual rather than abrupt, using a probabilistic approach [33]. Other methods that also model changes as gradual processes using probabilities include the frameworks due to Liao et al. [75] and ATiSE model [123]. TKGFrame [126] quantifies and models the temporal dependencies between relations over the same individual. TKGFrame is able to quantify, for example, that a transition from $bornIn$ to $marriedTo$ requires more "time" than the transition from $bornIn$ to $diedIn$ [126].

The existing models for temporal KG embeddings also differ regarding on whether they focus on events (individual time points), such as DE-SimplE [50], or if they are able to handle intervals, as is the case of TeRo [122]. Granularity of time intervals and precision of time points is also relevant to time-sensitive applications and it has been explored in the context of KG embeddings [70, 83, 124]. The TDG2E model [104], for instance, extends TransE with temporal information and uses a recurrent neural network to encode the dependencies between different snapshots and address issues related to the sparsity of temporal annotations. Li et al. [74] also use recurrent neural networks to represent temporal dependencies, but without using a geometric model underneath. The strategy relies primarily on an application of recurrence structures with graph convolutional networks [74]. The TIME2BOX model [25] is particularly expressive as it can deal with facts with unspecified time validity, open intervals and closed intervals (including left/right-open).

We note that some challenges that remain ubiquitous to KG embedding models in general are also being addressed in the temporal subfield. As usual, computational performance, in terms of resource usage, is an eminent concern [121], as well KG completion quality under adverse conditions, such as relations that occur in few facts in a KG [81]. Ensuring that the semantics of entities and relations is captured adequately (and ideally, with formal guarantees) is also a major concern and an active research problem in this field [78]. Models such as RETRA [117] and the approach due to Lie et al. [78] focus on contexts of facts defined by sharing of entities and relations.

Bourgaux, Ozaki and Pan [22]'s proposal also addresses the problem of capturing temporal aspects and ontological constraints. More precisely, the authors extended the geometric models as proposed by Gutiérrez-Basulto and Schockaert [53] from the classical DL setting to attributed DLs (as presented in Section 2.1.1). Then, they include the time dimension by enriching the attributed DL obtained with temporal operators. These operators allow a user to specify temporal dependencies such as "before", "until" and "during". Furthermore, the authors identify a fragments of the resulting language that ensure that a convex geometric model exists capturing its semantics.

4 Extracting Rules

Association Rule Mining (ARM) [4] and Formal Concept Analysis (FCA) [49] are two approaches for extracting rules from data. Initially, these techniques were applied for data structured in the format of databases. More recently, there has been increased interest in applying ARM and FCA for extracting rules from KGs [18, 19, 48, 52, 54, 69].

ARM is a practical and highly scalable approach for extracting rules from KGs. One of the most prominent tools to extract rules from KGs based on ARM is AMIE [48]. Current versions of this tool can extract rules from large KGs in a few minutes [69]. This approach uses two measures, called *support* and *confidence*, to guide the search of rules and to identify significant patterns to be extracted. In this context, the support of a rule is the number of true predictions of a rule in a KG. The authors of AMIE point out that for many real world KGs the support measure drastically decreases for the vast majority of the rules with more than 4 predicates in the body of the rule. Since low support for a rule means that the rule would only be applicable to few objects, this implies that, in most cases, long and difficult to interpret rules (with more than 4 predicates) can be discarded. Moreover, many combinations of 2 or 3 predicates already results in low suport.

▶ **Example 20.** Consider a rule with the binary predicate "teachesAt" in its body. The first argument of this predicate can only refer to educators while the second can only refer to educational institutions, which already significantly limits the possible combinations of predicates that can result in a rule with high support.

So the search space can be drastically reduced by applying the support measure to guide the search for significant patterns in the data. The main difficulty with the ARM approach is that, in order to achieve high scalability, the structure of the rules can be radically limited and that it does not provide theoretical guarantees regarding the extracted rules. In particular, this approach does not guarantee that all relevant rules in a certain rule language are extracted. It is also the case that the number of rules extracted can be very large (even though the size of each rule is limited).

The FCA approach addresses the difficulties of ARM, in particular, it gives theoretical guarantees for the rules extracted. The classical setting focuses on propositional Horn theory. In the DL context, there are e.g. works applying FCA to compute all concept inclusions of the $\mathcal{EL}$ ontology language (this language corresponds to the $\mathcal{ALC}$ description logic without negation, disjunction, and value restriction, see Table 1) that hold in a dataset [42, 52]. FCA also focus on succinctness, in particular, on constructing a set of rules that is minimal, called the *Duquenne-Guigues base* or *stem base*. The main challenges in applying FCA for extracting rules from KGs are (1) scalability, (2) the presence of a significant portion of erroneus information, and (3) the fact that rules generated with FCA can be complex and difficult to interpret. One way of dealing with challenges (1) and (2) is to combine FCA and ARM. A simple way of dealing with challenge (3) is by limiting the size of rules extracted, as it is the case in ARM.

5 Conclusion

In this work, we discussed different forms of inductive and deductive reasonings on Knowledge Graphs. Deductive and inductive reasoning have different strengths and shortcomings. Combining those to provide efficient and explainable reasoning is one of the current challenges in Artificial Intelligence. Besides deductive and inductive reasoning, there is also a third modality: abductive reasoning, which we briefly discuss next.

This third form of reasoning also aims at inferring facts, but from another perspective. Instead of focusing on new consequences that can be discovered, abductive reasoning concerns finding explanations for entailments already known or the lack of expected entailments. For example, suppose that we find out using inductive reasoning that the Wikidata KG implies that Artur Ávila is European. An abductive reasoning procedure could then be used to identify, among the numerous facts in Wikidata, which are strictly necessary to reach this conclusion. That is, we would like to find an explanation for that conclusion. Similarly, we might be interested in adding facts, axioms or rules to our KG or reasoning system to reach a particular conclusion without explicitly stating it.

Abduction in DLs, for instance, is a rich area of research with many different applications [13, 45, 64, 91]. One of them is debug and repair of DL ontologies via explanations and "explanation-like" objects, such as justifications [56], MinAs [9] and others [80]. The main goal is to find the explanations for undesired consequences and transform the ontology (or KG) so that it does not entail those anymore. Also, in the context of repair, we might be interested in finding the largest portions of the KG that do not entail a certain fact or present a particular behaviour. Such sets, and similar constructions, appear under many names in the Ontology Repair literature such as MaNAs [9], repairs [8], among others [80]. Some studies relate abductive reasoning and Datalog, either solving this problem for Datalog programs [24] or using Datalog to perform abduction in other languages. Additionally, as we mentioned in Section 2.3, there are also works on explanations for SHACL violations [5].

To conclude, we note that the approaches presented and mentioned here differ regarding the treatment of the KG and the reasoning problems they aim to solve. Understanding their advantages and disadvantages is key to devising better solutions for the existing challenges in reasoning with KGs.

References

1 Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/6dbbe6abe5f14af882ff977fc3f35501-Abstract.html.

2 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1994.

3 Serge Abiteboul and Victor Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43:62–124, 1991.

4 Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *Special Interest Group on Management Of Data SIGMOD*, 22(2):207–216, June 1993.

5 Shqiponja Ahmetaj, Robert David, Magdalena Ortiz, Axel Polleres, Bojken Shehu, and Mantas Šimkus. Reasoning about explanations for non-validation in SHACL. In *Proceedings of the Eighteenth International Conference on Principles of Knowledge Representation and Reasoning*. International Joint Conferences on Artificial Intelligence Organization, September 2021. doi:10.24963/kr.2021/2.

6 Medina Andresel, Julien Corman, Magdalena Ortiz, Juan L. Reutter, Ognjen Savkovic, and Mantas Simkus. Stable Model Semantics for Recursive SHACL. In *Proceedings of The Web Conference 2020*. ACM, April 2020. doi:10.1145/3366423.3380229.

7 Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

8 Franz Baader, Francesco Kriegel, Adrian Nuradiansyah, and Rafael Peñaloza. Making repairs in description logics more gentle. In Michael Thielscher, Francesca Toni, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, pages 319–328. AAAI Press, 2018. URL: `https://aaai.org/ocs/index.php/KR/KR18/paper/view/18056`.

9 Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. *J. Log. Comput.*, 20(1):5–34, 2010. `doi:10.1093/logcom/exn058`.

10 Luigi Bellomarini, Eleonora Laurenza, Emanuel Sallinger, and Evgeny Sherkhonov. Reasoning under uncertainty in knowledge graphs. In *Rules and Reasoning*, pages 131–139. Springer International Publishing, 2020. `doi:10.1007/978-3-030-57977-7_9`.

11 Luigi Bellomarini, Markus Nissl, and Emanuel Sallinger. Monotonic aggregation for temporal datalog. In Ahmet Soylu, Alireza Tamaddoni-Nezhad, Nikolay Nikolov, Ioan Toma, Anna Fensel, and Joost Vennekens, editors, *Proceedings of the 15th International Rule Challenge, 7th Industry Track, and 5th Doctoral Consortium @ RuleML+RR 2021 co-located with 17th Reasoning Web Summer School (RW 2021) and 13th DecisionCAMP 2021 as part of Declarative AI 2021, Leuven, Belgium (virtual due to Covid-19 pandemic), 8 - 15 September, 2021*, volume 2956 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL: `http://ceur-ws.org/Vol-2956/paper30.pdf`.

12 Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. The vadalog system. *Proceedings of the VLDB Endowment*, 11(9):975–987, May 2018. `doi:10.14778/3213880.3213888`.

13 Meghyn Bienvenu. Complexity of abduction in the EL family of lightweight description logics. In Gerhard Brewka and Jérôme Lang, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pages 220–230. AAAI Press, 2008. URL: `http://www.aaai.org/Library/KR/2008/kr08-022.php`.

14 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. In Richard Hull and Wenfei Fan, editors, *Proceedings of the 32nd Symposium on Principles of Database Systems (PODS'13)*, pages 213–224. ACM, 2013.

15 Bart Bogaerts and Maxime Jakubowski. Fixpoint Semantics for Recursive SHACL. *Electronic Proceedings in Theoretical Computer Science*, 345:41–47, September 2021. `doi:10.4204/eptcs.345.14`.

16 Bart Bogaerts, Maxime Jakubowski, and Jan Van den Bussche. SHACL: A Description Logic in Disguise. In *Proceedings of the 33rd Benelux Conference on Artificial Intelligence and the 30th Belgian Dutch Conference on Machine Learning (BNAIC/BENELEARN 2021)*, August 2021. `arXiv:2108.06096`.

17 Iovka Boneva, José Emilio Labra Gayo, and Eric G. Prud'hommeaux. Semantics and validation of shapes schemas for RDF. In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 104–120. Springer, 2017. `doi:10.1007/978-3-319-68288-4_7`.

18 Daniel Borchmann. *Learning terminological knowledge with high confidence from erroneous data.* PhD thesis, Higher School of Economics, 2014.

19 Daniel Borchmann and Felix Distel. Mining of $\mathcal{EL}$-GCIs. In *The 11th IEEE International Conference on Data Mining Workshops*, Vancouver, Canada, 2011.

20 Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

21 Camille Bourgaux and Ana Ozaki. Querying attributed dl-lite ontologies using provenance semirings. In *AAAI*, pages 2719–2726. AAAI Press, 2019.

22 Camille Bourgaux, Ana Ozaki, and Jeff Z. Pan. Geometric models for (temporally) attributed description logics. In Martin Homola, Vladislav Ryzhikov, and Renate A. Schmidt, editors, *Proceedings of the 34th International Workshop on Description Logics (DL 2021) part of Bratislava Knowledge September (BAKS 2021), Bratislava, Slovakia, September 19th to 22nd, 2021*, volume 2954 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2954/paper-7.pdf.

23 Sebastian Brandt, Elem Güzel Kalayci, Roman Kontchakov, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Ontology-based data access with a horn fragment of metric temporal logic. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1070–1076. AAAI Press, 2017. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14881.

24 F. Buccafurri, N. Leone, and P. Rullo. Enhancing Disjunctive Datalog by constraints. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):845–860, 2000. doi:10.1109/69.877512.

25 Ling Cai, Krzysztof Janowicz, Bo Yan, Rui Zhu, and Gengchen Mai. Time in a Box. In *Proceedings of the 11th on Knowledge Capture Conference*. ACM, December 2021. doi:10.1145/3460210.3493566.

26 Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. Datalog±. In *Proceedings of the 12th International Conference on Database Theory - ICDT '09*. ACM Press, 2009. doi:10.1145/1514894.1514897.

27 Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics*, 14:57–83, July 2012. doi:10.1016/j.websem.2012.03.001.

28 Andrea Calì, Georg Gottlob, and Andreas Pieris. Query answering under non-guarded rules in Datalog+/-. In Pascal Hitzler and Thomas Lukasiewicz, editors, *Proceedings of the 4th International Conference on Web Reasoning and Rule Systems (RR 2010)*, volume 6333 of *LNCS*, pages 1–17. Springer, 2010.

29 David Carral, Irina Dragoste, Larry González, Ceriel Jacobs, Markus Krötzsch, and Jacopo Urbani. VLog: A rule engine for knowledge graphs. In *Lecture Notes in Computer Science*, pages 19–35. Springer International Publishing, 2019. doi:10.1007/978-3-030-30796-7_2.

30 S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, March 1989. doi:10.1109/69.43410.

31 Melisachew Wudage Chekol. Tensor decomposition for link prediction in temporal knowledge graphs. In *Proceedings of the 11th on Knowledge Capture Conference*. ACM, December 2021. doi:10.1145/3460210.3493558.

32 Melisachew Wudage Chekol, Giuseppe Pirrò, Joerg Schoenfisch, and Heiner Stuckenschmidt. Marrying uncertainty and time in knowledge graphs. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 88–94. AAAI Press, 2017. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14730.

33 Shuo Chen, Lin Qiao, Biqi Liu, Jue Bo, Yuanning Cui, and Jing Li. Knowledge Graph Embedding Based on Hyperplane and Quantitative Credibility. In *Machine Learning and Intelligent Communications*, pages 583–594. Springer International Publishing, 2019. doi:10.1007/978-3-030-32388-2_50.

34 Jan Chomicki and Tomasz Imielinski. Temporal deductive databases and infinite objects. In Chris Edmondson-Yurkanan and Mihalis Yannakakis, editors, *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 21-23, 1988, Austin, Texas, USA*, pages 61–73. ACM, 1988. doi:10.1145/308386.308416.

35 Julien Corman, Juan L. Reutter, and Ognjen Savković. Semantics and validation of recursive SHACL. In *Lecture Notes in Computer Science*, pages 318–336. Springer International Publishing, 2018. doi:10.1007/978-3-030-00671-6_19.

36 Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5):750, May 2020. doi:10.3390/electronics9050750.

37 Claudia d'Amato, Nicola Flavio Quatraro, and Nicola Fanizzi. Injecting Background Knowledge into Embedding Models for Predictive Tasks on Knowledge Graphs. In *The Semantic Web*, pages 441–457. Springer International Publishing, 2021. doi:10.1007/978-3-030-77385-4_26.

38 Ariyam Das, Sahil M. Gandhi, and Carlo Zaniolo. ASTRO: A datalog system for advanced stream reasoning. In Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang, editors, *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1863–1866. ACM, 2018. doi:10.1145/3269206.3269223.

39 Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2001–2011. Association for Computational Linguistics, 2018. doi:10.18653/v1/d18-1225.

40 Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press, 2018. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366.

41 Boyang Ding, Quan Wang, Bin Wang, and Li Guo. Improving knowledge graph embedding using simple constraints. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 110–121. Association for Computational Linguistics, 2018. doi:10.18653/v1/P18-1011.

42 Felix Distel. *Learning description logic knowledge bases from data using methods from formal concept analysis.* PhD thesis, Dresden University of Technology, 2011.

43 Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: Integrating datalog and description logics. *Journal of Intelligent and Cooperative Information Systems*, 10(3):227–252, 1998.

44 Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–418, September 1997. doi:10.1145/261124.261126.

45 Corinna Elsenbroich, Oliver Kutz, and Ulrike Sattler. A case for abductive reasoning over ontologies. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006. URL: http://ceur-ws.org/Vol-216/submission_25.pdf.

46 Bahare Fatemi, Siamak Ravanbakhsh, and David Poole. Improved knowledge graph embedding using background taxonomic information. In *AAAI*, pages 3526–3533. AAAI Press, 2019.

47 Mónica Figuera, Philipp D. Rohde, and Maria-Esther Vidal. Trav-SHACL: Efficiently validating networks of SHACL constraints. In *Proceedings of the Web Conference 2021*. ACM, April 2021. doi:10.1145/3442381.3449877.

48 Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730, 2015.

49 Bernhard Ganter and Rudolph Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1997.

50 Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic Embedding for Temporal Knowledge Graph Completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3988–3995, April 2020. doi:10.1609/aaai.v34i04.5815.

51 Georg Gottlob, Erich Grädel, and Helmut Veith. Datalog LITE: a deductive query language with linear time model checking. *ACM Trans. Comput. Log.*, 3(1):42–79, 2002. doi:10.1145/504077.504079.

52 Ricardo Guimarães, Ana Ozaki, Cosimo Persia, and Baris Sertkaya. Mining EL bases with adaptable role depth. In *AAAI*, pages 6367–6374. AAAI Press, 2021.

53 Víctor Gutiérrez-Basulto and Steven Schockaert. From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules. In *Proceedings of KR*, 2018.

54 Tom Hanika, Maximilian Marx, and Gerd Stumme. Discovering implicational knowledge in wikidata. In Diana Cristea, Florence Le Ber, and Baris Sertkaya, editors, *ICFCA*, volume 11511 of *Lecture Notes in Computer Science*, pages 315–323. Springer, 2019.

55 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *ACM Comput. Surv.*, 54(4):71:1–71:37, 2021.

56 Matthew Horridge. *Justification based explanation in ontologies*. PhD thesis, University of Manchester, 2011.

57 Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin N. Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language*. W3C Member Submission, 21 May 2004. Available at http://www.w3.org/Submission/SWRL/.

58 Nitisha Jain, Trung-Kien Tran, Mohamed H. Gad-Elrab, and Daria Stepanova. Improving Knowledge Graph Embeddings with Ontological Reasoning. In Andreas Hotho, Eva Blomqvist, Stefan Dietze, Achille Fokoue, Ying Ding, Payam M. Barnaghi, Armin Haller, Mauro Dragoni, and Harith Alani, editors, *ISWC*, volume 12922 of *Lecture Notes in Computer Science*, pages 410–426. Springer, 2021.

59 Yevgeny Kazakov, Markus Krötzsch, and František Simančík. ELK reasoner: Architecture and evaluation. In Ian Horrocks, Mikalai Yatskevich, and Ernesto Jimenez-Ruiz, editors, *Proceedings of the OWL Reasoner Evaluation Workshop 2012 (ORE'12)*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

60 Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 21:70:1–70:73, 2020. URL: http://jmlr.org/papers/v21/19-447.html.

61 Mayank Kejriwal, Craig A. Knoblock, and Pedro Szekely. *Knowledge Graphs*. The MIT Press, March 2021. URL: https://www.ebook.de/de/product/39993807/mayank_kejriwal_craig_a_knoblock_pedro_szekely_knowledge_graphs.html.

62 Holger Knublauch and Dimitris Kontokostas, editors. *Shapes Constraint Language (SHACL)*. W3C Recommendation, 20 July 2017. Available at http://www.w3.org/TR/shacl/.

63 Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 2656–2661. AAAI Press/IJCAI, 2011.

64 Patrick Koopmann, Warren Del-Pinto, Sophie Tourret, and Renate A. Schmidt. Signature-based abduction for expressive description logics. In Diego Calvanese, Esra Erdem, and Michael Thielscher, editors, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pages 592–602, 2020. doi:10.24963/kr.2020/59.

65 Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Veronika Thost. Attributed description logics: Reasoning on knowledge graphs. In Jérôme Lang, editor, *IJCAI*, pages 5309–5313. ijcai.org, 2018.

66 Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description logic rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*, pages 80–84. IOS Press, 2008.

67 Markus Krötzsch, Sebastian Rudolph, and Peter H. Schmitt. On the semantic relationship between datalog and description logics. In Pascal Hitzler and Thomas Lukasiewicz, editors, *Proceedings of the 4th International Conference on Web Reasoning and Rule Systems (RR 2010)*, volume 6333 of *LNCS*, pages 88–102. Springer, 2010.

68 Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Veronika Thost. Attributed description logics: Ontologies for knowledge graphs. In *Lecture Notes in Computer Science*, pages 418–435. Springer International Publishing, 2017. doi:10.1007/978-3-319-68288-4_25.

69 Jonathan Lajus, Luis Galárraga, and Fabian M. Suchanek. Fast and exact rule mining with AMIE 3. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *ESWC*, volume 12123 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2020.

70 Julien Leblay, Melisachew Wudage Chekol, and Xin Liu. Towards Temporal Knowledge Graph Embeddings with Arbitrary Time Precision. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. ACM, October 2020. doi:10.1145/3340531.3412028.

71 Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6:167–195, 2015. doi:10.3233/SW-140134.

72 Martin Leinberger, Philipp Seifer, Tjitze Rienstra, Ralf Lämmel, and Steffen Staab. Deciding SHACL shape containment through description logics reasoning. In *Lecture Notes in Computer Science*, pages 366–383. Springer International Publishing, 2020. doi:10.1007/978-3-030-62419-4_21.

73 Nicola Leone, Carlo Allocca, Mario Alviano, Francesco Calimeri, Cristina Civili, Roberta Costabile, Alessio Fiorentino, Davide Fuscà, Stefano Germano, Giovanni Laboccetta, Bernardo Cuteri, Marco Manna, Simona Perri, Kristian Reale, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. Enhancing DLV for large-scale reasoning. In *Logic Programming and Nonmonotonic Reasoning*, pages 312–325. Springer International Publishing, 2019. doi:10.1007/978-3-030-20528-7_23.

74 Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal Knowledge Graph Reasoning Based on Evolutional Representation Learning. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 408–417. ACM, 2021. doi:10.1145/3404835.3462963.

75 Siyuan Liao, Shangsong Liang, Zaiqiao Meng, and Qiang Zhang. Learning Dynamic Embeddings for Temporal Knowledge Graphs. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. ACM, March 2021. doi:10.1145/3437963.3441741.

76 Lifan Lin and Kun She. Tensor decomposition-based temporal knowledge graph embedding. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, November 2020. doi:10.1109/ictai50040.2020.00151.

77 Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187. AAAI Press, 2015. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

78 Yu Liu, Wen Hua, Jianfeng Qu, Kexuan Xin, and Xiaofang Zhou. Temporal knowledge completion with context-aware embeddings. *World Wide Web*, 24(2):675–695, March 2021. doi:10.1007/s11280-021-00867-6.

79 Frank Manola and Eric Miller, editors. *Resource Description Framework (RDF): Primer.* W3C Recommendation, 10 February 2004. Available at http://www.w3.org/TR/rdf-primer/.

80 Vinícius Bitencourt Matos, Ricardo Ferreira Guimarães, Yuri David Santos, and Renata Wassermann. Pseudo-contractions as gentle repairs. In Carsten Lutz, Uli Sattler, Cesare Tinelli, Anni-Yasmin Turhan, and Frank Wolter, editors, *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, volume 11560 of *Lecture Notes in Computer Science*, pages 385–403. Springer, 2019. doi:10.1007/978-3-030-22102-7_18.

81 Mehrnoosh Mirtaheri, Mohammad Rostami, Xiang Ren, Fred Morstatter, and Aram Galstyan. One-shot learning for temporal knowledge graphs. In Danqi Chen, Jonathan Berant, Andrew McCallum, and Sameer Singh, editors, *3rd Conference on Automated Knowledge Base Construction, AKBC 2021, Virtual, October 4-8, 2021*, 2021. doi:10.24432/C55K56.

82 Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. RDFox: A highly-scalable RDF store. In *The Semantic Web - ISWC 2015*, pages 3–20. Springer International Publishing, 2015. doi:10.1007/978-3-319-25010-6_1.

83 Runyu Ni, Zhonggui Ma, Kaihang Yu, and Xiaohan Xu. Specific Time Embedding for Temporal Knowledge Graph Completion. In *2020 IEEE 19th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*. IEEE, September 2020. doi:10.1109/iccicc50026.2020.9450214.

84 W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview.* W3C Recommendation, 27 october 2009. Available at http://www.w3.org/TR/owl2-overview/.

85 Ana Ozaki, Markus Krötzsch, and Sebastian Rudolph. Temporally attributed description logics. In Carsten Lutz, Uli Sattler, Cesare Tinelli, Anni-Yasmin Turhan, and Frank Wolter, editors, *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, volume 11560 of *Lecture Notes in Computer Science*, pages 441–474. Springer, 2019.

86 Özgür Lütfü Özçep, Mena Leemhuis, and Diedrich Wolter. Cone semantics for logics with negation. In Christian Bessiere, editor, *IJCAI*, pages 1820–1826. ijcai.org, 2020.

87 Paolo Pareti, George Konstantinidis, and Fabio Mogavero. Satisfiability and containment of recursive SHACL. *CoRR*, abs/2108.13063, 2021. arXiv:2108.13063.

88 Paolo Pareti, George Konstantinidis, Fabio Mogavero, and Timothy J. Norman. SHACL satisfiability and containment. In *Lecture Notes in Computer Science*, pages 474–493. Springer International Publishing, 2020. doi:10.1007/978-3-030-62419-4_27.

89 Bijan Parsia, Nicolas Matentzoglu, Rafael S. Gonçalves, Birte Glimm, and Andreas Steigmiller. The OWL reasoner evaluation (ORE) 2015 competition report. *Journal of Automated Reasoning*, 59(4):455–482, February 2017. doi:10.1007/s10817-017-9406-8.

90 Eric Prud'hommeaux and Andy Seaborne, editors. *SPARQL Query Language for RDF.* W3C Recommendation, 15 January 2008. Available at http://www.w3.org/TR/rdf-sparql-query/.

91 Júlia Pukancová and Martin Homola. Abductive Reasoning with Description Logics: Use Case in Medical Diagnosis. In Diego Calvanese and Boris Konev, editors, *Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7-10, 2015*, volume 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015. URL: http://ceur-ws.org/Vol-1350/paper-60.pdf.

92 Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, January 2006. doi:10.1007/s10994-006-5833-1.

93 Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar, editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1119–1129. The Association for Computational Linguistics, 2015. doi:10.3115/v1/n15-1118.

94 Riccardo Rosati. $\mathcal{DL}$+*log*: A tight integration of description logics and disjunctive datalog. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 68–78. AAAI Press, 2006.

95 Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Description logic reasoning with decision diagrams: Compiling $\mathcal{SHIQ}$ to disjunctive datalog. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *Proceedings of the 7th International Semantic Web Conference (ISWC'08)*, volume 5318 of *LNCS*, pages 435–450. Springer, 2008.

96 Ali Sadeghian, Mohammadreza Armandpour, Anthony Colas, and Daisy Zhe Wang. ChronoR: Rotation Based Temporal Knowledge Graph Embedding. 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*, pages 6471–6479. AAAI Press, 2021. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/16802`.

97 Floriano Scioscia, Michele Ruta, Ivano Bilenchi, Filippo Gramegna, Eugenio Di Sciascio, and Davide Loconte. Owl reasoner evaluation results, 2021. `doi:10.5281/ZENODO.5013799`.

98 Pengpeng Shao, Dawei Zhang, Guohua Yang, Jianhua Tao, Feihu Che, and Tong Liu. Tucker decomposition-based temporal knowledge graph completion. *Knowledge-Based Systems*, 238:107841, February 2022. `doi:10.1016/j.knosys.2021.107841`.

99 Umang Sharan and Jennifer Neville. Temporal-relational classifiers for prediction in evolving domains. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, December 2008. `doi:10.1109/icdm.2008.125`.

100 Rob Shearer, Boris Motik, and Ian Horrocks. Hermit: A Highly-Efficient OWL Reasoner. In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *OWLED*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

101 Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

102 Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System description. *Journal of Web Semantics*, 27-28:78–85, August 2014. `doi:10.1016/j.websem.2014.06.003`.

103 Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: `https://openreview.net/forum?id=HkgEQnRqYQ`.

104 Xiaoli Tang, Rui Yuan, Qianyu Li, Tengyun Wang, Haizhi Yang, Yundong Cai, and Hengjie Song. Timespan-Aware Dynamic Knowledge Graph Embedding by Incorporating Temporal Evolution. *IEEE Access*, 8:6849–6860, 2020. `doi:10.1109/access.2020.2964028`.

105 Edward Thomas, Jeff Z. Pan, and Yuan Ren. TrOWL: Tractable OWL 2 reasoning infrastructure. In *Lecture Notes in Computer Science*, pages 431–435. Springer Berlin Heidelberg, 2010. `doi:10.1007/978-3-642-13489-0_38`.

106 Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016. URL: `http://proceedings.mlr.press/v48/trouillon16.html`.

107 Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *LNCS*, pages 292–297. Springer, 2006.

108 Jeffrey Ullman. *Principles of database and knowledge-base systems*. Computer Science Press, Rockville, Md, 1988.

109 Jeffrey D. Ullman and Allen Van Gelder. Parallel complexity of logical query programs. *Algorithmica*, 3(1-4):5–42, November 1988. `doi:10.1007/bf01762108`.

110 Jacopo Urbani, Ceriel Jacobs, and Markus Krötzsch. Column-oriented datalog materialization for large knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 2016. To appear.

111 Boris Villazón-Terrazas, Nuria García-Santa, Yuan Ren, Alessandro Faraotti, Honghan Wu, Yuting Zhao, Guido Vetere, and Jeff Z. Pan. Knowledge graph foundations. In Jeff Z. Pan, Guido Vetere, José Manuél Gómez-Pérez, and Honghan Wu, editors, *Exploiting Linked Data and Knowledge Graphs in Large Organisations*, pages 17–55. Springer, 2017.

112 Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10), 2014.

113 Przemyslaw Andrzej Walega, Mark Kaminski, and Bernardo Cuenca Grau. Reasoning over streaming data in metric temporal datalog. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3092–3099. AAAI Press, 2019. `doi:10.1609/aaai.v33i01.33013092`.

114 Dingmin Wang, Pan Hu, Przemyslaw Andrzej Walega, and Bernardo Cuenca Grau. Meteor: Practical reasoning in datalog with metric temporal operators. *CoRR*, abs/2201.04596, 2022. `arXiv:2201.04596`.

115 Jingbin Wang, Wang Zhang, Xinyuan Chen, Jing Lei, and Xiaolian Lai. 3DRTE: 3D Rotation Embedding in Temporal Knowledge Graph. *IEEE Access*, 8:207515–207523, 2020. `doi:10.1109/access.2020.3036897`.

116 Zhihao Wang and Xin Li. Hybrid-TE: Hybrid Translation-Based Temporal Knowledge Graph Embedding. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, November 2019. `doi:10.1109/ictai.2019.00205`.

117 Simon Werner, Achim Rettinger, Lavdim Halilaj, and Jürgen Lüttin. RETRA: Recurrent transformers for learning temporally contextualized knowledge graph embeddings. In *The Semantic Web*, pages 425–440. Springer International Publishing, 2021. `doi:10.1007/978-3-030-77385-4_25`.

118 Kemas Wiharja, Jeff Z. Pan, Martin J. Kollingbaum, and Yu Deng. Schema aware iterative knowledge graph completion. *Journal of Web Semantics*, 65:100616, December 2020. `doi:10.1016/j.websem.2020.100616`.

119 Honghan Wu, Ronald Denaux, Panos Alexopoulos, Yuan Ren, and Jeff Z. Pan. Understanding knowledge graphs. In Jeff Z. Pan, Guido Vetere, José Manuél Gómez-Pérez, and Honghan Wu, editors, *Exploiting Linked Data and Knowledge Graphs in Large Organisations*, pages 147–180. Springer, 2017.

120 Jiapeng Wu, Yishi Xu, Yingxue Zhang, Chen Ma, Mark Coates, and Jackie Chi Kit Cheung. TIE: A framework for embedding-based incremental temporal knowledge graph completion. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 428–437. ACM, 2021. `doi:10.1145/3404835.3462961`.

121 Tianxing Wu, Arijit Khan, Huan Gao, and Cheng Li. Efficiently embedding dynamic knowledge graphs. *CoRR*, abs/1910.06708, 2019. `arXiv:1910.06708`.

122 Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. TeRo: A time-aware knowledge graph embedding via temporal rotation. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, 2020. `doi:10.18653/v1/2020.coling-main.139`.

123 Chenjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi, and Jens Lehmann. Temporal Knowledge Graph Completion Based on Time Series Gaussian Embedding. In *Lecture Notes in Computer Science*, pages 654–671. Springer International Publishing, 2020. `doi:10.1007/978-3-030-62419-4_37`.

124 Yonghui Xu, Shengjie Sun, Huiguo Zhang, Chang'an Yi, Yuan Miao, Dong Yang, Xiaonan Meng, Yi Hu, Ke Wang, Huaqing Min, Hengjie Song, and Chuanyan Miao. Time-Aware Graph Embedding: A Temporal Smoothness and Task-Oriented Approach. *ACM Transactions on Knowledge Discovery from Data*, 16(3):1–23, June 2022. `doi:10.1145/3480243`.

125 Youri Xu, Haihong E, Meina Song, Wenyu Song, Xiaodong Lv, Haotian Wang, and Jinrui Yang. RTFE: A Recursive Temporal Fact Embedding Framework for Temporal Knowledge Graph Completion. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5671–5681. Association for Computational Linguistics, 2021. `doi:10.18653/v1/2021.naacl-main.451`.

126 Jiasheng Zhang, Yongpan Sheng, Zheng Wang, and Jie Shao. TKGFrame: A Two-Phase Framework for Temporal-Aware Knowledge Graph Completion. In *Web and Big Data*, pages 196–211. Springer International Publishing, 2020. `doi:10.1007/978-3-030-60259-8_16`.

127 Yujing Zhou, Jia Peng, Lei Wang, Daren Zha, and Nan Mu. SEDE: semantic evolution-based dynamic knowledge graph embedding. *Aust. J. Intell. Inf. Process. Syst.*, 16(4):64–73, 2019. URL: `http://ajiips.com.au/papers/V16.4/v16n4_68-77.pdf`.