information

MDPI

Article

LPG-Based Knowledge Graphs: A Survey, a Proposal and Current Trends

Davide Di Pierro [1,†], Stefano Ferilli [1,*,†] and Domenico Redavid [2,†]

[1] Department of Computer Science, University of Bari Aldo Moro, 70125 Bari, Italy
[2] Department of Economic and Finance, University of Bari Aldo Moro, 70124 Bari, Italy
* Correspondence: stefano.ferilli@uniba.it; Tel.: +39-080-544-2293
† These authors contributed equally to this work.

Abstract: A significant part of the current research in the field of Artificial Intelligence is devoted to knowledge bases. New techniques and methodologies are emerging every day for the storage, maintenance and reasoning over knowledge bases. Recently, the most common way of representing knowledge bases is by means of graph structures. More specifically, according to the Semantic Web perspective, many knowledge sources are in the form of a graph adopting the Resource Description Framework model. At the same time, graphs have also started to gain momentum as a model for databases. Graph DBMSs, such as Neo4j, adopt the Labeled Property Graph model. Many works tried to merge these two perspectives. In this paper, we will overview different proposals aimed at combining these two aspects, especially focusing on possibility for them to add reasoning capabilities. In doing this, we will show current trends, issues and possible solutions. In this context, we will describe our proposal and its novelties with respect to the current state of the art, highlighting its current status, potential, the methodology, and our prospect.

Keywords: knowledge graphs; labeled property graphs; ontologies; graph db; automated reasoning

Citation: Di Pierro, D.; Ferilli, S.; Redavid, D. LPG-Based Knowledge Graphs: A Survey, a Proposal and Current Trends. *Information* **2023**, *14*, 154. https://doi.org/10.3390/info14030154

Academic Editor: Ryutaro Ichise

Received: 17 December 2022
Revised: 15 February 2023
Accepted: 20 February 2023
Published: 1 March 2023

1. Introduction

In 1956 at Dartmouth College, Hanover, four researchers (John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude Shannon) conducted the Dartmouth Summer Research Project on Artificial Intelligence (AI). It was the first time the term "Artificial Intelligence" was used. It denotes a behavior of a machine that, should a human behave in the same way, would be considered intelligent [1]. It is difficult to extend this definition because the factors describing human intelligence are still unclear and undefined. One alternate definition does exist that captures the nature of the work being carried out in the field. This was proposed by Rich, and reads as follows: "Artificial Intelligence is the study of making computers do things which, at the moment, humans do better" [2].

Decades have passed but today's definitions do not depart significantly from that perspective. According to Dick, the aim of research on AI is "to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" [3]. Artificial Intelligence is a general term that implies the use of a computer to model intelligent behavior with minimal human intervention [4]. Since, according to Newell and Simon [5], a software agent is "intelligent" if and only if it is knowledge-based, the representation and handling of knowledge are one of the most studied problems in AI. Inspired by human problem solving, this gave rise to the branch of Knowledge Representation and Reasoning (KRR), aimed at representing knowledge for intelligent systems so as to endow them with the ability to solve complex tasks [5,6].

Structured repositories of knowledge are called knowledge bases (KBs). A KB is a centralized repository of information. A library, an archive or a database about a particular subject are all examples of KBs. In the field of AI, they are used as a source for the

distribution and retrieval of information in a human-like format, amenable to manipulation by symbolic AI approaches. They are also a component of knowledge management systems. One of their possible applications is for Question Answering (QA). The purpose of a QA system is to find the correct answers to arbitrary user questions in both non-structured and structured collections of data [7]. In order to make it more applicable to real-world scenarios, researchers have shifted their attention from simple questions to complex questions, which require constraint inference [8].

Specifically, when KBs are organized according to a graph structure, they are also referred to as knowledgegraphs(KGs). I.e., a KG can be viewed as a graph when considering its graph structure [9]. When it involves formal semantics, it can be taken as a KB for interpretation and inference over facts [10]. KGs that represent structural relations between entities have become an increasingly popular research direction toward cognition and human-level intelligence [11].

KGs are used to store large amounts of knowledge. Hence, they need a "conceptualization" that delimits and defines what they can express and how. The problem of formalizing what is present in the world (or in a domain) has its roots in philosophy but has many uses in the AI field, where the conceptualization of what exists in a particular domain or scenario of interest is called an ontology. Ontology as a branch of philosophy is the science of what exists, of the kinds and structures of objects, properties, events, processes, and relations in every subset of reality. In AI, an ontology is a formal, explicit specification of a shared conceptualization [12].

More recently, graphs have also been exploited as a model for building databases (DBs). In graph databases, data manipulation is expressed by graph-oriented operations and type constructors [13]. Compared to traditional DBs, graph DBs are more suitable to support applications, in which instance-based processing is needed, as opposed to batch processing. This is a typical need of AI, as opposed to traditional software, (e.g., for accounting). This is why we are interested in the possibility of combining graph DB technologies, ensuring efficiency in data handling, with KRR technologies, allowing advanced knowledge manipulation and exploitation (e.g., reasoning). Unfortunately, the graph models traditionally adopted in the literature for these two perspectives are different and partly incompatible, which raised the need for an investigation into how they can be combined.

In the next sections, we will provide some background notions on these topics, and then we will discuss state-of-the-art approaches and their limitations, before delving into a proposal for overcoming some of the issues related to the problem of merging the two, different perspectives of graph DBs and KRR.

2. Preliminaries

A graph is defined as a set of "vertexes", some of which are connected by "edges" [14]. The vertexes may represent concepts or instances. Concepts refer to general categories of objects, such as "person", "place", or "organization". Instances are specific objects belonging to concepts, such as a specific person (e.g., Micheal Jordan), a specific location (e.g., New York) or a specific organization (e.g., IBM). The edges represent binary relationships between concepts or instances.

Two main approaches to graphs are of interest for the purposes of this paper: those based on the Resource Description Framework (RDF) [15] model, adopted in KRR research, and those based on the Labeled Property Graph (LPG) [16] model, adopted in DB practice.

- The former model provides for graphs made up of "atomic" nodes and arcs only. Usually, their structure is represented as a set of triples (also called "statements") of the form $(subject, predicate, object)$ where the subject is a node identifier, the predicate is an arc label and the object can be either another node identifier or a literal. In many cases, it may be necessary to express additional information on a triple as a whole. RDF provides a pre-defined vocabulary for describing RDF statements. A description of a statement using this vocabulary is called a "reification" of the statement. The RDF

reification vocabulary consists of the type rdf:Statement, and the properties rdf:subject, rdf:predicate, and rdf:object [15].

- LPG graphs still comprise nodes and arcs, but both have their own identifiers and can be provided with labels expressing their type and with properties that describe their features in the form of $(key, value)$ pairs.

The LPG model looks promising for supporting general graph data processing, but its performance has not yet been studied enough [17]. However, it is adopted by Neo4j, the most used graph DB, which also owes its success to the high performances it ensures. Automatic sharding, built-in cache and replication are just some of the main benefits [18]. The spreading of Neo4j increased also the popularity of the LPG model. The adoption of RDF, on the other hand, has been boosted by current emerging technologies. The adoption of Linked Data technologies has shifted the Web from a space for connecting documents to a global space where pieces of data from different domains are semantically linked and integrated to create a global Web of Data [19]. The versatility of the graph models has been massively used in other technologies such as the Semantic Web.

2.1. Graphs and Databases

Since their conception, graphs have been extensively used to represent complex relations among entities [20]. Graphs are strongly supported by modern DataBase Management Systems (DBMSs). The transition from traditional (relational) DBs to NoSQL ones [21] set a milestone in the history of DBs, since it introduced many advantages, better performance and scalability above all. Graph DBs are an example of NoSQL databases. Using a graph-based abstraction of knowledge has a number of additional benefits compared to a relational model or to other NoSQL alternatives [22]. Many companies have developed in-house implementations in order to cope with the need for graph DB systems. Neo4j is the most outstanding graph DB currently available. We will now describe the main characteristics and some relevant uses of graph DBs in different fields.

Graph DBs are outstanding for application in areas where information about data interconnectivity or topology is more important, or at least as important as, the data itself [13]. In their first computer applications, they have mostly been used to represent social networks and interactions among people. Nowadays, the importance of graph data is not limited to social networks but spans biology (e.g., to model gene regulation) and other types of networks in different domains. Graph DBs have seen extensive application in the field of bioinformatics. Bioinformatics uses graph DBs to relate a complex web of information that includes genes, proteins and enzymes. A prime example is the Bio4j project [23], a framework powered by Neo4j that performs protein-related querying and management [24]. Another major application of graph DBs is for Recommender Systems, aimed at advising users about relevant products and information by predicting their interests based on various types of information [25]. Graph DBs have also been used for transactions OLTP systems [26]: these are designed for transaction integrity and operational availability. Promising techniques are now emerging for applying Machine Learning over graphs.

The term 'data model' has been widely used in the information management community, where it covers various meanings [13]. In the most general sense, a data(base) model is a collection of conceptual tools used to model representations of real-world entities and the relationships among them [27]. A DB model consists of three components:

- a collection of data structure types (the building blocks of any DB that conforms to the model);
- a collection of operators or inference rules, which can be applied to any valid instance of the data types, to retrieve or derive data from any part of those structures in any combination desired;
- a collection of general integrity rules, which implicitly or explicitly define the set of consistent DB states, or changes of state, or both—these rules may sometimes be expressed as insert-update-delete rules [28].

An efficient graph DB model is necessary for better management of graphs. 'Industrial' graph DBs implement the Property Graph (PG) data model [26] but other models are available when moving to other technologies such as those of the Semantic Web. In the property graph data model, the graph structure's elements can have some user-defined attributes. With the rise of big data, there has been a huge demand to design data models and tools. These data models should be capable of handling a variety of data structures. Analysis of graph properties is deeply studied by the Data Mining community [29]. Graphs have an important advantage: they can keep all the information about an entity at a single node and show related information by arcs connected to that node [30].

Even though Graph DBs are very flexible, their main drawback is that they are usually not consistent, since they have very limited tools to ensure consistency [31]. Actually, for the purposes of browsing the data, in many cases, it may be convenient to ignore the schema [32] since graph DBs may provide implementations of special graph storage structures, and efficient graph algorithms for realizing specific operations [33]. Graph DB systems follow CRUD (create, read, update, delete) methods that are used in a graph data model [34]. Graph DBs store the data in nodes and arcs, not in tables. Hence, no join operations are allowed.

DBs in general are queried by means of a Query Language, i.e., a collection of operators or inference rules that can be applied to any valid instance of the types of data structures provided by the model, with the aim of manipulating and querying the data in those structures in any combination desired [35]. Associated with graphs are specific graph operations in the query language algebra, such as finding shortest paths, determining certain subgraphs, and so forth [33]. Queries can refer directly to the graph structure. Full knowledge of the structure is not necessary to express meaningful queries [36]. Based on the technology, graphs can be represented according to different formalisms and the modelling phase must follow the structure accordingly. Relational DBMS systems use Structured Query Language (SQL) for inserting, updating and deleting data and/or schemes. Cypher, a well-established language for querying and updating property graph DBs, born with the Neo4j product, is a declarative query language for property graphs. Cypher provides capabilities for both querying and modifying the data, as well as for specifying schema definitions [37].

2.2. Knowledge Graphs

Recently, knowledge graphs, as a form of structured representation of human knowledge, have attracted a great deal of attention from academia and industry [22,38–40]. KGs use a graph-based data model to capture knowledge in application scenarios that involve integrating, managing, and extracting value from, diverse sources of data on a large scale. The essential elements involved in the notion of a KG can be traced to ancient history in the core idea of representing knowledge in diagrammatic form. Examples include Aristotle and visual forms of reasoning, around 350 BC; Lull and his tree of knowledge; Linnaeus and taxonomies of the natural world; and in the XIX century, the works on formal and diagrammatic reasoning of scientists such as J.J. Sylvester, C. Peirce and G. Frege. These ideas also involve several disciplines such as mathematics, philosophy, linguistics, library sciences, and psychology, among others [41].

The interest in KGs is due to their human-understandable structures that facilitate comprehension and understanding in many AI applications. In particular, the KG depicts an integrated collection of real-world entities which are connected by semantic relations. In this respect, data are provided in a formal language via data annotation and manipulation in a machine-readable format, thereby reducing ambiguity and deriving meaningful information that is specific to an application domain [42]. In this perspective, in order to deal with this extraordinary growth in the available data, data analytics and mining tools for KGs emerged.

Leveraging KGs, fragmented or partially observed entities and concepts can be connected to form a complete and structured knowledge repository, facilitating the management, retrieval, usage and understanding of the information it contains [43].

Many AI fields took advantage of the expressiveness of KGs. Among them, Semantic Web (SW), KRR, Natural Language Processing (NLP) and Machine Learning (ML). In turn, this gave KGs the opportunity to borrow ideas also from other domains.

KGs are crucial to many enterprises today: they provide the structured data and factual knowledge that drive many products and make them more intelligent and "magical" [44].

In many critical applications, KGs are empowered with schemes or ontologies so as to infer new knowledge and provide a shared vocabulary with respect to what already exists in other domains.

Recent years have witnessed a rapid growth in KG construction and application [40]. A large number of KGs, such as Freebase [45], DBpedia [46], YAGO [47], and NELL [48] have been created and successfully applied to many real-world applications, from semantic parsing [49,50], to named entity disambiguation [51,52], information extraction [53,54], and question answering [55].

Automated reasoning is a form of simulated thinking, and a process of inferring new knowledge (conclusions) from existing one (premises). Many forms of reasoning leverage KGs. New relations among entities can be derived through reasoning and can be fed back to enrich the KGs, and then support the advanced applications [56]. Reasoning over graphs may also be exploited for error detection.

Initially, studies were carried out in the fields of logic and knowledge engineering. The logic experts used to formalise the complexity of a domain (or application area) in languages based on First-Order Logic (i.e., predicate logic). The goal was to infer correct conclusions starting from a knowledge base consisting of facts and rules. To make up for the rigidity of this approach, which was a limitation in many contexts, techniques such as non-monotonic and fuzzy reasoning [57] were adopted, capable of grasping the uncertainty and the fuzziness intrinsically present in the real world. Differently from the first researchers, knowledge engineers came up with new formalisms based on semantic networks, able to represent richer concepts, relationships and constraints among entities and attributes. These nets also provide a visual representation, particularly appreciated for the interpretability of the knowledge they aimed to describe. Yet, they turned out to be limited by the massive increase in available knowledge. Before the development of automatic knowledge acquisition methods, knowledge (in the form of entities, relationships and attributes) was entirely handcrafted by experts in the fields. With the massive growth of information, traditional methods based on artificially built knowledge bases have been overcome by data-driven approaches.

2.3. Semantic Web

Thanks to their versatility, effectiveness and performance, KGs have been embraced by the Semantic Web initiative. The Semantic Web is usually envisioned as an enhancement of the current World Wide Web with machine-understandable information as described in [58], which arguably marks the birth of the field. The traditional architecture is enriched with logic-based semantics that admits reasoning over the meaning of the data [59]. In Tim-Berners Lee's vision of the WWW, pieces of information are not only to be read by humans, but also by machines, so as to enable automatic information processing.

The SW leverages ontologies as a way of performing reasoning tasks such as instance checking, consistency checking and so on. Ontological knowledge bases enable formal querying and reasoning and, consequently, a main research focus has been the investigation of how deductive reasoning can be exploited in ontological representations to support more advanced applications [60].

2.3.1. Ontologies

An ontology seeks to provide a definitive and exhaustive classification of entities in any sphere of being. The classification should be definitive in the sense that it can serve as an answer to such questions as what classes of entities are needed for a complete description and explanation of all the goings-on in the universe? It should be exhaustive in the sense that all types of entities should be included in the classification, including also the types of relations by which entities are tied together to form larger wholes [61]. The term "ontology" has gained prominence in recent years in the field of computer and information science [62]. Designing ontologies has the goal to solve the so-called "Tower of Babel" problem [63]. Each ontology, database and source of knowledge has its own definition, terms and concepts and it is expected to find the same exact meaning expressed by different words in different sources. Historically, this problem used to be tackled by specific name mappings. However, the need for a shared conceptualization and namings quickly arose leading to the huge development of ontologies for many varieties of domains. The Semantic Web embraced this perspective and defined a language to describe what exists in the WWW through the Web Ontology Language (OWL).

2.3.2. Web Ontology Language

The Web Ontology Language is an ontology language for the Semantic Web with a formally defined meaning. OWL ontologies provide classes, properties, individuals, and data values, and are stored as Semantic Web documents. OWL ontologies can be used along with information written in RDF, and OWL ontologies themselves are primarily exchanged as RDF documents [64]. A key feature of OWL, and thus of SW in general, is the possibility of implementing inference engines. Several approaches for the implementation of inference engines were proposed in the early years after the publication of the OWL specification. In particular, in [65] the features that an inference engine for OWL must fulfill ontological consistency checking, entailment computation, query processing, reasoning with rules, and handling of XML data types-were formulated. Apart from reasoning with rules, for which research is ongoing, these features have been implemented in reasoners for SW based on Description Logics [66].

2.3.3. Technologies for the Semantic Web

A fundamental objective for the SW development was interoperability. Notable among W3C standardization efforts are XML (eXtensible Markup Language)/XML schema and RDF/RDF schema, which facilitates semantic interoperability [67].

XML is a standard for structuring documents, notably for the World Wide Web. An XML document is made up of a tree of (possibly nested) tags with different attributes [68]. In XML, Document Type Definitions (DTDs) [69] allow adding constraints about how tags can be combined and the possible attributes at the different levels. They specify a grammar that limits all possible combinations when writing tags, attributes and so on.

Just as XML schema provides a vocabulary-definition facility, RDF schema lets developers define a particular vocabulary for RDF data (such as authorOf) and specify the kinds of objects to which these attributes can be applied. In other words, the RDF schema mechanism provides a basic type system for RDF models. This type of system uses some predefined terms, such as Class, subPropertyOf, and subClassOf, for the application-specific schema. RDF schema expressions are also valid RDF expressions [67].

The RDF data model [70] uses Uniform Resource Identifiers (URIs) to unambiguously identify different resources. One could make single statements about each element of a collection; however, if the aim is to make a statement about all the different instances of a concept (where the individual members might change), a container must be used [71]. Remind that RDF is a graph and, as for LPG, the fetching is provided by navigation. RDF graphs can be queried using the SparQL [72] language. SparQL is a language that lets users query RDF graphs by specifying "templates" against which to compare the graph components. Data which matches ("satisfies") a template is returned from the query. The

SparQL query engine will return an exhaustive list of the subject component of triples that satisfies the query through value substitution [73].

SparQL is also implemented in frameworks for the management of SW resources, such as Apache Jena. It provides an interface for specifying queries and getting the result streams in a more readable form for manipulation. Recently, SPARQL has been extended for querying real-time data coming from RDF data streams. Generally, the SPARQL query is performed over the streams using a sliding temporal window and static knowledge data. Performance analysis was conducted using data in the field of smart cities, and Virtuoso RDF [74] turned out to be the best platform for query execution both in static and dynamic context [75].

2.4. Comparison and Differences between LPG and RDF

Even though the two models are used in completely different scenarios, a comparison for underlying some characteristics is necessary to understand the reasons why they have become state-of-the-art in modern AI approaches. In this work by Baken et al. [76], a comparison among many graph models was performed, mostly focusing on comparing query performances.

In Table 1, we summarized the main characteristics of the two graph models.

Table 1. A comparison between the two graph models.

Graph Model	Directed Edges (i)	Labels (ii)	Attributes (iii)	URI (iv)	Reasoning (v)
LPG	✓	✓	✓		
RDF	✓	✓		✓	✓

As we can see from the table, the labels and directions of relationships are allowed in both models. When describing relationships, the concept of direction is fundamental to understanding what is the subject of the relationship and the object (or literal). Directed edges (i) [77] are always represented as arrows and queries in Cypher/SPARQL take directions into account. In general, labels (ii) are human-readable properties of a node. Even if the concept has the same name, some distinctions must be specified when talking about the two graph models. Standard RDF Primer [15] defines rdfs:label as an instance of rdf:Property that may be used to provide a human-readable version of a name of a resource. In LPG, each node or relationship has an ID tag and one or more "labels" describing its type or class [78]. The difference between the use of labels in the two models is that labels in RDF are URIs that identify resources, while LPG labels provide freedom in their choice and are no different than any other property. Attributes (iii) in LPG are in the form of key-value pairs. Values are literals (string, integer, float, ...). Attributes are not present in RDF graphs. Attributes in RDF are so reified. With reification, we mean to create a node having as a label the type of the value and create a relationship (probably named as the name of the attribute) from the node to the literal. Many classification works rely on graph attributes [79]. In the LPG model, they are allowed in relationships as well. RDF lacks attributes and that is the main structural difference. URI (iv) [80] is the mechanism SW uses to uniquely identify resources on the Web. No similar approach exists in graph databases where the DBMS identifies resources by means of internal ids. This means that we have no guarantees at all that resources are not duplicated. Finally, reasoning (v) is the objective we aim for. RDF is structurally suitable to be treated with SW reasoners for inferencing knowledge. Differently from RDF, LPG is inherently unsuitable for reasoning purposes, even though some standard ML algorithms such as Page Rank are implemented in some queries.

The mechanism of reification in RDF easily scales when dealing with a higher order of abstraction. Imagine representing the sentence "Mark says that John passed the exam". The triple 'John'-'passes'-'exam' is not enough. There is no way to link this triple to the other part of the sentence. With reification, we can create a unique resource for the triple itself, which is actually just assigning it a URI. Given that, the triple 'Mark'-'says'-['John'-'passes'-'exam'] is now possible. The impossibility of assigning URIs is one of the main downsides of the LPG model. Basically, LPG graphs were not thought to store public (web) data, so there was no need of providing unique identifiers. On the other hand, if you think about the above-mentioned example, it can be solved by creating a node and by putting properties representing the subject of the internal statement ('John'), the predicate ('passes') and the object ('exam'). You could simply imagine how cumbersome this process becomes when the levels of abstraction increase. Regarding weights, there are many applications which need them on arcs, or even on nodes. We can list just some of the most common algorithms used on the Web such as Page Rank [81] or HITS [82].

There is also plenty of recent works in which weights represent an estimation of the real case scenario [83–85]. Weights are not supported by any of the two models. Thanks to attributes, we are able to weight links by adding a specific "weight" attribute to which a numeric value is added. This property has been largely exploited [86,87]. Yet this approach is strongly application-dependent. You may also consider that handling weights in RDF becomes much heavier from a computational point of view because each triple should be reified and a relationship must be created to store the weight. Navigation will involve a much greater number of nodes. Finally, LPG graphs are just a different way to store data. They are thought in order to perform the CRUD operations on data. RDF, on the other hand, goes far beyond. Invented in the context of the Semantic Web, it is developed to be exploited by an SW reasoner in order to infer new knowledge. On the other hand, graph DBMS, such as Neo4j, provides higher performances for navigating data. The best we can perform is to create a bridge between the two models to have the highest efficiency and the capability of reasoning, combining the benefits of both.

Hence, we can state that RDF, differently from LPG, and in combination with OWL, is suitable for the creation of Knowledge Graphs.

3. Knowledge Graphs Applications

Many works make use of the OWL formalism to express and reason with knowledge graphs. Yet depending on the context, some personalization is applied by academics or companies so as to fit their need. The main problem is that any personalization goes far away from the standard and the lack of interoperability among different solutions becomes evident. In this section, we are going to list different relevant applications in which KGs have been adopted. We are going to list works that have a very relevant application from the social point of view or the ones that realised a relevant novel approach.

Asamoah et al. [88] examined using KG as the basis in the design of a knowledge representation system that drives the filtration process of a company's cyber security ecosystem in cloud computing. They defined the entity relations of the cyber security threat entity hierarchy. According to them, OWL/XML format has no extension for supporting custom relations but rather emulates custom relations with complex object and data properties. The algorithm scans the different rules in order to detect possible threats.

Recent developments of KGs have been employed in the field of education for the design of Intelligent Tutoring Systems (ITSs) [89]. Graphs allow the possibility of rapidly moving along different objects, subjects and disciplines thanks to the insight (perhaps unexpected) connections among different concepts.

Among the most common scenarios, we can find the management of cultural heritage. Carriero et al. [90] worked on ArCO, the Italian Cultural Heritage knowledge graph. It collects and validates the catalogue records of (ideally) all Italian Cultural Heritage properties (excluding libraries and archives). ArCO is made up of:

- A knowledge graph consisting of:
  - a network of ontologies modeling the Cultural Heritage domain;
  - a Linked Open Dataset of nearly 169M triples.
- a software for automatically converting catalogue records;
- a detailed documentation reporting;
- a set of running examples that potential consumers can use as training material. They consist of natural language CQs and their corresponding SPARQL queries, which can be directly tested against ArCo's SPARQL endpoint;
- a test suite, implemented as OWL files and SPARQL queries, used for validating ArCo knowledge graph. It provides a real-case implementation of an ontology testing methodology, useful to both students, teachers, researchers, and practitioners;
- a SPARQL endpoint to explore the resource, run tests, etc.

Farazi et al. [91] demonstrated through examples how the concept of a Semantic Web-based knowledge graph can be used to integrate combustion modeling into cross-disciplinary applications and in particular how inconsistency issues in chemical mechanisms can be addressed.

When trying to assemble a mechanism by combining collections of species and reactions from multiple sources, one encounters two well-known classes of consistency problems. The first one relates to a unique identification. Additionally, vice versa, species that ought to be distinct may have been given identical labels in different mechanisms. The second problem relates to data inconsistency: the same species or reaction from different sources may have been assigned different thermodynamic or kinetic parameter values, with variations at times well beyond the reported uncertainties. For these reasons, they provided ontologies to be used to solve conflicts.

The healthcare system is recently experiencing a strong increase in the form of data expressed through Textual Medical Knowledge (TMK). Shi et al. [92] explored a model to organize and integrate the TMK into conceptual graphs. They then employed a framework to automatically retrieve knowledge in knowledge graphs with high precision. Their model consists of three parts: Medical Knowledge Model, Health Data Model (HDM), and Terminology Glossary. Medical Knowledge Model is used to organize the TMK into conceptual graphs. Health Data Model is used to define the detailed structures and relationships of the unstructured health data. Terminology Glossary provides a metathesaurus to express the instances of both TMK and HDM and provides semantic mappings to achieve integration. After the construction procedure, they were able to utilize the interconnections among medical terms to perform chain inference rules to explore the complex semantics between entities. First-order predicate logic is used to perform reasoning. Inferences are proceeded by forward chaining and back chaining over the knowledge graph.

Fathalla et al. [93] had the vision that ultimately researchers will work on a common knowledge base comprising comprehensive descriptions of their research, thus making research contributions transparent and comparable. Due to the representation of review articles as unstructured text, it is impossible to automatically extract and analyze information from them. In the paper, they introduced the concepts, terms and vocabularies that they defined for representing the content of review articles. The overall workflow of the study comprises four steps: article selection, formalization, ontology development, and querying the ontology to demonstrate its potential usage.

Tomic et al. [94] presented their experience with using the representation and query standards and tools of the Semantic Web to encode and manipulate the dairy farming domain knowledge in a form of the Dairy Farming Ontology (DFO).

Rossetto et al. [95] presented the first iteration of LifeGraph, a knowledge graph for lifelogging data. With the increase in both capability and availability of mobile computation and sensing technologies, the means for capturing a growing fraction of the human experience become increasingly available. The data produced by efforts such as life logging is commonly multi modal and can have manifold interrelations with external information.

Bader et al. [96] proposed a structured dataset in the form of a semantically annotated knowledge graph for Industry 4.0-related standards, norms and reference frameworks. The graph provides a Linked Data-conform collection of annotated, classified reference guidelines supporting newcomers and experts alike in understanding how to implement Industry 4.0 systems.

The objective of the work by Szekely et al. [97] is to create generic technology to enable the rapid construction of knowledge graphs for specific domains together with query, visualization and analysis capabilities that enable end-users to solve complex problems. The challenge is to exploit all available sources, including web pages, document collections, databases, delimited text files, structured data such as XML or JSON, images, and videos. This paper describes the technologies and their application to build a large knowledge graph for the human trafficking domain.

Energy and sustainability are hot topics today and the amount of related research is increasing every day. An eco-industrial park (EIP) aims for industrial symbiosis that promises improved energy and resource efficiency as well as reduced environmental impact. Numerous studies have been carried out focusing on resource networks within a single domain such as water [98–100], energy [101–103], and material [104–106]. Zhou et al. [107] implemented the J-Park Simulator (JPS), a cross-domain knowledge graph for the process industry, which includes ontologies in domains such as chemical process engineering, chemical kinetics, internal combustion engines, etc. Ontologies play pivotal roles in the JPS project. Ontologies from different domains offer a formal definition of classes and relations in a certain field; the JPS project has been developing and integrating ontologies systematically. The JPS graph could be distributed across the Web.

After this short overview of the main domains of application, we are going to list in Table 2 the number of works to which KGs have been applied, displaying the number of works between 2019 and 2022 and the number of works between 2015 to 2018 so that a small naive comparison can be carried out. Papers indexed by Google are taken into account.

Table 2. Number of KG applications per domain from 2019 to 2022, and from 2015 to 2018.

Domain	Number of Papers (2019–2022)	Number of Papers (2015–2018)
Cybersecurity	~4400	~900
Culture Heritage	~3600	~2000
Biology	~17,600	~15,300
Healthcare	~17,500	~17,400
Industry	~17,500	~18,000
Smart City	~9000	~3000
Medicine	~18,000	~16,800

As you can easily spot, there has been a relevant increase in the Smart City and Cybersecurity demands. In the Smart City domain, the interrelation among services naturally fosters the use of graphs, given the heterogeneity of available data. In a broader sense, citizens and tourists, as well as service providers, build the knowledge base. Heterogeneity includes also diversity, that is the possibility of linking several resources for a wider social objective. On the other hand, in the Cybersecurity field, recent developments [108–110] exploit graph models for interpretability reasons as well as to capture long-distance relationships among features, promoting the development of interpretable solutions that are considerably more preferred in contexts such as law and in decision processing tasks. However, the number of works for other domains is also noteworthy.

4. LPG Graph

Graph databases provide better support for highly interconnected datasets than relational databases. However, labeled property graph databases are schema optional, making them prone to data corruption, especially when new users switch from relational databases

to graph databases [111]. In this section, we are going to briefly describe the multitude of domains of application and some of the main Machine Learning tasks associated with their application. Graph databases have become increasingly popular as they provide better support for highly interconnected datasets [112,113]. Highly interconnected datasets are found in most big data applications [114,115] including social networks, bioinformatics and astronomy. Such data can be more easily expressed using the nodes and edges of a graph database than the table-based structure offered by relational databases [111].

Many Artificial Intelligence tasks rely on graphs, and clustering analysis is one of them. It is a crucial component of unsupervised learning tasks with applications ranging from pattern recognition to biology, suitable for images and gene technologies. Wang et al. [116] proposed a new clustering algorithm based on the properties of data.

Kalva et al. [117] provided the entire idea of how to retrieve the user's hidden interests and what is a semantic network. The hidden interests of an individual can be inferred by keenly observing their social profile data and blending this data with a semantic network. Getting user interests without the user's manual intervention is very beneficial for companies feeding on users' regular behavior.

From the first decade of this century, the problem of learning from data given by labeled graphs attracted much attention in Machine Learning and Data Mining communities [118–126]. Kuznetsov et al. [127] used an approach based on the generation of closed sets of labeled graphs and their approximations. An important application for learning with labeled graphs is the analysis of the properties of chemical substances.

Life sciences and mathematics are usually considered quite distant areas of research. Yet there are close relationships among them, especially with the increase in computational power. Formanowicz et al. [128] provided a short review of selected applications of labeled graphs in biology and chemistry. Graph theory problems concerning molecules of chemical compounds and DNA sequencing are examples of applications in which graph theory plays an especially important role.

5. Merging LPG and RDF

In the literature, although in completely heterogeneous application domains, there have been several works that have tried to use the SW and graph databases simultaneously or that have created an intermediate layer to make one converge with the other.

5.1. Merging Sources

Historically, this need has existed for some time. There are several works, even not too distant, that would have liked a method for connecting graph databases with RDF, including [129]. In this work, the huge amount of data available in traditional databases is mentioned, but mainly, the authors had to work with the integration of data already present in RDF format. The approach described, however, for integrating URIs coming from different sources is relevant and provides interesting insights in cases where similar techniques can be reused.

Interest also spread the NLP area. In [130], they proposed a method to build a knowledge graph from a well-known lexical database: WordNet [131]. In particular, their aim was to build a new KB that could be used for reasoning operations of various natures, in their case for Text Entailment Recognition. In particular, the graph construction followed different steps: synsets sample selection, automatic pre-processing, data curation, classifier training, database classification, data post-processing, and RDF conversion [130].

In the field of microbiology, Pency et al. [132] present OpenBiodiv: an infrastructure that combines semantic publishing workflows, text and data mining, common standards, ontology modelling and graph database technologies for managing biodiversity knowledge. It is presented as a Linked Open Dataset generated from the scientific literature. It is a good example of an overall infrastructure dedicated to the integration.

5.2. Structure Mapping

In [133], a framework for the projection and validation of a portion of a graph in RDF using the graph database formalism is described. Specifically, the framework allows a SparQL query to be written, translated into JSON and parsed as to be interpreted by the LPG model. In the end, the initial query is displayed in the new formalism. It is a useful tool for information verification and consistency checking with reference ontologies.

There are also attempts [134,135] to integrate SW techniques into relational databases using the fixed properties of the relations. Entities are seen as classes, keys as relationships and properties as attributes.

In [136], Angles et al. studied the RDF model from a database perspective. A lot of effort has been put into understanding the efficiency of languages for querying graphs. For instance, in [137], a variant of SparQL, G-SPARQL, was proposed, which has its own grammar and uses relational databases as a source base for finding data. Some improvements have been developed such as combining different types of queries into one and the separation of concerns between model and instances storage.

5.3. Language Mappings

In [138], even a new language is proposed. The goal is to introduce languages that work directly over triples and are closed, without using graph models. They also extend their language with recursion with good results when compared with other languages. In [139], a new language is proposed, Gremlinator, which extends Gremlin [140] and is able to navigate graph databases from SparQL queries. On this side, there are numerous attempts to enrich languages to increase the functionality and power of queries.

5.4. New Graph Models

Some approaches take one perspective and bring it to the other part. These approaches are more sophisticated since the translation from one technology into the other one is very complex and an open issue.

The most frequent is the translation of RDF data into graph databases. By adding an intermediate layer, SparQL queries are translated into Cypher and data can be retrieved as they are written directly in RDF. A proposal is described here [141], where the mapping translation follows an algorithm based on the structure of the nodes and an ontology, which acts as a reference for creating labels on each node.

In [142], is proposed the system Yet Another RDF Serialization (YARS), which allows preparing RDF data to exchange on the property graph data stores. Their serialization is textual. It has three different parts:

- prefix directives: a part where prefixes are defined.
- vertex declarations: parts where vertices are created.
- relationship declarations: parts where edges and properties are created.

Iordanov [143] even proposes a different graph database model, named HyperGraphDB, based on generalized hypergraphs where hyperedges can contain other hyperedges. This generalization automatically reifies every entity expressed in the database thus removing many of the usual difficulties in dealing with higher-order relationships. For our purposes, we can reuse some general ideas of how to treat higher levels of relationships with reification, even if the final purpose is to build a general system able to collect huge sources of (heterogeneous) graphs, in order to construct a common structure.

In [144], the authors consider the problem of supporting PGs as RDF in the Oracle Database. They introduce a PG to RDF transformation scheme. They propose three models:

- Namedgraph based. This proposal involves the use of quads (as opposed to triples) to create a unique named graph IRI for each edge. Then the label and the key/value properties of the edge are associated with the graph IRI.
- Subproperty based. Id, label, and key/values for an edge can be modelled by creating a unique RDF property for each edge to represent the edge id, creating an RDF triple

with that property as the predicate, associating the key/value pair with that property, and then making the property a subproperty of another property created based on the edge label.

- *Reification based.* In order to accommodate the id, label, and key/value pairs for an edge, reification in RDF can create a new resource to represent every reified RDF statement.

5.5. Interoperability

In [145], RDF triples are integrated into PGs in order to be exploited by some graph engines. There is an intermediate layer: G2GML, which follows an algorithm for bringing triples into the database. The drawbacks are the same as the similar approaches described above. There exist more sophisticated algorithms based on G2GML, such as [146], who implement a better algorithm for the management of URIs, and [147], who implement mapping rules for schemes and instances. Tomaszuk et al. [148] present an ontology-based approach to transform (automatically) PGs into RDF graphs. The ontology, called PGO, defines a set of terms that allows describing the elements of a PG. The algorithm corresponding to the transformation method is described, and some properties of the method are discussed (complexity, data preservation, and monotonicity). The strength of this solution lies in the fact that it is not domain-dependent, and that the solution is generic and applicable in different contexts with acceptable performance.

Schatzle et al. [149] propose a mapping that is native to GraphX (a parallel processing system implemented on Apache Spark). The proposed graph model is an extension of the regular graph but lacks the concept of attributes. The mapping uses a special attribute label to store the node and edge identifiers, i.e., each triple t = (s, p, o) is represented using two vertices.

It should also be pointed out that one almost always ends up with data that are not complete. There are robust works that are well suited to these issues and have translation algorithms that are effective even in the most critical circumstances. An example of this is [150], in which some issues such as empty labels are handled.

There are also attempts to carry neither formalism to the other side but to use an intermediate storage medium. This is the case of [151], which presents an effective unified relational storage scheme, that can seamlessly accommodate both RDF and PGs. Furthermore, it has been implemented the storage schema on an open-source graph database to verify its effectiveness.

It is evident that RDF and graph database systems are tightly connected as they are based on graph-oriented database models. On one hand, RDF database systems (or triplestores) are based on the RDF data model, their standard query language is SparQL, and there are languages to describe structure, restrictions and semantics on RDF data (e.g., RDF Schema, OWL, SHACL, and ShEx). On the other hand, most graph database systems are based on the PG data model, there is no standard query language and the notions of graph schema and integrity constraints are limited [152].

Given these differences both in terms of structures and uses, it is relevant and necessary to cope with the interoperability of the two. For instance, how data can be moved, how information can be shared and merged, and so on. Studying syntactic interoperability is not enough but it would represent a good start for this process.

RDF data can be encoded in different formats such as Turtle, NTRIPLE, RDF/XML, RDF/JSON and JSON-LD. In contrast, there is no data format to encode PGs. Yet none of them is able to cover all the features presented by the PG data model.

A recent extension of RDF that aims to bridge the gap between the two data models is called RDF* which is already supported by several RDF systems. Indeed, as a foundation of such conversions, several authors have introduced direct mappings from LPGs to RDF* [153].

Hartig et al. [154] propose two transformations from RDF* to PGs:

- Mapping any RDF triple as an edge in the resulting PG. Each node has the "kind" attribute to describe the type of a node (e.g., IRI).
- Distinguishing data and object properties. The former is transformed into node properties and the latter into edges of a PG.

A common approach to support semantic interoperability is the definition of data and schema transformation methods. The schema transformation method takes as input the schema of the source database and generates a schema for the target database. Similarly, the data transformation method allows moving the data from the source database to the target database but taking care of the target schema. The transformation methods can be implemented by using data formats or data definition languages [152].

In [155], there is a proposal to combine the benefits into a single graph abstraction layer called Semantic Property Graph (SPG). The SPG layer sits on top of the RDF and simulates the property graph model. They describe the SPG model and its queries, which are SW-compliant, to be executed inside property graph databases. This middle layer is able to interpret SparQL queries. This work consists of the developing of a graph model as an abstraction graph layer on top of the RDF singleton property that can simulate the two distinct characteristics of the PG model and a graph query pattern that can express the PG traversals to the key-value properties of the nodes and edges.

In [156], there is a big survey of the different techniques for storing RDF data and the peculiarities we can exploit from each of them. As you can imagine, triple stores fit better with NoSQL models and are inherently thought to be used as graphs. Hence, we want to stress the fact that we are just proposing the most common idea of integrating RDF with graph databases but future ongoings of this work can also spread to other models.

Other works, such as [157], describes how to move data from OpenAPI Specification to RDF. Structures of OpenAPI Specifications fit very well for this task since tabular forms are much easier to be translated in forms of triples. This is seen as a very natural and standard approach. As usual, URI generation is relevant also in this case.

Different mappings have been proposed. Nguyen et al. [155] introduce an approach to convert arbitrary LPGs into a specific class of LPGs that, then, can be mapped directly into RDF. The idea of this approach is to transform every edge into a vertex which gets connected to the two vertices incident to the transformed edge by adding two new edges, labeled "in" and "out". Notice that these new edges are the only types of edges after this transformation step, and there are no more edge properties (they have become vertex properties). The first phase covers the vertices and is equivalent to the first phase of the algorithm described before. The second phase starts by mapping each edge into its corresponding RDF triple; then, for each property defined for the edge, the algorithm adds a nested triple containing the triple representing the edge as a quoted triple in the subject, and the property and value mappings in the predicate and object positions.

The work that best matches our needs and that we can work on to achieve our goal is GraphBRAIN [158], an existing system that manages ontologies, arranges data in a graph database and has minimal cues for integrating tools from SW. This work will be our starting point, and everything we will design and develop will be integrated into this platform. We will devote Section 6 to the description of our proposal.

5.6. Open Issues

We are going to mention here the main limitations of the above-mentioned state-of-the-art approaches. Some of them may be mitigated but, in general, they are open issues.

In [129], the way data is integrated is probably too tied with the biological field, in the sense that it does not provide general strategies to integrate data but exploits biological insights.

Penev et al. [132] lack in the process of integrating (and disambiguating) knowledge coming from many sources and, differently from them, we do not want to take the huge amount of knowledge expressed in RDF and bring them into a graph database.

Purohit et al. [133] provide an interesting approach for data validation but it is not really usable for connecting the two perspectives of the graphs.

In [141] the authors used graph database technology and developed support to make this technology suitable also for RDF triples. The overall result is not satisfactory for our purposes for different reasons:

- there is a loss of data at the end of the process, since, in a graph database, we do not have a link among other objects in the Web.
- for large KBs, there is a need for strong computational and memory power.
- after having translated RDF triples into the graph, we cannot perform any other operations but visualizing data.
- there is no way to capture potential inconsistencies after the data is inside the graph.
- transferring a huge amount of RDF triples into LPG graphs requires a lot of computational time. This is one of the most underrated problems.

The main limitation of [143] is that this approach does not use the tools of the SW at all. Hence, also the queries and operations possible with HyperGraphDB are not the same.

Das et al. [144] lacks in the reasoning part. The resulting graph can only be queried in SparQL.

In [148] what, from our point of view, can be improved is the possibility of performing translations from one model to another also taking ontologies into account in the translation process.

The shortcoming of [152] is that RDF* is not supported by the majority of RDF triplestores and requires changes from RDF data beforehand. Interoperability between sources happens when both are able to understand the meaning of the data to be exchanged. For this purpose, both data and instances must be appropriately treated and transformed.

In [155], the authors have not analyzed the problems related to control over the information that you can store in the graph database and the management of the corresponding semantic descriptions.

To the best of our knowledge, there is no method that supports data and schema (i.e., class and property axioms) transformations between RDF and PGs preserving all the advantages of both. Based on our literature review about RDF and PGs interoperability, we identified the most relevant issues and challenges:

- There is no single graph model. LPG is just one of the most used models.
- RDF is based on triples while PG contains attributes and there are different ways to move from one structure to another, none of them prevailed among the others.
- Refication cannot be fully automatized. The SW provides some mechanism to represent attributes on triples (e.g., reified statements) but different contexts can privilege other solutions.
- RDF reification leads to a considerable increase in the size of the resulting graph.

In [159], the authors propose a novel approach called Singleton Property for representing statements about statements and providing formal semantics for it. They also demonstrate the use of singleton property in the representation and querying of meta knowledge giving a satisfying performance in terms of the number of triples, query length and query execution time compared to existing approaches.

6. Proposal

Due to unsatisfactory results (in terms of generality), research is going on into this topic and we would also like to contribute with a proposal for the unification of the two perspectives, proposing also mechanisms for enriching the types of reasoning available with graphs. Our proposal, named *GraphBRAIN*, uses Neo4j as Graph DB, thanks to the many features described above. Graph DB technologies are not inherently purposed for representing schemes. In our mind, schemes are also intended as ontologies. This is the reason why we leverage GraphBRAIN, which is prone to provide intelligent reasoning operations as well as traditional CRUD database operations.

It is compliant with SW formalism, since it embodies mechanisms to import SW data and map data into the SW-compliant formalism (OWL).

6.1. General Concept and Applications

To the best of our knowledge, state-of-the-art approaches do not have the possibility of exploiting the two main characteristics of the two graph models: the effectiveness of RDF and the efficiency of LPG. This is due to the fact that whenever we move from one perspective to another we lose advantages. For this reason, we apply the separation of concerns. In our methodology, we are going to use LPG as a graph model for storing instances while schemes are kept separated in order to be used as traditional database schemes for managing CRUD operations. Additionally, we can exploit SW reasoning by mapping data from LPG to RDF, and enrich the knowledge base with ontologies. Ontologies are expressed through XML, and we will explain later the motivations of this choice. While instances are mapped into the RDF formalism, ontologies are mapped into OWL. This allows merging schemes and instances, and reasoning becomes achievable. Apart from SW reasoning, other reasoning approaches can be considered when dealing with KBs. For this reason, we aim to translate schemes (in XML) and instances (in the graph) into a first-order logic language, e.g., Prolog. Consequently, a plethora of techniques can be applied, covered in what we call "multistrategy reasoning". We will discuss this part in Section 7. The overall platform is GraphBRAIN, which provides an interface for the creation and management of ontologies. This general system aims to move in different contexts. As an example, schemes can be moved in the logic programming field following some rule-mapping criteria. Furthermore, the GraphBRAIN application can apply consistency-checking mechanisms to verify the integrity of schemes with instances. Further details of GraphBRAIN are available in [158]. Our proposal is meant to be used in every context in which intelligent information retrieval and/or reasoning are considered relevant. This proposal is not intended for those interested in the mere storage of data. GraphBRAIN will allow importing of resources and reasoning capabilities on them. At this moment, we cannot provide substantial numerical results or metrics of our proposal since it is under development. However, a preliminary analysis of possible limitations has been discussed in Section 9. Thanks to its generality, our proposal can be used in almost every domain in which historically KGs spread. We have already different data coming from the different domains (retrocomputing, food, tourism, . . .) and we are now employed in gathering data coming from the education domain. Each domain has its proper ontology formalization. Given the possibility of using intelligent information retrieval tasks, and logical (and SW) reasoning, many applications come to mind such as recommender systems, ITS devices and so on. The range of the applications is strongly related to the domains of data available.

6.2. Schemes

For the above-mentioned reasons, we decided to provide common schemes for data which can, possibly, be used as ontologies for reasoning. For this reason, we do not represent schemes in the data itself. We have already mentioned that LPG is not able to conceptually separate schemes from instances. We store different (partial) models of the data in external (XML) documents.

Schemes determine the concepts we can describe and can also be used as a way to check integrity constraints. This will be better examined in Section 8.3. Additionally, schemes provide an abstraction of data. Schemes are modelled regardless of the specific representation of instances. The storage process is fully transparent to the end user. When external data is to be imported, OWL schemes will be mapped as well and data will refer to them too.

Our scheme, at the state of the art, provides the following concepts to be represented: classes and subclasses, relationships and sub-relationships, and attributes on classes and relationships, which may have as types not only literals (string, integer, float, . . .) but also other classes as well. Schemes are built (or imported) in order to stay in OWL-DL, which

is decidable. This property would not be assured if we entered in the OWL-Full that is undecidable [160,161]. A first overview of our DTD has been described in [158]. In that version, sub-relationships and user types were not available. The structure has been slightly modified in order to implement new features.

For the implementation of our schema, we chose XML as a formal language to describe concepts.

The choice of XML is two-fold. First of all, it is a standard, general and machine-readable format to express concepts and relationships through tags; secondly, XML is at a higher level of abstraction than a graph. This means that we can use it to move to different contexts. As the first aftermath, XML documents can be edited by (non-lay) users and this encourages students, researchers or other people concerning the topic to create and/or update their own schemes for returning them to the community. Moreover, XML can function as an intermediate layer for many kinds of reasoning. For example, XML schemes can be rendered into OWL, which is on another level of the Semantic Web pyramid. This does not mean the two languages are equivalent, although we can map XML schemes into equivalent ones expressed in OWL. Additionally, XML schemes can be translated into formal logic languages, making us prone to use Logic Programming (LP) as well. Logics contribute to many kinds of reasoning frameworks and we will briefly introduce them later on in Section 7. Finally, XML guarantees semantic interoperability, which is the reason why it is so popular for data exchange on the Web. The first step was to define the DTD, which is the formal rigorous structure of any scheme document into GraphBRAIN. The DTD not only specifies the format and the syntax, but it also expresses the power of the language, given all the possible elements (tags) we can represent. It mostly describes how we represent the first (most obvious) features of an ontology such as (sub-)classes, (sub-)relationships, attributes of classes (or relationships) and so on. Tables 3 and 4 show a portion of ontology in our format with the main components present.

6.3. Import/Export with Semantic Web

A fundamental part of our work is devoted to the possibility of allowing GraphBRAIN and the SW to communicate with each other. From one side, researchers may find it advantageous to bring OWL/RDF resources into GraphBRAIN to explore the information-retrieving functionalities provided. On the other side, we are interested in enriching the SW community with data gathered from several resources. Moreover, one of the main advantages of the proposal is exactly the possibility of exhibiting reasoning capabilities by means of different mapping into (logical) formalisms.

In literature, there is a plethora of works which mapped two different data models, starting from mapping relational tables and RDF [162] until modern ones, some already mentioned in Section 5. For the mapping between the two formalisms, we are considering a mixed strategy between a fully automated mapping and a manually defined one. All in all, we would rather have a default strategy which can be slightly modified by specific user requests. For instance, a user may want to specify how to map a specific data property, object property and node, even changing the type of the concept.

The default mapping strategy is reported in Section 6.4.

Dealing with the opposite direction, the mapping goes vice versa always taking into account possible user preferences. In this direction, the ontological SW reasoning becomes feasible.

6.4. Mapping RDF and LPG

As described above, nevertheless RDF and LPG both use the graph data structure, they are partly incompatible and need a process of integration, or an intermediate layer in between. Since one goal of our proposal is to be able to perform SW reasoning, we are now focusing on moving from LPG to RDF, mapping data and schemes into a (formal) SW-compliant language. Yet the inverse process is a priority for us since we also aim to import already-available resources in the SW to enrich our database. First of all, when

moving to the SW field, URIs are imperative. For our purposes, we are using the following namespace: "https://gbnamespace" whose given abbreviation is "gb". Indeed, every concept (class/relationship) must have that namespace in its URI.

The first step is to define how to translate schemes expressed in XML into OWL, the language for describing schemes in the SW. In this case, the mapping becomes quite intuitive given the analogy among concepts. For example, there is a natural connection between entities in ontologies and classes in OWL, or between attributes and datatype properties or relationships and object properties. We easily report the translation from XML to OWL in Table 5, already presented in our previous work.

Additionally, we need a methodology to express resource data from the DB perspective in LPG into the SW perspective in the form of RDF triples, with the subject-predicate-object structure. Before going on to describe the methodology, several relevant issues emerge. First of all, one essential decision regards how much and what information to map. At a first glance, it seems natural to translate everything you find, but it may lead to a substantial collapse in performance, as well as security and privacy issues. Should you decide to move just part of the data, you will also need an approach to establish what to translate and reason with. Another relevant factor is that consistency cannot be guaranteed for free. Traditionally, databases change constantly. There is no denying that translating data once is not enough. Whenever something changes, you need a new version of the KB expressed in RDF; otherwise, wrong or inconsistent (with the respect to the external world) conclusions can emerge.

We also provide a graphical representation of how concepts are mapped in the two different graphs. We provide the JSON code associated with a relationship between two nodes. Each concept also has properties as shown in Table 6. For the sake of simplicity, but without losing generality, we just have a student, an exam, and a relationship stating that the student passed that exam. In Figure 1, the LPG representation is shown. The mapped RDF graph is shown in Figure 2 after applying all the mappings described in the table.

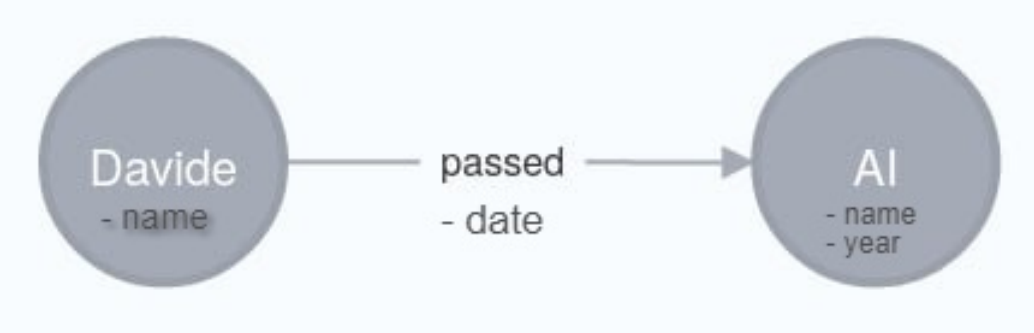

Figure 1. LPG representation of two nodes and a relationship with attributes.

Finally, when integrating the schema into OWL and the triples in RDF, SW reasoner [163] can execute several interesting tasks, among them the most common is consistency checking. In [164] you can also find an example of an application of this mechanism with other considerations related to the specific use case.

Even though the advantages of enriching KBs with ontological reasoning are evident, we are also going to explore alternative approaches, leveraging logic again. Being able to export data into a logical formalism opens new scenarios. With logic formulas, we would be able to apply many state-of-the approaches and even combine them, possibly. We will refer to these approaches with the umbrella term of "multistrategy reasoning".

Table 3. Sample fragment of ontology in GraphBRAIN format (part 1).

```
<!--  <!DOCTYPE domain SYSTEM "graphbrain.dtd">  -->
<domain name="retrocomputing" author="stefano" version="1">
   <entities>
      <entity name="Component">
         <attributes>
            <attribute name="name" mandatory="true" datatype="string"/>
            <attribute name="description" mandatory="false" datatype="text"/>
            <attribute name="originalPrice" mandatory="false" datatype="real"/>
            <attribute name="announcementDate" mandatory="false" datatype="date"/>
         </attributes>
         <taxonomy>
            <value name="Chip">
               <values>
                  <value name="Logic">
                     <taxonomy>
                        <value name="FlipFlop">
                           <attributes>
                              <attribute name="type"
                                    mandatory="false" datatype="select">
                                 <values>
                                    <value name="D"/>
                                    <value name="FK"/>
                                    <value name="JK"/>
                                    <value name="T"/>
                                 </values>
                              </attribute>
                           </attributes>
                        </value>
                        <value name="Memory">
                           <attributes>
                              <attribute name="capacity"
                                    mandatory="false" datatype="string"/>
                              <attribute name="speed"
                                    mandatory="false" datatype="string"/>
                           </attributes>
                           <taxonomy>
                              <value name="EPROM"/>
                              <value name="PROM"/>
                              <value name="RAM"/>
                              <value name="ROM">
                                 <attributes>
                                    <attribute name="content"
                                          mandatory="false" datatype="string"/>
                                 </attributes>
                              </value>
                           </taxonomy>
                        </value>
                     </taxonomy>
                  </value>
                  <value name="MicroProcessor">
                     <attributes>
                        <attribute name="speed" mandatory="false" datatype="string"/>
                        <attribute name="bits" mandatory="false" datatype="integer"/>
                     </attributes>
                  </value>
                  <value name="PLA"/>
                  <value name="RRIOT"/>
               </taxonomy>
            </taxonomy>
            [...]
         </taxonomy>
      </entity>
      [...]
   </entities>
```

Table 4. Sample fragment of ontology in GraphBRAIN format (part 2).

```
   <relationships>
      <relationship name="wasIn" inverse="hosted">
         <references>
            <reference subject="Company" object="Event"/>
            [...]
         </references>
         <attributes>
            <attribute name="reason" mandatory="false" datatype="string"/>
            <attribute name="position" mandatory="false" datatype="string"/>
         </attributes>
         <taxonomy>
            <value name="workedIn"/>
         </taxonomy>
      </relationship>
      [...]
   </relationships>
</domain>
```

Table 5. Translations from concepts in the ontology to concepts in the SW [164].

XML Scheme	Semantic Web
Entity	owl:Class
Entity Attribute	owl:DatatypeProperty
Relationship	owl:ObjectProperty
Subject of Relationship	owl:ObjectPropertyDomain
Object of Relationship	owl:ObjectPropertyRange

With the assumption that attributes are always literal, which is not guaranteed in principle by Neo4j, some mapping rules can be applied. Again, we show the rules through Table 7.

Table 6. JSON fragment of the graph.

```
{
  "identity": 371278,
  "labels": [
    "Student"
  ],
  "properties": {
"name": "Davide",
"idStudent": "661292"
  }
}
```

```
{
  "identity": 1070551,
  "start": 371278,
  "end": 371298,
  "type": "passed",
  "properties": {
"date": "17/07/2020"
  }
}
```

```
{
  "identity": 371298,
  "labels": [
    "Course"
  ],
  "properties": {
"name": "AI",
"year": 2,
  }
}
```

Table 7. Translations from data in the LPG graph to data in the SW [164].

LPG	RDF
Node	rdf:subject
Arc	rdf:predicate
Attribute on node	rdf:predicate between the rdf:subject (node) and the literal (Attribute value)
Attribute on relationship	rdf:predicate between the rdf:statement representing the triple (relationship) and the literal (Attribute value)

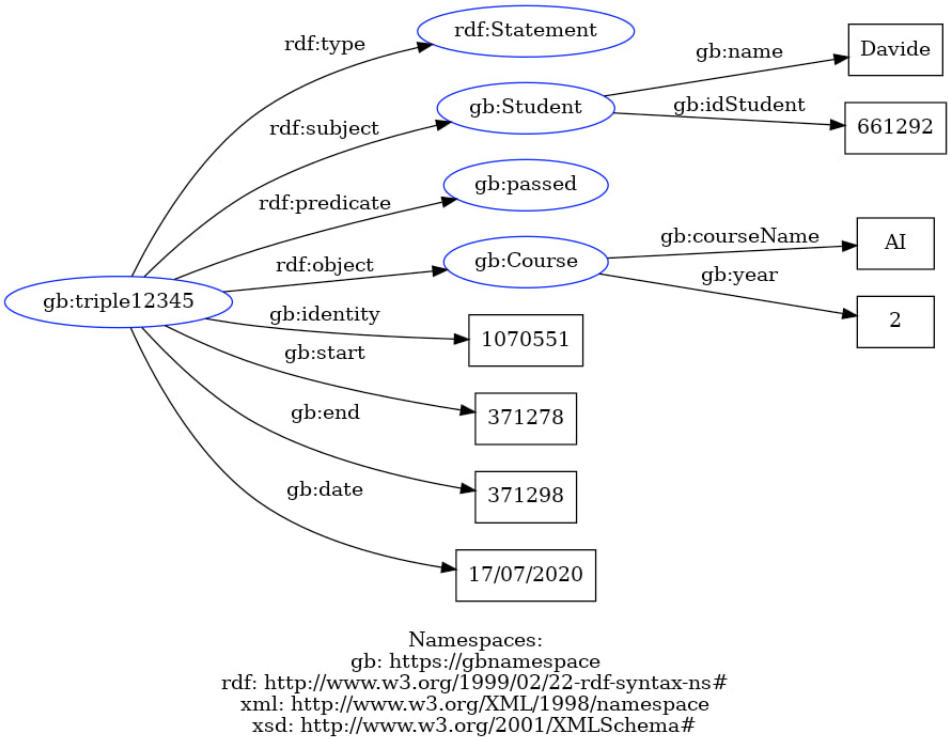

Figure 2. RDF representation after mapping. As you can see, the "passed" relationship is now reified as a statement.

7. Multistrategy Reasoning

"Multistrategy reasoning" [165] is concerned with extracting information from a KB by following one (or more) strategies. The different strategies follow human reasoning methods that are feasible in any real-world context. We will briefly explain which inference mechanisms we consider part of multistrategy reasoning. Once a KB is expressed through a logic language (e.g., Prolog), all these approaches can be adopted to enrich the overall knowledge. This approach must be considered in specific application domains such as GraphBRAIN, in which the Closed World Assumption [166] holds. This is not true in general, and in particular for the Semantic Web where the Open World Assumption [167] is adopted.

GraphBRAIN provides a mechanism to map XML schemes into Prolog, while the translation of graph instances is a work in progress.

7.1. Abstraction

Conceptual abstraction and analogy-making are key abilities underlying humans' abilities to learn, reason, and robustly adapt their knowledge to new domains [168]. Abstraction is the type of reasoning aimed at removing unimportant details during the modelling of the solution. It solves different problems:

- The complexity of the problem is reduced.
- The problem becomes goal-dependent.
- Problems previously considered impossible become possible.

In general terms, we can define the abstraction process as a mapping at the level of the perceived world. Formally, given a world W, let $R_g = (P_g(W), S_g, L_g)$ and $R_a = (P_a(\mathrm{W}), S_a, L_a)$ be two reasoning contexts ground and abstract, an abstraction is a functional mapping A: $P_g(\mathrm{W}) \rightarrow P_a(\mathrm{W})$ between a perception $P_g(\mathrm{W})$ and a simpler perception $P_a(\mathrm{W})$ of the same world W. For "simpler" we mean that there is a function from the elements of the first perception to the element of the second one and this function is not injective.

7.2. Deduction

Deduction is the most common and famous type of reasoning, for historical and practical reasons. The psychology of reasoning has, for many years, been centred on the deductive reasoning paradigm in which people are asked to assess logical arguments or generate valid conclusions from given premises. The deductive reasoning paradigm is focused on logical arguments. These are arguments whose conclusions must necessarily follow from their premises. That is to say, logic guarantees that a valid conclusion is true if all of the premises are true. Whether the premises are, in fact, true is neither here nor there [169].

7.3. Induction

Inductive Logic Programming (ILP) can be seen as the merge of two areas: Logic Programming and Inductive Learning. It provides a formal framework and algorithms for inductive learning of relational descriptions expressed as logic programs. In logical terms, the framework is the following: consider LO the language of observations, LB the language of background knowledge and LH the language of hypotheses.

The problem of induction is: given $O \subseteq LO$ and $B \subseteq LB$, find an hypothesis $H \in LH$ s.t.:

- $B \wedge H \models O$

that is, the background knowledge and the hypothesis logically prove the observations [170]. In a good inductive argument, the premises should provide some degree of support for the conclusion, where such support means that the truth of the premises indicates with some degree of strength that the conclusion is true. Presumably, if the logic of good inductive arguments is to be of any real value, the measure of support it articulates should meet the following condition: as evidence accumulates, the degree to which the collection of true evidence statements comes to support a hypothesis, as measured by the logic, should tend to indicate that false hypotheses are probably false and that true hypotheses are probably true [171].

7.4. Abduction

Abduction is the type of reasoning aimed at inferring causes from effects. Peirce defined it as the inference process of forming a hypothesis that explains given observed phenomena [172]. Its goal is to hypothesize unknown information starting from observations. It has the following characteristics:

- It handles missing information.
- Different explanations are possible.
- Many constraints must be satisfied.

More formally, given:

- a logical theory T representing the expert knowledge and
- a formula Q representing an observation on the problem domain,

an abductive inference is an explanation formula E such that:

- E is satisfiable with respect to T.
- $T \models E \rightarrow \mathrm{Q}$

In general, we prefer E to be minimal (e.g., by restricting the number of predicates).

The abductive explanation of an observation is a formula which logically entails the observation and represents a cause for it.

We remind the strong difference between abduction and induction. The first one wants to infer causes from observation while the second wants to deduct consequences from observations. We provide an example:

- If I say that my car will not start this morning, an *abductive solution* is the explanation that its battery is empty. An *inductive inference* can infer that if the battery is empty, then the car will not start.

7.5. Argumentation

Argumentation studies the processes and activities involved in the production and exchange of arguments. It aims at identifying, analyzing and evaluating arguments. It captures diverse kinds of reasoning and dialogue activities in a formal but still intuitive way and provides procedures for making and explaining decisions. In AI, it is used for giving reasons to support claims that are open to doubt and/or defend these claims against attack. Argumentation takes a clue from everyday arguing which is something typical, involving different people who have their perspectives which look convincing.

In general, arguments are represented as nodes and relationships amongst arguments as arcs. It is easy to recognize a graph structure. Arcs are directed and mean that the source argument attacks the second one.

There are different frameworks for representing arguments. The basic one was proposed by Dung [173] but it turned out to be too limited. Hence, the following ones emerged:

- *Bipolar*: provides arcs also for supporting arguments, not only to attack.
- *Weighted*: provides weights for attacks, to distinguish them.
- *Trusted*: provides weight also for the arguments themselves.
- *Mixed*: uses a combination of the previous ones, for example, bipolar and weighted.

Through this formalization, relevant aspects of the arguments can emerge, in particular some subsets of the overall set of arguments, such as the conflict-free, the admissible, the semi-stable and so on.

Major applications of this kind of reasoning are the debates, in which we can well formalize arguments, attacks between them, and, if using richer frameworks, possible weights for describing support relationships.

7.6. Analogy

Analogy is the process of transferring knowledge across domains. The main reasons for which it is used are the following:

- Relevance in the study of learning, for moving from one domain to another.
- Often used in problem-solving.
- Relevance when studying new domains.
- In the past, it inspired great scientists in new discoveries.
- Frequently used in communication.
- Frequently used for explanations.

The Analogy process is not straightforward. It requires the following phases:

- *Retrieval*: finding the better base domain that can be useful to solve the task in the target domain.
- *Mapping*: searching for a mapping between base and target domains.
- *Evaluation*: providing some criteria to evaluate the candidate mapping.
- *Pattern*: shifting the representation of both domains to their roles schema, converging to the same analogical pattern.
- *Re-representation*: adapting one or more pieces of the representation to improve the match.

A summary of all the strategy reasoning techniques and a schema describing the collocation context of each of them is shown in [174].

8. Validation

One of the main problems when dealing with this validation is that inferences coming up need some external expert (or tool) in order to be verified. Most case studies require completely variegated teams which carry out this task specifically. It is not unusual that the information extracted is already well-known to the experts and so further investigation is needed in order to discover non-trivial information. The validation team is strongly dependent on the context of use and, at the state of the art, there is no way to make this process general.

8.1. SHACL

Despite the difficulties in validating results output from the process of inference, much attention is being put on the validation of the KBs themselves. In particular, Shapes Constraint Language (SHACL) is the W3C recommendation language for integrity constraints over RDF knowledge graphs [175]. In SHACL, validation is based on shapes, which define particular constraints and specify which nodes in a graph should be validated against these constraints. Data validation requires two inputs: an rdf graph G to be validated and a shacl document M that defines the conditions against which G must be evaluated. The shacl specification defines the output of the data validation process as a validation report, detailing all the violations that were found in G of the conditions set by M. If the violation report contains no violations, a graph G is valid w.r.t. SHACL document M. Formally, a shacl document is a set of shapes. Validating a graph against a shacl document involves validating it against each shape. Shapes restrict the structure of a valid graph by focusing on certain nodes and examining whether they satisfy their constraints [176]. We can think about this problem from a new perspective: as a problem of satisfiability. Given a particular SHACL document, satisfiability is the problem of deciding whether there is an RDF graph which is validated by the document [177].

8.2. ShEx

For validation purposes, there is also an alternative sometimes used in different contexts but also in combination with SHACL. Shape Expression Schema (ShEx) is a novel schema formalism for RDF currently under development by W3C. ShEx assigns types to the nodes of an RDF graph and allows constraining the admissible neighbourhoods of nodes of a given type with regular bag expressions. A ShEx allows defining a set of types that impose structural constraints on nodes and their immediate neighbourhood. The complexity of single-type validation for ShEx is NP-complete [178].

The construction of SHACL or ShEx schemes remains a difficult problem. It requires mastering different tools and languages and swapping between them in order to complete a schema construction task: the syntax and semantics of the constraint language, the existing validation APIs or tools, query languages, or other means of exploring the data [179]. There are plenty of works [180–185] trying to embed these instruments into graphical tools in order to facilitate experts of other fields (lay users) to help during the validation process.

8.3. Our Validation

With respect to the standard approaches used today, we have at our disposal a natural solution to this problem since we separated the schemes from the instances. In this way, each time data is going to be inserted, we have to check whether some inconsistencies with respect to some attributes or relationships emerge. Further complex controls such as mandatory relationships and/or cardinalities on some properties are out of our scope for now, but we are considering how to empower integrity constraints. Very naturally, we are going to implement mechanisms to assuring that no external attributes or relationships appear and that label mismatch will not be allowed.

Up to a certain extent, we are considering SHACL and ShEx but, for the reasons described above, we keep separate schemes and instances. Hence, their application is out of scope now.

9. Possible Limitations

We are going to now briefly discuss some possible drawbacks or limitations of our proposal. Please be reminded that the proposal is not fully implemented so any performance or quantitative analysis would be unfeasible. Nonetheless, we are able to point out possible critical points. First of all, if the LPG graph is relatively vast (as it is), it is neither advisable nor even feasible to translate all instances in RDF. For this reason, we are considering which are the possible techniques to tackle this problem. At the moment, we are thinking about extracting a single domain of interest so that the overall dimension is considerably less

than the total amount. Even this idea turns out to be inefficient in the case where a specific domain is general and full of knowledge. In parallel with the first thought, we are also examining how to extract relevant pieces of knowledge starting from specific instances of interest. With some graph mining algorithms, we can depict the hubs for a specific domain and, starting from them, retrieve neighbours of those hubs.

Another relevant aspect that needs consideration: the RDF processing. In principle, we are considering processing RDF KBs on the fly. That means that whenever we want to exploit SW reasoning, we map pieces of data and apply some reasoners. However, it may be convenient to store pieces of KBs as triple stores so they are available in case multiple runs on the same knowledge are considered valuable. Keep also in mind that our database can be updated at any moment since it is developed for a web application. For this reason, were we to decide to store some RDF triples we would encounter the inconsistency problem whenever a piece of data in the graph is updated and already mapped into RDF.

Despite all these discussions being relevant, specific answers to these questions can be given only when applications will be based on our solution. According to specific needs, some applications may privilege some approaches rather than others.

10. Conclusions

In this work, we illustrated the state-of-the-art issues related to the construction of knowledge graphs with graph databases, in particular the LPG model used by the most common graph DBMS. Until now, many methodologies have been proposed but none of them seem to be completely context-independent. Furthermore, when dealing with different graph models, many problems in terms of mapping and/or translation occur. In contrast with the state of the art, we proposed a novel approach which applies the separation of concerns between schemes and instances, and we provided a mapping from them into OWL/RDF, so that SW reasoning becomes feasible. Moreover, we mentioned the possibility of enriching the kind of reasoning, including logic-based reasoning. We aimed to integrate ontologies, graph databases and a Semantic Web to support and stimulate the use of graphs for the storage and maintenance of data sources, especially in the context of knowledge extraction. Future works comprise the generality of the problem and the understanding of the real potentiality of the approach, also developing (academic or production) solutions for the community.

Author Contributions: Conceptualization of the problem, S.F. and D.D.P.; Methodology, S.F., D.R. and D.D.P.; Software, S.F. and D.D.P.; SW perspective, D.R.; State of the art, D.R. and D.D.P.; Validation, S.F. and D.R.; Supervision, S.F. and D.R.; Resource, S.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
KRR	Knowledge Representation and Reasoning
KB	Knowledge Base
QA	Question Answering
KG	Knowledge Graph
SW	Semantic Web
NLP	Natural Language Processing
PG	Property Graph

DBMS	DataBase Management System
QL	Query Language
SQL	Structured Query Language
DSL	Domain Specific Language
LPG	Labelled Property Graph
RDF	Resource Description Framework
OWL	Web Ontology Language
XML	eXtensible Markup Language
DTD	Document Type Definition
URI	Uniform Resource Identifier
PG	Property Graph
RNN	Recurrent Neural Network
YARS	Yet Another RDF Serialization
SPG	Semantic Property Graph
LP	Logic Programming
ILP	Inductive Logic Programming
SHACL	Shapes Constraint Language
ShEx	Shape Expression Schema

References

1. Simmons, A.B.; Chappell, S.G. Artificial intelligence-definition and practice. *IEEE J. Ocean. Eng.* **1988**, *13*, 14–42. [CrossRef]
2. Rich, E. *Artificial Intelligence*; McGraw-Hill, Inc.: New York, NY, USA, 1983.
3. Dick, S. Artificial intelligence. *Harv. Data Sci. Rev.* **2019**. [CrossRef]
4. Holmes, J.; Sacchi, L.; Bellazzi, R. Artificial intelligence in medicine. *Ann. R. Coll. Surg. Engl.* **2004**, *86*, 334–338.
5. Newell, A.; Shaw, J.; Simon, H. Report on a general problem solving program. In Proceedings of the IFIP Congress, Pittsburgh, PA, USA, 15–20 June 1959; Volume 256, p. 64.
6. Shortliffe, E. *Computer-Based Medical Consultations: MYCIN*; Elsevier: Amsterdam, The Netherlands, 2012; Volume 2.
7. Moussa, A.M.; Abdel-Kader, R.F. Qasyo: A question answering system for yago ontology. *Int. J. Database Theory Appl.* **2011**, *4*, 99–112.
8. Fu, B.; Qiu, Y.; Tang, C.; Li, Y.; Yu, H.; Sun, J. A survey on complex question answering over knowledge base: Recent advances and challenges. *arXiv* **2020**, arXiv:2007.13069.
9. Stokman, F.; Vries, P. Structuring knowledge in a graph. In *Human-Computer Interaction*; Springer: Berlin, Germany, 1988; pp. 186–206.
10. Bordes, A.; Weston, J.; Collobert, R.; Bengio, Y. Learning structured embeddings of knowledge bases. In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011.
11. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Philip, S. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 494–514. [CrossRef] [PubMed]
12. Gruber, T. A translation approach to portable ontology specifications. *Knowl. Acquis.* **1993**, *5*, 199–220. [CrossRef]
13. Angles, R.; Gutierrez, C. Survey of graph database models. *ACM Comput. Surv. (CSUR)* **2008**, *40*, 1–39. [CrossRef]
14. Erdös, P. Some remarks on the theory of graphs. *Bull. Am. Math. Soc.* **1947**, *53*, 292–294. [CrossRef]
15. W3C. RDF Primer. 2014. Available online: https://www.w3.org/TR/rdf-primer/ (accessed on 20 January 2023).
16. Uifălean, S.; Ghiran, A.; Buchmann, R.A. From BPMN Models to Labelled Property Graphs. In Proceedings of the 30th International Conference on Information Systems Development (ISD2022), CLUJ-Napoca, Romania, 31 August–2 September 2022; pp. 697–706.
17. Anikin, D.; Borisenko, O.; Nedumov, Y. Labeled property graphs: SQL or NoSQL? In Proceedings of the 2019 Ivannikov Memorial Workshop (IVMEM), Velikiy Novgorod, Russia, 13–14 September 2019; pp. 7–13.
18. Raj, S. *Neo4j High Performance*; Packt Publishing Ltd.: Birmingham, UK, 2015.
19. Wylot, M.; Hauswirth, M.; Cudré-Mauroux, P.; Sakr, S. RDF data storage and query processing schemes: A survey. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–36. [CrossRef]
20. Berge, C. *The Theory of Graphs*; Courier Corporation: Chelmsford, MA, USA, 2001.
21. Han, J.; Haihong, E.; Le, G.; Du, J. Survey on NoSQL database. In Proceedings of the 2011 6th International Conference on Pervasive Computing and Applications, Port Elizabeth, South Africa, 25–29 October 2011; pp. 363–366.
22. Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; de Melo, G.; Gutierrez, C.; Kirrane, S.; Gayo, J.; Navigli, R.; Neumaier, S.; et al. Knowledge graphs. *ACM Comput. Surv.* **2021**, *54*, 1–37.
23. Pareja-Tobes, P.; Tobes, R.; Manrique, M.; Pareja, E.; Pareja-Tobes, E. Bio4j: A high-performance cloud-enabled graph-based data platform. *bioRxiv* **2015**, 016758.

24. Miller, J. Graph database applications and concepts with Neo4j. In Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, 24–25 March 2013; Volume 2324.
25. Huang, Z.; Chung, W.; Ong, T.; Chen, H. A graph-based recommender system for digital library. In Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, Portland, OR, USA, 14–18 July 2002; pp. 65–73.
26. Robinson, I.; Webber, J.; Webber, J.; Eifrem, E. *Graph Databases*; O'Reilly: Springfield, IL, USA, 2013.
27. Silberschatz, A.; Korth, H.; Sudarshan, S. Data models. *ACM Comput. Surv. (CSUR)* **1996**, *28*, 105–108. [CrossRef]
28. Codd, E. Data models in database management. In Proceedings of the 1980 Workshop on Data Abstraction, Databases and Conceptual Modeling, Colorado, CO, USA, 23 June 1980; pp. 112–114.
29. Han, J.; Pei, J.; Tong, H. *Data Mining: Concepts and Techniques*; Morgan Kaufmann: San Francisco, CA, USA, 2022.
30. Paredaens, J.; Peelman, P.; Tanca, L. G-Log: A graph-based query language. *IEEE Trans. Knowl. Data Eng.* **1995**, *7*, 436–453. [CrossRef]
31. Pokorný, J.; Valenta, M.; Kovačič, J. Integrity constraints in graph databases. *Procedia Comput. Sci.* **2017**, *109*, 975–981. [CrossRef]
32. Buneman, P.; Davidson, S.; Hillebrand, G.; Suciu, D. A query language and optimization techniques for unstructured data. In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, QC, Canada, 4–6 June 1996; pp. 505–516.
33. Güting, R. GraphDB: Modeling and querying graphs in databases. In *VLDB*; Citeseer: Princeton, NJ, USA, 1994; Volume 94, pp. 12–15.
34. Kaliyar, R.K. Graph databases: A survey. In Proceedings of the International Conference on Computing, Communication & Automation, Greater Noida, India, 15–16 May 2015; pp. 785–790.
35. Codd, E. A relational model of data for large shared data banks. In *Software Pioneers*; Springer: Berlin, Germany, 2002; pp. 263–294.
36. Abiteboul, S. Querying semi-structured data. In Proceedings of the International Conference on Database Theory, Delphi, Greece, 8–10 January 1997; Springer: Berlin, Germany, 1997; pp. 1–18.
37. Francis, N.; Green, A.; Guagliardo, P.; Libkin, L.; Lindaaker, T.; Marsault, V.; Plantikow, S.; Rydberg, M.; Selmer, P.; Taylor, A. Cypher: An evolving query language for property graphs. In Proceedings of the 2018 International Conference on Management of Data, Houston, TX, USA, 10–15 June 2018; pp. 1433–1445.
38. Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 601–610.
39. Nickel, M.; Murphy, K.; Tresp, V.; Gabrilovich, E. A review of relational machine learning for knowledge graphs. *Proc. IEEE* **2015**, *104*, 11–33. [CrossRef]
40. Wang, Q.; Mao, Z.; Wang, B.; Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 2724–2743. [CrossRef]
41. Gutierrez, C.; Sequeda, J.F. Knowledge graphs. *Commun. ACM* **2021**, *64*, 96–104. [CrossRef]
42. Abu-Salih, B. Domain-specific knowledge graphs: A survey. *J. Netw. Comput. Appl.* **2021**, *185*, 103076. [CrossRef]
43. Yan, J.; Wang, C.; Cheng, W.; Gao, M.; Zhou, A. A retrospective of knowledge graphs. *Front. Comput. Sci.* **2018**, *12*, 55–74. [CrossRef]
44. Noy, N.; Gao, Y.; Jain, A.; Narayanan, A.; Patterson, A.; Taylor, J. Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. *Queue* **2019**, *17*, 48–75. [CrossRef]
45. Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008; pp. 1247–1250.
46. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.; Hellmann, S.; Morsey, M.; Kleef, P.V.; Auer, S.; et al. Dbpedia—A large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web* **2015**, *6*, 167–195. [CrossRef]
47. Suchanek, F.M.; Kasneci, G.; Weikum, G. Yago: A core of semantic knowledge. In Proceedings of the 16th International Conference on World Wide Web, Banff, AB, Canada, 8–12 May 2007; pp. 697–706.
48. Fensel, D.; Şimşek, U.; Angele, K.; Huaman, E.; Kärle, E.; Panasiuk, O.; Toma, I.; Umbrich, J.; Wahler, A. Introduction: What is a knowledge graph? In *Knowledge Graphs*; Springer: Berlin, Germany, 2020; pp. 1–10.
49. Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E.; Mitchell, T. Toward an architecture for never-ending language learning. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010.
50. Heck, L.; Hakkani-Tür, D.; Tur, G. Leveraging knowledge graphs for web-scale unsupervised semantic parsing. In Proceedings of the INTERSPEECH, Lyon, France, 25–29 August 2013.
51. Damljanovic, D.; Bontcheva, K. Named entity disambiguation using linked data. In Proceedings of the 9th Extended Semantic Web Conference, Heraklion, Greece, 27–31 May 2012; pp. 231–240.
52. Zheng, Z.; Si, X.; Li, F.; Chang, E.Y.; Zhu, X. Entity disambiguation with freebase. In Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Washington, DC, USA, 4–7 December 2012; Volume 1, pp. 82–89.
53. Hoffmann, R.; Zhang, C.; Ling, X.; Zettlemoyer, L.; Weld, D. Knowledge-based weak supervision for information extraction of overlapping relations. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 541–550.

54. Daiber, J.; Jakob, M.; Hokamp, C.; Mendes, P. Improving efficiency and accuracy in multilingual entity extraction. In Proceedings of the 9th International Conference on Semantic Systems, New York, NY, USA, 4 September 2013; pp. 121–124.
55. Bordes, A.; Weston, J.; Usunier, N. Open question answering with weakly supervised embedding models. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Nancy, France, 15–19 September 2014; Springer: Berlin, Germany, 2014; pp. 165–180.
56. Chen, X.; Jia, S.; Xiang, Y. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.* **2020**, *141*, 112948. [CrossRef]
57. Gaines, B.R. Foundations of fuzzy reasoning. *Int. J. Man-Mach. Stud.* **1976**, *8*, 623–668. [CrossRef]
58. Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. *Sci. Am.* **2001**, *284*, 34–43. [CrossRef]
59. Hitzler, P. A review of the semantic web field. *Commun. ACM* **2021**, *64*, 76–83. [CrossRef]
60. Rettinger, A.; Lösch, U.; Tresp, V.; d'Amato, C.; Fanizzi, N. Mining the semantic web. *Data Min. Knowl. Discov.* **2012**, *24*, 613–662. [CrossRef]
61. Smith, B. Ontology. In *The Furniture of the World*; Brill: Leiden, The Netherlands, 2012; pp. 47–68.
62. Guarino, N. *Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98), 6–8 June 1998, Trento, Italy*; IOS Press: Cambridge, UK, 1998; Volume 46.
63. Iliadis, A. The Tower of Babel problem: Making data make sense with Basic Formal Ontology. *Online Inf. Rev.* **2019**, *43*, 1021–1045. [CrossRef]
64. Hitzler, P.; Krötzsch, M.; Parsia, B.; Patel-Schneider, P.F.; Rudolph, S. OWL 2 web ontology language primer. *W3C Recomm.* **2009**, *27*, 123.
65. Zou, Y.; Finin, T.; Chen, H. F-owl: An inference engine for semantic web. In *International Workshop on Formal Approaches to Agent-Based Systems*; Springer: Berlin, Germany, 2004; pp. 238–248.
66. Baader, F.; Calvanese, D.; McGuinness, D.; Patel-Schneider, P.; Nardi, D. *The Description Logic Handbook: Theory, Implementation and Applications*; Cambridge University Press: Cambridge, UK, 2003.
67. Decker, S.; Melnik, S.; Harmelen, F.V.; Fensel, D.; Klein, M.; Broekstra, J.; Erdmann, M.; Horrocks, I. The semantic web: The roles of XML and RDF. *IEEE Internet Comput.* **2000**, *4*, 63–73. [CrossRef]
68. Achard, F.; Vaysseix, G.; Barillot, E. XML, bioinformatics and data integration. *Bioinformatics* **2001**, *17*, 115–125. [CrossRef] [PubMed]
69. Papakonstantinou, Y.; Vianu, V. DTD inference for views of XML data. In Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Dallas, TX, USA, 15–18 May 2000; pp. 35–46.
70. W3C. RDF 11 Primer. 2014. Available online: https://www.w3.org/TR/rdf11-primer/ (accessed on 20 January 2023).
71. Decker, S.; Mitra, P.; Melnik, S. Framework for the semantic Web: An RDF tutorial. *IEEE Internet Comput.* **2000**, *4*, 68–73.
72. Pérez, J.; Arenas, M.; Gutierrez, C. Semantics and complexity of SPARQL. *ACM Trans. Database Syst. (TODS)* **2009**, *34*, 1–45. [CrossRef]
73. Grobe, M. Rdf, jena, sparql and the 'semantic web'. In Proceedings of the 37th Annual ACM SIGUCCS Fall Conference: Communication and Collaboration, St. Louis, MO, USA, 11–14 October 2009; pp. 131–138.
74. Erling, O.; Mikhailov, I. RDF Support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media*; Springer: Berlin, Germany, 2009; pp. 7–24.
75. Bellini, P.; Nesi, P. Performance assessment of RDF graph databases for smart city services. *J. Vis. Lang. Comput.* **2018**, *45*, 24–38. [CrossRef]
76. Baken, N. Linked data for smart homes: Comparing RDF and labeled property graphs. In Proceedings of the LDAC2020—8th Linked Data in Architecture and Construction Workshop, Online, 17–19 June 2020; pp. 23–36.
77. Holten, D.; Wijk, J.J.V. A user study on visualizing directed edges in graphs. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA, 4–9 April 2009; pp. 2299–2308.
78. Technologies, O.S. What Is a Labeled Property Graph? Available online: https://www.oxfordsemantic.tech/fundamentals/what-is-a-labeled-property-graph#:~:text=A%20labeled%20property%20graph%20(LPG,corresponding%20key%20to%20allow%20referencing (accessed on 20 January 2023).
79. Li, G.; Semerci, M.; Yener, B.; Zaki, M.J. Effective graph classification based on topological and label attributes. *Stat. Anal. Data Mining Asa Data Sci. J.* **2012**, *5*, 265–283. [CrossRef]
80. Jaffri, A.; Glaser, H.; Millard, I. Uri identity management for semantic web data integration and linkage. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*; Springer: Berlin, Germany, 2007; pp. 1125–1134.
81. Xing, W.; Ghorbani, A. Weighted pagerank algorithm. In Proceedings of the Second Annual Conference on Communication Networks and Services Research, Fredericton, NB, Canada, 21 May 2004; pp. 305–314.
82. Deng, H.; Lyu, M.; King, I. A generalized co-hits algorithm and its application to bipartite graphs. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 239–248.
83. He, Z.; Ma, Z.; Li, Z.; Giua, A. Parametric transformation of timed weighted marked graphs: Applications in optimal resource allocation. *IEEE/CAA J. Autom. Sin.* **2020**, *8*, 179–188. [CrossRef]
84. Elmoataz, A.; Desquesnes, X.; Toutain, M. On the game p-Laplacian on weighted graphs with applications in image processing and data clustering. *Eur. J. Appl. Math.* **2017**, *28*, 922–948. [CrossRef]
85. Pavlopoulos, G.A.; Kontou, P.I.; Pavlopoulou, A.; Bouyioukos, C.; Markou, E.; Bagos, P.G. Bipartite graphs in systems biology and medicine: A survey of methods and applications. *GigaScience* **2018**, *7*, giy014. [CrossRef]

86. Chan, T.M. More algorithms for all-pairs shortest paths in weighted graphs. In Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, San Diego, CA, USA, 11–13 June 2007; pp. 590–598.
87. Althöfer, I.; Das, G.; Dobkin, D.; Joseph, D.; Soares, J. On sparse spanners of weighted graphs. *Discret. Comput. Geom.* **1993**, *9*, 81–100. [CrossRef]
88. Asamoah, C.; Tao, L.; Gai, K.; Jiang, N. Powering filtration process of cyber security ecosystem using knowledge graph. In Proceedings of the 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), Beijing, China, 25–27 June 2016; pp. 240–246.
89. Troussas, C.; Krouska, A. Path-Based Recommender System for Learning Activities Using Knowledge Graphs. *Information* **2023**, *14*, 9. [CrossRef]
90. Anita, V.C.; Gangemi, A.; Mancinelli, M.; Marinucci, L.; Nuzzolese, A.G.; Presutti, V.; Veninata, C. ArCo: The Italian cultural heritage knowledge graph. In Proceedings of the International Semantic Web Conference, Auckland, New Zealand, 26–30 October 2019; Springer: Berlin, Germany, 2019; pp. 36–52.
91. Farazi, F.; Salamanca, M.; Mosbach, S.; Akroyd, J.; Eibeck, A.; Aditya, L.K.; Chadzynski, A.; Pan, K.; Zhou, X.; Zhang, S.; et al. Knowledge graph approach to combustion chemistry and interoperability. *ACS Omega* **2020**, *5*, 18342–18348. [CrossRef] [PubMed]
92. Shi, S.; Li, S.; Yang, X.; Qi, J.; Pan, G.; Zhou, B. Semantic health knowledge graph: Semantic integration of heterogeneous medical knowledge and services. *BioMed Res. Int.* **2017**, *2017*, 2858423. [CrossRef] [PubMed]
93. Fathalla, S.; Vahdati, S.; Auer, S.; Lange, C. Towards a knowledge graph representing research findings by semantifying survey articles. In Proceedings of the International Conference on Theory and Practice of Digital Libraries, Thessaloniki, Greece, 18–21 September 2017; Springer: Berlin, Germany, 2017; pp. 315–327.
94. Tomic, D.; Drenjanac, D.; Hoermann, S.; Auer, W. Experiences with creating a precision dairy farming ontology (DFO) and a knowledge graph for the data integration platform in agriOpenLink. *J. Agric. Inform.* **2015**, *6*. [CrossRef]
95. Rossetto, L.; Baumgartner, M.; Ashena, N.; Ruosch, F.; Pernischová, R.; Bernstein, A. LifeGraph: A knowledge graph for lifelogs. In Proceedings of the Third Annual Workshop on Lifelog Search Challenge, New York, NY, USA, 9 June 2020; pp. 13–17.
96. Bader, S.R.; Grangel-Gonzalez, I.; Nanjappa, P.; Vidal, M.; Maleshkova, M. A knowledge graph for industry 4.0. In Proceedings of the European Semantic Web Conference, Online, 2–4 June 2020; Springer: Berlin, Germany, 2020; pp. 465–480.
97. Szekely, P.; Knoblock, C.A.; Slepicka, J.; Philpot, A.; Singh, A.; Yin, C.; Kapoor, D.; Natarajan, P.; Marcu, D.; Knight, K.; et al. Building and using a knowledge graph to combat human trafficking. In Proceedings of the International Semantic Web Conference, Bethlehem, PA, USA, 11–15 October 2015; Springer: Berlin, Germany, 2015; pp. 205–221.
98. Liao, Z.; Wu, J.T.; Jiang, B.B.; Wang, J.D.; Yang, Y.R. Design methodology for flexible multiple plant water networks. *Ind. Eng. Chem. Res.* **2007**, *46*, 4954–4963. [CrossRef]
99. Leong, Y.T.; Lee, J.; Tan, R.R.; Foo, J.J.; Mei, I.C.L. Multi-objective optimization for resource network synthesis in eco-industrial parks using an integrated analytic hierarchy process. *J. Clean. Prod.* **2017**, *143*, 1268–1283. [CrossRef]
100. Tiu, B.T.C.; Cruz, D.E. An MILP model for optimizing water exchanges in eco-industrial parks considering water quality. *Resour. Conserv. Recycl.* **2017**, *119*, 89–96. [CrossRef]
101. Nair, S.K.; Guo, Y.; Mukherjee, U.; Karimi, I.A.; Elkamel, A. Shared and practical approach to conserve utilities in eco-industrial parks. *Comput. Chem. Eng.* **2016**, *93*, 221–233. [CrossRef]
102. Afshari, H.; Farel, R.; Peng, Q. Improving the resilience of energy flow exchanges in eco-industrial parks: Optimization under uncertainty. *ASCE-ASME J. Risk Uncert. Engrg. Sys. Part B Mech. Engrg.* **2017**, *3*. [CrossRef]
103. Zhang, C.; Zhou, L.; Chhabra, P.; Garud, S.S.; Aditya, K.; Romagnoli, A.; Comodi, G.; Magro, F.D.; Meneghetti, A.; Kraft, M. A novel methodology for the design of waste heat recovery network in eco-industrial park using techno-economic analysis and multi-objective optimization. *Appl. Energy* **2016**, *184*, 88–102. [CrossRef]
104. Tan, R.R.; Aviso, K.B. An inverse optimization approach to inducing resource conservation in eco-industrial parks. In *Computer Aided Chemical Engineering*; Elsevier: Amsterdam, The Netherlands, 2012; Volume 31, pp. 775–779.
105. Haslenda, H.; Jamaludin, M.Z. Industry to industry by-products exchange network towards zero waste in palm oil refining processes. *Resour. Conserv. Recycl.* **2011**, *55*, 713–718. [CrossRef]
106. Cimren, E.; Fiksel, J.; Posner, M.E.; Sikdar, K. Material flow optimization in by-product synergy networks. *J. Ind. Ecol.* **2011**, *15*, 315–332. [CrossRef]
107. Zhou, X.; Eibeck, A.; Lim, M.Q.; Krdzavac, N.B.; Kraft, M. An agent composition framework for the J-Park Simulator-A knowledge graph for the process industry. *Comput. Chem. Eng.* **2019**, *130*, 106577. [CrossRef]
108. Kaiser, F.K.; Dardik, U.; Elitzur, A.; Zilberman, P.; Daniel, N.; Wiens, M.; Schultmann, F.; Elovici, Y.; Puzis, R. Attack Hypotheses Generation Based on Threat Intelligence Knowledge Graph. *IEEE Trans. Dependable Secur. Comput.* **2023**, 1–17. [CrossRef]
109. Li, Z.; Li, Y.; Liu, Y.; Liu, C.; Zhou, N. K-CTIAA: Automatic Analysis of Cyber Threat Intelligence Based on a Knowledge Graph. *Symmetry* **2023**, *15*, 337. [CrossRef]
110. Kaloroumakis, P.E.; Smith, M.J. *Toward a Knowledge Graph of Cybersecurity Countermeasures*; MITRE Corporation: Bedford, MA, USA, 2021; p. 11.
111. Sharma, C.; Sinha, R. A schema-first formalism for labeled property graph databases: Enabling structured data loading and analytics. In Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, Auckland, New Zealand, 2–5 December 2019; pp. 71–80.
112. Rodriguez, M.A.; Neubauer, P. Constructions from dots and lines. *Bull. Am. Soc. Inf. Sci. Technol.* **2010**, *36*, 35–41. [CrossRef]

113. Rodriguez, M.A.; Neubauer, P. The graph traversal pattern. In *Graph Data Management: Techniques and Applications*; IGI Global: Hershey, PA, USA, 2012; pp. 29–46.
114. Cudré-Mauroux, P.; Elnikety, S. Graph data management systems for new application domains. *Proc. Vldb Endow.* **2011**, *4*, 1510–1511. [CrossRef]
115. Dominguez-Sal, D.; Martinez-Bazan, N.; Muntes-Mulero, V.; Baleta, P.; Larriba-Pey, J.L. A discussion on the design of graph database benchmarks. In Proceedings of the Technology Conference on Performance Evaluation and Benchmarking, Singapore, 13–17 September 2010; Springer: Berlin, Germany, 2010; pp. 25–40.
116. Wang, D.; Cui, W.; Qin, B. CK-modes clustering algorithm based on node cohesion in labeled property graph. *J. Comput. Sci. Technol.* **2019**, *34*, 1152–1166. [CrossRef]
117. Kalva, C.T.; Abhishek, K.; Vutnoori, A.; Maloth, V.K. Semantic Filtering of Twitter Data Using Labeled Property Graph (LPG). *J. Comput. Theor. Nanosci.* **2020**, *17*, 195–200. [CrossRef]
118. Bryant, C.H.; Muggleton, S.H.; Kell, D.B.; Reiser, P.; King, R.D.; Oliver, S.G. Combining inductive logic programming, active learning and robotics to discover the function of genes. *Electron. Trans. Artif. Intell.* **2001**, *5*, 1–36.
119. Goto, S.; Okuno, Y.; Hattori, M.; Nishioka, T.; Kanehisa, M. LIGAND: Database of chemical compounds and reactions in biological pathways. *Nucleic Acids Res.* **2002**, *30*, 402–404. [CrossRef] [PubMed]
120. Hughes, T.R.; Marton, M.J.; Jones, A.R.; Roberts, C.J.; Stoughton, R.; Armour, C.D.; Bennett, H.A.; Coffey, E.; Dai, H.; D, Y.H.; et al. Functional discovery via a compendium of expression profiles. *Cell* **2000**, *102*, 109–126. [CrossRef] [PubMed]
121. King, R.D.; Muggleton, S.H.; Srinivasan, A.; Sternberg, M.J. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proc. Natl. Acad. Sci. USA* **1996**, *93*, 438–442. [CrossRef] [PubMed]
122. King, R.D.; Whelan, K.E.; Jones, F.M.; Reiser, P.G.K.; Bryant, H.C.; Muggleton, S.H.; Kell, D.B.; Oliver, S.G. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature* **2004**, *427*, 247–252. [CrossRef] [PubMed]
123. Kitano, H. Computational systems biology. *Nature* **2002**, *420*, 206–210. [CrossRef]
124. Kitano, H. Systems biology: A brief overview. *Science* **2002**, *295*, 1662–1664. [CrossRef]
125. Muggleton, S.H.; Bryant, C.H. Theory completion using inverse entailment. In Proceedings of the International Conference on Inductive Logic Programming, London, UK, 24–27 July 2000; Springer: Berlin, Germany, 2000; pp. 130–146.
126. Muggleton, S.H.; Lodhi, H.; Amini, A.; Sternberg, M.J.E. Support vector inductive logic programming. In Proceedings of the International Conference on Discovery Science, Singapore, 8–11 October 2005; Springer: Berlin, Germany, 2005; pp. 163–175.
127. Kuznetsov, S.O.; Samokhin, M.V. Learning closed sets of labeled graphs for chemical applications. In Proceedings of the International Conference on Inductive Logic Programming, Bonn, Germany, 10–13 August 2005; Springer: Berlin, Germany, 2005; pp. 190–208.
128. Formanowicz, P.; Kasprzak, M.; Wawrzyniak, P. Labeled Graphs in Life Sciences—Two Important Applications. In *Graph-Based Modelling in Science, Technology and Art*; Springer: Berlin, Germany, 2022; pp. 201–217.
129. Zaki, N.; Chandana, T.; Hany, A.A. Knowledge graph construction and search for biological databases. In Proceedings of the 2017 International Conference on Research and Innovation in Information Systems (ICRIIS), Langkawi, Malaysia, 16–17 July 2017; pp. 1–6.
130. Silva, V.; Freitas, A.; Handschuh, S. Building a Knowledge Graph from Natural Language Definitions for Interpretable Text Entailment Recognition. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018.
131. Fellbaum, C. WordNet. In *Theory and Applications of Ontology: Computer Applications*; Springer: Berlin, Germany, 2010; pp. 231–243.
132. Penev, L.; Dimitrova, M.; Senderov, V.; Zhelezov, G.; Georgiev, T.; Stoev, P.; Simov, K. OpenBiodiv: A Knowledge Graph for Literature-Extracted Linked Open Data in Biodiversity Science. *Pensoft Publ.* **2019**, *7*, 38. [CrossRef]
133. Purohit, S.; Van, N.; Chin, G. Semantic Property Graph for Scalable Knowledge Graph Analytics. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021.
134. Spanos, D.E.; Stavrou, P.; Mitrou, N. Bringing relational databases into the semantic web: A survey. *Semant. Web* **2012**, *3*, 169–209. [CrossRef]
135. Vavliakis, K.N.; Grollios, T.; Mitkas, P. RDOTE–Publishing Relational Databases into the Semantic Web. *J. Syst. Softw.* **2013**, *86*, 89–99. [CrossRef]
136. Angles, R.; Gutierrez, C. Querying RDF Data from a Graph Database Perspective. In *The Semantic Web: Research and Applications*; Springer International Publishing: New York, NY, USA, 2005; pp. 346–360.
137. Sakr, S.; Elnikety, S.; He, Y. G-SPARQL: A Hybrid Engine for Querying Large Attributed Graphs. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, New York, NY, USA, 29 October–2 November 2012; CIKM'12, pp. 335–344.
138. Libkin, L.; Reutter, J.; Vrgoč, D. Trial for RDF: Adapting graph query languages for RDF data. In Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, New York, NY, USA, 22 June 2013; pp. 201–212.
139. Thakkar, H.; Punkani, D.; Keswani, Y.; Lehmann, J.; Auer, S. A Stitch in Time Saves Nine—SPARQL querying of Property Graphs using Gremlin Traversals. *arXiv* **2018**, arXiv:1801.02911.
140. Rodriguez, M. The gremlin graph traversal machine and language (invited talk). In Proceedings of the 15th Symposium on Database Programming Languages, Pittsburgh, PA, USA, 27 October 2015; pp. 1–10.

141. Virgilio, R.D. Smart RDF data storage in graph databases. In Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain, 14–17 May 2017; pp. 872–881.
142. Tomaszuk, D. RDF data in property graph model. In Proceedings of the Research Conference on Metadata and Semantics Research, Göttingen, Germany, 22–25 November 2016; Springer: Berlin, Germany, 2016; pp. 104–115.
143. Iordanov, B. Hypergraphdb: A generalized graph database. In Proceedings of the International Conference on Web-Age Information Management, Jiuzhaigou, China, 15–17 July 2010; Springer: Berlin, Germany, 2010; pp. 25–36.
144. Das, S.; Perry, M.; Srinivasan, J.; Chong, E. *A Tale of Two Graphs: Property Graphs as RDF in Oracle*; EDBT: Konstanz, Germany, 2010; pp. 762–773.
145. Chiba, H.; Yamanaka, R.; Keswani, Y.; Matsumoto, S. G2GML: Graph to Graph Mapping Language for Bridging RDF and Property Graphs. In Proceedings of the International Semantic Web Conference, Online, 1–6 November 2020; Springer: Berlin, Germany, 2020; pp. 160–175.
146. Matsumoto, S.; Yamanaka, R.; Chiba, H. Mapping RDF graphs to property graphs. *arXiv* **2018**, arXiv:1812.01801.
147. Angles, R.; Thakkar, H.; Tomaszuk, D. Mapping rdf databases to property graph databases. *IEEE Access* **2020**, *8*, 86091–86110. [CrossRef]
148. Tomaszuk, D.; Angles, R.; Thakkar, H. PGO: Describing Property Graphs in RDF. *IEEE Access* **2020**, *8*, 118355–118369. [CrossRef]
149. Schätzle, A.; Przyjaciel-Zablocki, M.; Berberich, T.; Lausen, G. S2X: Graph-parallel querying of RDF with GraphX. In *Biomedical Data Management and Graph Online Querying*; Springer: Berlin, Germany, 2015; pp. 155–168.
150. Haihong, E.; Han, P.; Song, M. Transforming RDF to Property Graph in Hugegraph. In Proceedings of the 6th International Conference on Engineering and MIS 2020, Almaty, Kazakhstan, 14–16 September 2020; ICEMIS'20.
151. Zhang, R.; Liu, P.; Guo, X.; Li, S.; Wang, X. A unified relational storage scheme for RDF and property graphs. In *Proceedings of the International Conference on Web Information Systems and Applications*; Springer: Berlin, Germany, 2019; pp. 418–429.
152. Angles, R.; Thakkar, H.; Tomaszuk, D. *RDF and Property Graphs Interoperability: Status and Issues*; AMW: Cartagena de Indias, Colombia, 2019.
153. Khayatbashi, S.; Ferrada, S.; Hartig, O. *Converting Property Graphs to RDF: A Preliminary Study of the Practical Impact of Different Mappings*; GRADES-NDA@ SIGMOD: New York, NY, USA, 2022; pp. 10–11.
154. Hartig, O. Reconciliation of RDF* and property graphs. *arXiv* **2014**, arXiv:1409.3288.
155. Nguyen, V.; Yip, H.Y.; Thakkar, H.; Li, Q.; Bolton, E.; Bodenreider, O. Singleton Property Graph: Adding A Semantic Web Abstraction Layer to Graph Databases. *BlockSW/CKG@ ISWC* **2019**, *2599*, 1–13.
156. Modoni, G.; Sacco, M.; Terkaj, W. A survey of RDF store solutions. In Proceedings of the 2014 International Conference on Engineering, Technology and Innovation (ICE), Bergamo, Italy, 23–25 June 2014; pp. 1–7.
157. Muhamad, W.; Suhardi; Bandung, Y. Transforming OpenAPI Specification 3.0 documents into RDF-based semantic web services. *J. Big Data* **2022**, *9*, 55. [CrossRef]
158. Ferilli, S.; Redavid, D. The GraphBRAIN System for Knowledge Graph Management and Advanced Fruition. In *International Symposium on Methodologies for Intelligent Systems*; Springer: Berlin, Germany, 2020; pp. 308–317.
159. Nguyen, V.; Bodenreider, O.; Sheth, A. Don't like RDF reification? Making statements about statements using singleton property. In Proceedings of the 23rd International Conference on World WIDE Web, Seoul, Korea, 7–11 April 2014; pp. 759–770.
160. Motik, B. On the properties of metamodeling in OWL. *J. Log. Comput.* **2007**, *17*, 617–637. [CrossRef]
161. Horrocks, I. Owl: A description logic based ontology language. In Proceedings of the International Conference on Principles and Practice of Constraint Programming, Sitges, Spain, 1–5 October 2005; Springer: Berlin, Germany, 2005; pp. 5–8.
162. Rodriguez-Muro, M.; Rezk, M. Efficient SPARQL-to-SQL with R2RML mappings. *J. Web Semant.* **2015**, *33*, 141–169. [CrossRef]
163. Khamparia, A.; Pandey, B. Comprehensive analysis of semantic web reasoners and tools: A survey. *Educ. Inf. Technol.* **2017**, *22*, 3121–3145. [CrossRef]
164. Di Pierro, D.; Redavid, D.; Ferilli, S. Linking Graph Databases and Semantic Web for Reasoning in Library Domains. In Proceedings of the IRCDL 2022: Italian Research Conference on Digital Libraries, Padova, Italy, 24–25 February 2022.
165. Ferilli, S. GEAR: A General Inference Engine for Automated MultiStrategy Reasoning. *Electronics* **2023**, *12*, 256. [CrossRef]
166. Minker, J. On indefinite databases and the closed world assumption. In Proceedings of the International Conference on Automated Deduction, New York, NY, USA, 7–9 June 1982; Springer: Berlin, Germany, 1982; pp. 292–308.
167. Moore, P.; Van Pham, H.H. On context and the open world assumption. In Proceedings of the 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, Gwangju, Korea, 24–27 March 2015; pp. 387–392.
168. Mitchell, M. Abstraction and analogy-making in artificial intelligence. *Ann. N. Y. Acad. Sci.* **2021**, *1505*, 79–101. [CrossRef] [PubMed]
169. Evans, J. Logic and human reasoning: An assessment of the deduction paradigm. *Psychol. Bull.* **2002**, *128*, 978. [CrossRef] [PubMed]
170. Carnap, R.; Jeffrey, R.C. *Studies in Inductive Logic and Probability*; University of California Press: Oakland, CA, USA, 1980; Volume 2.
171. Hawthorne, J. Inductive Logic. In *The Stanford Encyclopedia of Philosophy*; Zalta, E.N., Ed.; The Metaphysics Research Lab: Stanford, CA, USA, 2011.
172. Svennevig, J. Abduction as a Methodological Approach to the Study of Spoken Interaction. 2001. Available online: https://www.researchgate.net/publication/251398301_Abduction_as_a_methodological_approach_to_the_study_of_spoken_interaction (accessed on 21 January 2023).

173. Dung, P.M. An argumentation-theoretic foundation for logic programming. *J. Log. Program.* **1995**, *22*, 151–177. [CrossRef]
174. Michalski, R.S. Inferential theory of learning as a conceptual basis for multistrategy learning. *Mach. Learn.* **1993**, *11*, 111–151. [CrossRef]
175. Figuera, M.; Rohde, P.D.; Vidal, M. Trav-SHACL: Efficiently validating networks of SHACL constraints. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 3337–3348.
176. Pareti, P.; Konstantinidis, G. A Review of SHACL: From Data Validation to Schema Reasoning for RDF Graphs. *arXiv* **2021**, arXiv:2112.01441.
177. Pareti, P.; Konstantinidis, G.; Mogavero, F.; Norman, T.J. SHACL satisfiability and containment. In Proceedings of the International Semantic Web Conference, Online, 1–6 November 2020; Springer: Berlin, Germany, 2020; pp. 474–493.
178. Staworko, S.; Boneva, I.; Gayo, J.L.E.; Hym, S.; Prud'Hommeaux, E.G.; Solbrig, H. Complexity and Expressiveness of ShEx for RDF. In Proceedings of the 18th International Conference on Database Theory (ICDT 2015), Brussels, Belgium, 23–27 March 2015.
179. Boneva, I.; Dusart, J.; Alvarez, D.F.; Gayo, J.E.L. Shape designer for ShEx and SHACL constraints. In Proceedings of the ISWC 2019-18th International Semantic Web Conference, Auckland, New Zealand, 26–30 October 2019.
180. Wright, J.; Méndez, S.J.R.; Haller, A.; Taylor, K.; Omran, P.G. Schímatos: A SHACL-based web-form generator for knowledge graph editing. In Proceedings of the International Semantic Web Conference, Online, 1–6 November 2020; Springer: Berlin, Germany, 2020; pp. 65–80.
181. Arndt, N.; Valdestilhas, A.; Publio, G.; Arriaga, A.C.; Höffner, K.; Riechert, T. *A Visual SHACL Shapes Editor Based on OntoPad*; SEMANTiCS Posters&Demos, CEUR-WS: Aachen, Germany, 2021.
182. Ekaputra, F.J.; Lin, X. SHACL4P: SHACL constraints validation within Protégé ontology editor. In Proceedings of the 2016 International Conference on Data and Software Engineering (ICoDSE), Denpasar, Indonesia, 26–27 October 2016; pp. 1–6.
183. Senthilvel, M.; Beetz, J.; Computation, D. *A Visual Programming Approach for Validating Linked Building Data*; Universitätsbibliothek der RWTH Aachen: Aachen, Germany, 2020.
184. Alom, H. A Library for Visualizing SHACL over Knowledge Graphs. Master's Thesis, Gottfried Wilhelm Leibniz Universität Hannover, Hannover, Germany, 2022.
185. Echegaray, D. Making a Common Graphical Language for the Validation of Linked Data. 2017. Available online: https://kth.diva-portal.org/smash/get/diva2:1127525/FULLTEXT01.pdf (accessed on 21 January 2023).