# Knowledge Graphs

AIDAN HOGAN, IMFD, DCC, Universidad de Chile, Chile
EVA BLOMQVIST, Linköping University, Sweden
MICHAEL COCHEZ, Vrije Universiteit, The Netherlands
CLAUDIA D'AMATO, University of Bari, Italy
GERARD DE MELO, Rutgers University, USA
CLAUDIO GUTIERREZ, IMFD, DCC, Universidad de Chile, Chile
JOSÉ EMILIO LABRA GAYO, Universidad de Oviedo, Spain
SABRINA KIRRANE, SEBASTIAN NEUMAIER, and AXEL POLLERES, WU Vienna, Austria
ROBERTO NAVIGLI, Sapienza University of Rome, Italy
AXEL-CYRILLE NGONGA NGOMO, DICE, Universität Paderborn, Germany
SABBIR M. RASHID, Tetherless World Constellation, Rensselaer Polytechnic Institute, USA
ANISA RULA, University of Milano-Bicocca, Italy and University of Bonn, Germany
LUKAS SCHMELZEISEN, WeST, Universität Koblenz–Landau, Germany
JUAN SEQUEDA, data.world, USA
STEFFEN STAAB, Universität Stuttgart, Germany and University of Southampton, UK
ANTOINE ZIMMERMANN, École des mines de Saint-Étienne, France

In this paper we provide a comprehensive introduction to knowledge graphs, which have recently garnered significant attention from both industry and academia in scenarios that require exploiting diverse, dynamic, large-scale collections of data. After a general introduction, we motivate and contrast various graph-based data models and query languages that are used for knowledge graphs. We discuss the roles of schema, identity, and context in knowledge graphs. We explain how knowledge can be represented and extracted using a combination of deductive and inductive techniques. We summarise methods for the creation, enrichment, quality assessment, refinement, and publication of knowledge graphs. We provide an overview of prominent open knowledge graphs and enterprise knowledge graphs, their applications, and how they use the aforementioned techniques. We conclude with high-level future research directions for knowledge graphs.

## 1 INTRODUCTION

Though the phrase "knowledge graph" has been used in the literature since at least 1972 [440], the modern incarnation of the phrase stems from the 2012 announcement of the Google Knowledge Graph [459], followed by further announcements of the development of knowledge graphs by Airbnb [83], Amazon [280], eBay [392], Facebook [365], IBM [123], LinkedIn [214], Microsoft [457], Uber [205], and more besides. The growing industrial uptake of the concept proved difficult for academia to ignore: more and more scientific literature is being published on knowledge graphs, which includes books (e.g. [400]), as well as papers outlining definitions (e.g., [136]), novel techniques (e.g., [298, 399, 521]), and surveys of specific aspects of knowledge graphs (e.g., [375, 519]).

Underlying all such developments is the core idea of using graphs to represent data, often enhanced with some way to explicitly represent knowledge [365]. The result is most often used in application scenarios that involve integrating, managing and extracting value from diverse sources of data at large scale [365]. Employing a graph-based abstraction of knowledge has a number benefits in such settings when compared with, for example, a relational model or NoSQL alternatives. Graphs provide a concise and intuitive abstraction for a variety of domains, where edges capture the (potentially cyclical) relations between the entities inherent in social data, biological

interactions, bibliographical citations and co-authorships, transport networks, and so forth [15]. Graphs allow maintainers to postpone the definition of a schema, allowing the data – and its scope – to evolve in a more flexible manner than typically possible in a relational setting, particularly for capturing incomplete knowledge [2]. Unlike (other) NoSQL models, specialised graph query languages support not only standard relational operators (joins, unions, projections, etc.), but also navigational operators for recursively finding entities connected through arbitrary-length paths [14]. Standard knowledge representation formalisms – such as ontologies [66, 228, 344] and rules [242, 270] – can be employed to define and reason about the semantics of the terms used to label and describe the nodes and edges in the graph. Scalable frameworks for graph analytics [314, 478, 529] can be leveraged for computing centrality, clustering, summarisation, etc., in order to gain insights about the domain being described. Various representations have also been developed that support applying machine learning techniques directly over graphs [519, 527].

In summary, the decision to build and use a knowledge graph opens up a range of techniques than can be brought to bear for integrating and extracting value from diverse sources of data. However, we have yet to see a general unifying summary that describes how knowledge graphs are being used, what techniques they employ, and how they relate to existing data management topics.

The goal of this tutorial paper is to motivate and give a comprehensive introduction to knowledge graphs: to describe their foundational data models and how they can be queried; to discuss representations relating to schema, identity, and context; to discuss deductive and inductive ways to make knowledge explicit; to present a variety of techniques that can be used for the creation and enrichment of graph-structured data; to describe how the quality of knowledge graphs can be discerned and how they can be refined; to discuss standards and best practices by which knowledge graphs can be published; and to provide an overview of existing knowledge graphs found in practice. Our intended audience includes researchers and practitioners who are new to knowledge graphs. As such, we do not assume that readers have specific expertise on knowledge graphs.

*Knowledge graph.* The definition of a "*knowledge graph*" remains contentious [36, 53, 136], where a number of (sometimes conflicting) definitions have emerged, varying from specific technical proposals to more inclusive general proposals; we address these prior definitions in Appendix A. Herein we adopt an inclusive definition, where we view a knowledge graph as *a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities*. The graph of data (aka *data graph*) conforms to a graph-based data model, which may be a *directed edge-labelled graph*, a *property graph*, etc. (we discuss concrete alternatives in Section 2). By *knowledge*, we refer to something that is *known*[1]. Such knowledge may be accumulated from external sources, or extracted from the knowledge graph itself. Knowledge may be composed of simple statements, such as "*Santiago is the capital of Chile*", or quantified statements, such as "*all capitals are cities*". Simple statements can be accumulated as edges in the data graph. If the knowledge graph intends to accumulate quantified statements, a more expressive way to represent knowledge – such as *ontologies* or *rules* – is required. *Deductive methods* can then be used to entail and accumulate further knowledge (e.g., "*Santiago is a city*"). Additional knowledge – based on simple or quantified statements – can also be extracted from and accumulated by the knowledge graph using *inductive methods*.

Knowledge graphs are often assembled from numerous sources, and as a result, can be highly diverse in terms of structure and granularity. To address this diversity, representations of *schema*, *identity*, and *context* often play a key role, where a *schema* defines a high-level structure for the knowledge graph, *identity* denotes which nodes in the graph (or in external sources) refer to the same real-world entity, while *context* may indicate a specific setting in which some unit of

---

[1]A number of specific definitions for knowledge have been proposed in the literature on epistemology.

knowledge is held true. As aforementioned, effective methods for *extraction*, *enrichment*, *quality assessment*, and *refinement* are required for a knowledge graph to grow and improve over time.

*In practice.* Knowledge graphs aim to serve as an ever-evolving shared substrate of knowledge within an organisation or community [365]. We distinguish two types of knowledge graphs in practice: *open knowledge graphs* and *enterprise knowledge graphs*. Open knowledge graphs are published online, making their content accessible for the public good. The most prominent examples – DBpedia [291], Freebase [51], Wikidata [515], YAGO [232], etc. – cover many domains and are either extracted from Wikipedia [232, 291], or built by communities of volunteers [51, 515]. Open knowledge graphs have also been published within specific domains, such as media [406], government [222, 450], geography [472], tourism [11, 263, 308, 540], life sciences [79], and more besides. Enterprise knowledge graphs are typically internal to a company and applied for commercial use-cases [365]. Prominent industries using enterprise knowledge graphs include Web search (e.g., Bing [457], Google [459]), commerce (e.g., Airbnb [83], Amazon [127, 280], eBay [392], Uber [205]), social networks (e.g., Facebook [365], LinkedIn [214]), finance (e.g., Accenture [368], Banca d'Italia [32], Bloomberg [326], Capital One [65], Wells Fargo [355]), among others. Applications include search [457, 459], recommendations [83, 205, 214, 365], personal agents [392], advertising [214], business analytics [214], risk assessment [107, 495], automation [223], and more besides. We will provide more details on the use of knowledge graphs in practice in Section 10.

*Running example.* To keep the discussion accessible, throughout the paper, we present concrete examples in the context of a hypothetical knowledge graph relating to tourism in Chile (loosely inspired by, e.g., [263, 308]). The knowledge graph is managed by a tourism board that aims to increase tourism in the country and promote new attractions in strategic areas. The knowledge graph itself will eventually describe tourist attractions, cultural events, services, and businesses, as well as cities and inter-city travel routes. Some applications the organisation envisages are to:

- create a tourism portal that allows visitors to search for attractions, upcoming events, and other related services (in multiple languages);
- gain insights into tourism demographics in terms of season, nationalities, etc.;
- analyse sentiment about various attractions and events, including positive reviews, summaries of complaints about events and services, reports of crime, etc.;
- understand tourism trajectories: the sequence of attractions, events, etc., that tourists visit;
- cross-reference trajectories with available flights/buses to suggest new strategic routes;
- offer personalised recommendations of places to visit;
- and so forth.

*Structure.* The remainder of the paper is structured as follows:

**Section 2** outlines graph data models and the languages that can be used to query them.
**Section 3** describes representations of schema, identity, and context in knowledge graphs.
**Section 4** presents deductive formalisms by which knowledge can be represented and entailed.
**Section 5** describes inductive techniques by which additional knowledge can be extracted.
**Section 6** discusses the creation and enrichment of knowledge graphs from external sources.
**Section 7** enumerates quality dimensions by which a knowledge graph can be assessed.
**Section 8** discusses various techniques for knowledge graph refinement.
**Section 9** discusses principles and protocols for publishing knowledge graphs.
**Section 10** surveys some prominent knowledge graphs and their applications.
**Section 11** concludes with a summary and future research directions for knowledge graphs.
**Appendix A** provides historical background and previous definitions for knowledge graphs.
**Appendix B** enumerates formal definitions that will be referred to from the body of the paper.

## 2 DATA GRAPHS

At the foundation of any knowledge graph is the principle of first applying a graph abstraction to data, resulting in an initial data graph. We now discuss a selection of graph-structured data models that are commonly used in practice to represent data graphs. We then discuss the primitives that form the basis of graph query languages used to interrogate such data graphs.

### 2.1 Models

Leaving aside graphs, let us assume that the tourism board from our running example has not yet decided how to model relevant data about attractions, events, services, etc. The board first considers using a tabular structure – in particular, relational databases – to represent the required data, and though they do not know precisely what data they will need to capture, they start to design an initial relational schema. They begin with an Event table with five columns:

$$\text{Event}(\underline{\text{name}}, \text{venue}, \text{type}, \underline{\text{start}}, \text{end})$$

where name and start together form the primary key of the table in order to uniquely identify recurring events. But as they start to populate the data, they encounter various issues: events may have multiple names (e.g., in different languages), events may have multiple venues, they may not yet know the start and end date-times for future events, events may have multiple types, and so forth. Incrementally addressing these modelling issues as the data become more diverse, they generate internal identifiers for events and adapt their relational schema until they have:

$$\text{EventName}(\underline{\text{id}}, \underline{\text{name}}), \text{EventStart}(\underline{\text{id}}, \text{start}), \text{EventEnd}(\underline{\text{id}}, \text{end}), \tag{1}$$
$$\text{EventVenue}(\underline{\text{id}}, \underline{\text{venue}}), \text{EventType}(\underline{\text{id}}, \underline{\text{type}})$$

With the above schema, the organisation can now model events with $0-n$ names, venues, and types, and $0-1$ start dates and end dates (without needing relational nulls/blank cells in tables).

Along the way, the board has to incrementally change the schema several times in order to support new sources of data. Each such change requires a costly remodelling, reloading, and reindexing of data; here we only considered one table. The tourism board struggles with the relational model because they do not know, *a priori*, what data will need to be modelled or what sources they will use. But once they reach the latter relational schema, the board finds that they can integrate further sources without more changes: with minimal assumptions on *multiplicities* ($1-1$, $1-n$, etc.) this schema offers a lot of flexibility for integrating incomplete and diverse data.

In fact, the refined, flexible schema that the board ends up with – shown in (1) – is modelling a set of binary relations between entities, which indeed can be viewed as modelling a graph. By instead adopting a graph data model from the outset, the board could forgo the need for an upfront schema, and could define any (binary) relation between any pair of entities at any time.

We now introduce three graph data models commonly used in practice [14].

*2.1.1 Directed edge-labelled graphs.* A directed edge-labelled graph is defined as a set of nodes – like Santiago, Arica, EID16, 2018-03-22 12:00 – and a set of directed labelled edges between those nodes, like Santa Lucía —city→ Santiago. In the case of knowledge graphs, nodes are used to represent entities and edges are used to represent (binary) relations between those entities. Figure 1 provides an example of how the tourism board could model some relevant event data as a directed edge-labelled graph (for a formal definition of a directed edge-labelled graph see Definition B.1 in Appendix B). The graph includes data about the names, types, start and end date-times, and venues for events.[2] Adding information to such a graph typically involves adding new nodes and edges (with some exceptions

---

[2]We represent bidirectional edges as Viña del Mar ←bus→ Arica, which more concisely depicts two directed edges: Viña del Mar —bus→ Arica and Viña del Mar ←bus— Arica. Also while some naming conventions recommend more
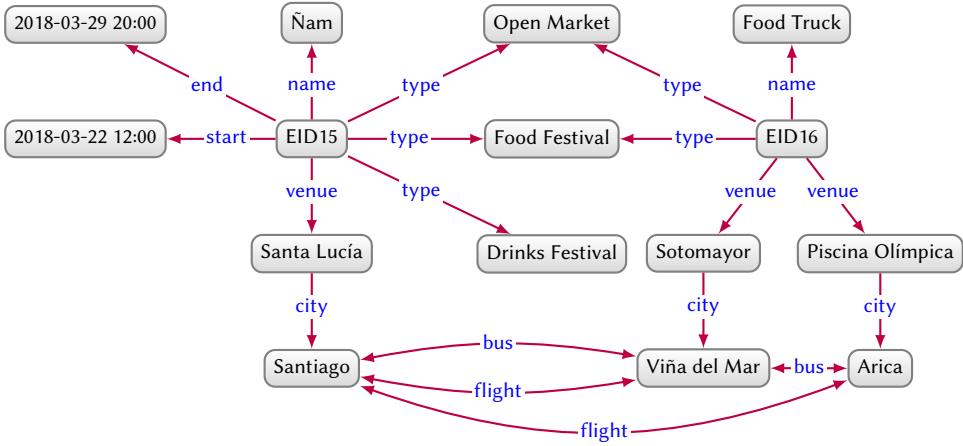
Fig. 1. Directed edge-labelled graph describing events and their venues.

discussed later). Representing incomplete information requires simply omitting a particular edge; for example, the graph does not yet define a start/end date-time for the Food Truck festival.

Modelling data as a graph in this way offers more flexibility for integrating new sources of data, compared to the standard relational model, where a schema must be defined upfront and followed at each step. While other structured data models such as trees (XML, JSON, etc.) would offer similar flexibility, graphs do not require organising the data hierarchically (should venue be a parent, child, or sibling of type for example?). They also allow cycles to be represented and queried (e.g., note the directed cycle in the routes between Santiago, Arica, and Viña del Mar).

A standardised data model based on directed edge-labelled graphs is the Resource Description Framework (RDF) [106], which has been recommended by the W3C. The RDF model defines different types of nodes, including *Internationalized Resource Identifiers* (IRIs) [129] which allow for global identification of entities on the Web; *literals*, which allow for representing strings (with or without language tags) and other datatype values (integers, dates, etc.); and *blank nodes*, which are anonymous nodes that are not assigned an identifier (for example, rather than create internal identifiers like EID15, EID16, in RDF, we have the option to use blank nodes). We will discuss these different types of nodes further in Section 3.2 when we speak about issues relating to identity.

*2.1.2 Graph dataset.* Although multiple directed edge-labelled graphs can be merged by taking their union, it is often desirable to manage several graphs rather than one monolithic graph; for example, it may be beneficial to manage multiple graphs from different sources, making it possible to update or refine data from one source, to distinguish untrustworthy sources from more trustworthy ones, and so forth. A graph dataset then consists of a set of *named graphs* and a *default graph*. Each named graph is a pair of a graph ID and a graph. The default graph is a graph without an ID, and is referenced "by default" if a graph ID is not specified. Figure 2 provides an example where events and routes are stored in two named graphs, and the default graph manages meta-data about the named graphs (for a formal definition of a graph dataset, see Definition B.3 in Appendix B). We highlight that graph names can also be used as nodes in a graph. Furthermore, nodes and edges can be repeated across graphs, where the same node in different graphs will typically refer to the same entity, allowing data on that entity to be integrated when merging graphs.

---

complete edge labels that include a verb, such as has venue or is valid from, in this paper, for presentation purposes, we will omit the "has" and "is" verbs from such labels, using simply venue or valid from.
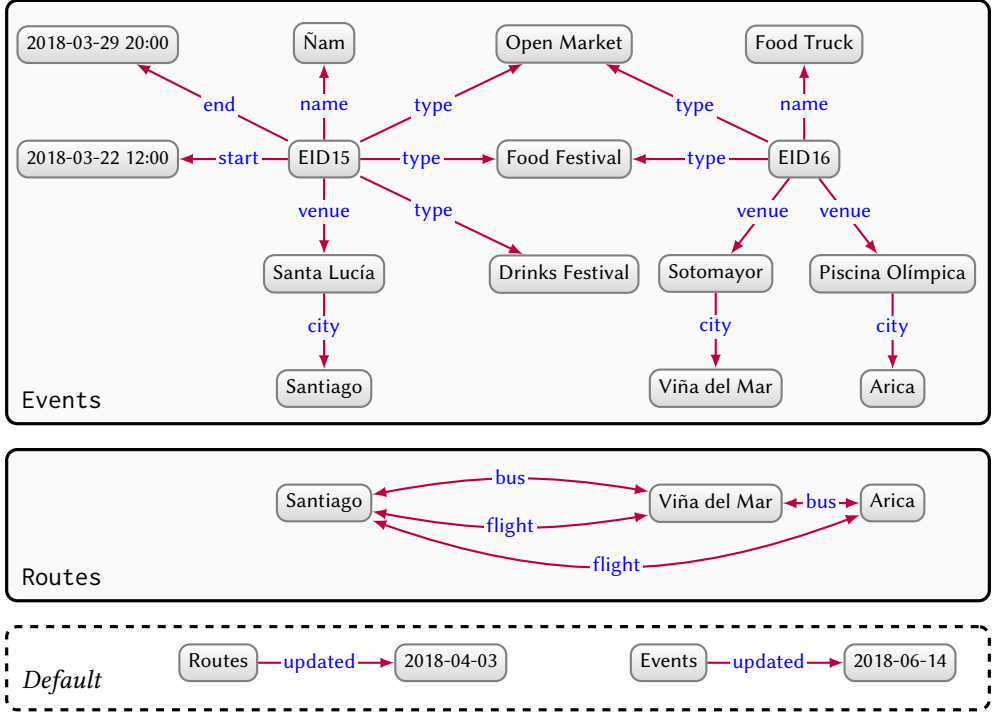
Fig. 2. Graph dataset with two named graphs and a default graph describing events and routes

A prominent use-case for graph datasets is to manage and query *Linked Data* composed of interlinked documents of RDF graphs spanning the Web. When dealing with Web data, tracking the source of data becomes of key importance [54, 125, 546]. We will discuss Linked Data later in Section 3.2 and further discuss provenance in Section 3.3.

*2.1.3 Property graphs.* Property graphs were introduced to provide additional flexibility when modelling more complex relations. Consider integrating incoming data that provides information on which companies offer fares on which flights, allowing the board to better understand available routes between cities (for example, on national airlines). In the case of directed-edge labelled graphs, we cannot directly annotate an edge like (Santiago)—flight→(Arica) with the company (or companies) offering that route. But we could add a new node denoting a flight, connect it with the source, destination, companies, and mode, as shown in Figure 3. Applying this modelling to all of the data in Figure 1 would, however, involve a significant change to the graph. Another option might be to put the flights of different companies in different named graphs, but if named graphs are already being used to track the source of graphs (for example), this could become cumbersome.

The property graph model was thus proposed to offer additional flexibility when modelling data as a graph [14, 332]. A property graph allows a set of *property–value* pairs and a *label* to be associated with both nodes and edges. Figure 4 provides an example, again showing flights between Santiago and Arica and the companies offering those routes (for a formal definition of a property graph, we refer to Definition B.5 in Appendix B). This time we use property–value pairs on edges to
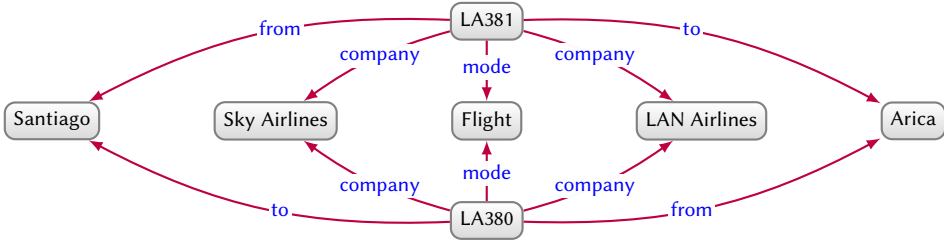
Fig. 3. Directed edge-labelled graph with companies offering flights between Santiago and Arica
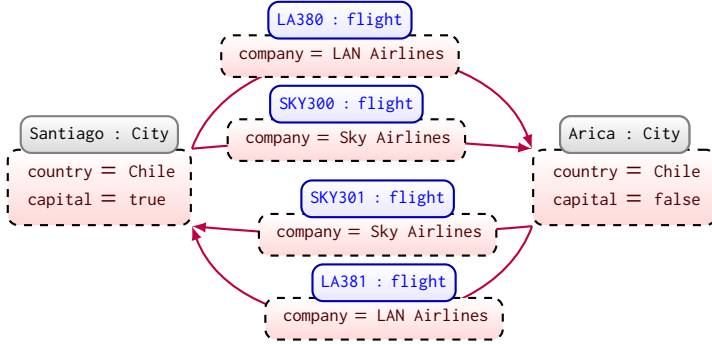


Fig. 4. Property graph with companies offering flights between Santiago and Arica

model the companies[3], as well as the distance. The type of relation is captured by the label `flight`. We further model that Santiago and Arica are cities using a node label, as well as their country and whether or not they are the capital using property–value pairs on the nodes.

Property graphs are most prominently used in popular graph databases, such as Neo4j [14, 332]. In choosing between graph models, it is important to note that property graphs can be translated to/from directed edge-labelled graphs and/or graph datasets without loss of information [16, 224] (per, e.g., Figure 4). In summary, directed-edge labelled graphs offer a more minimal model, while property graphs offer a more flexible one. Often the choice of model will be secondary to other practical factors, such as the implementations available for different models, etc.

*2.1.4 Other graph data models.* The previous three models are popular examples of graph representations. Other graph data models exist with *complex nodes* that may contain individual edges [15, 212] or nested graphs [15, 38] (sometimes called *hypernodes* [295]). Likewise the mathematical notion of a *hypergraph* defines *complex edges* that connect sets rather than pairs of nodes. In our view, a knowledge graph can adopt any such graph data model based on nodes and edges: often data can be converted from one model to another (see Figure 3 vs. Figure 4). In recent years, the most popular graph models in practice have become directed-edge labelled graphs, graph datasets, and property graphs [14]. In the rest of the paper, we prefer discussing directed-edge labelled graphs given their relative succinctness, but most discussion extends naturally to other models.

---

[3]In practical implementations of property graphs, properties with multiple values may be expressed, for example, as a single array value. Such issues do not, however, affect expressivity, nor our discussion.
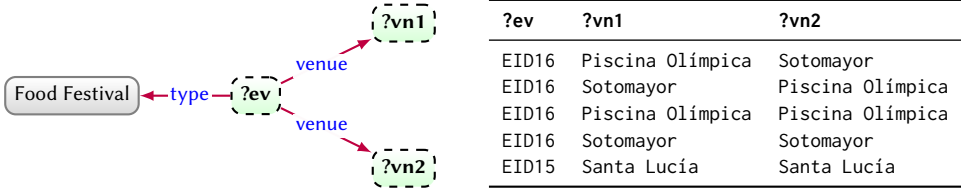
Fig. 5. Graph pattern (left) with mappings generated over the graph of Figure 1 (right)

| ?ev | ?vn1 | ?vn2 |
|-----|------|------|
| EID16 | Piscina Olímpica | Sotomayor |
| EID16 | Sotomayor | Piscina Olímpica |
| EID16 | Piscina Olímpica | Piscina Olímpica |
| EID16 | Sotomayor | Sotomayor |
| EID15 | Santa Lucía | Santa Lucía |

## 2.2 Querying

A number of practical languages have been proposed for querying graphs [14], including the SPARQL query language for RDF graphs [207]; and Cypher [159], Gremlin [420], and G-CORE [13] for querying property graphs.[4] Underlying these query languages are some common primitives, including (basic) graph patterns, relational operators, path expressions, and more besides [14]. We now describe these core features for querying graphs in turn, starting with graph patterns.

*2.2.1 Graph patterns.* At the core of every structured query language for graphs are (*basic*) *graph patterns* [14, 96], which follow the same model as the data graph being queried (see Section 2.1), additionally allowing variables as terms.[5] Terms in graph patterns are thus divided into constants, such as (Arica) or venue, and variables, which we prefix with question marks, such as ?event or ?rel. A graph pattern is then evaluated against the data graph by generating mappings from the variables of the graph pattern to constants in the data graph such that the image of the graph pattern under the mapping (replacing variables with the assigned constants) is contained within the data graph.

In Figure 5, we provide an example of a graph pattern looking for the venues of Food Festivals, along with the possible mappings generated by the graph pattern against the data graph of Figure 1. In some of the presented mappings (the last two listed), multiple variables are mapped to the same term, which may or may not be desirable depending on the application. Hence a number of semantics have been proposed for evaluating graph patterns [14], amongst which the most important are: *homomorphism-based semantics*, which allows multiple variables to be mapped to the same term such that all mappings shown in Figure 5 would be considered results; and *isomorphism-based semantics*, which requires variables on nodes and/or edges to be mapped to unique terms, thus excluding the latter three mappings of Figure 5 from the results. Different practical languages adopt different semantics for evaluating graph patterns where, for example, SPARQL adopts a homomorphism-based semantics, while Cypher adopts an isomorphism-based semantics on edges.

As we will see in later examples (particularly Figure 7), graph patterns may also form cycles (be they directed or undirected), and may replace edge labels with variables. Graph patterns in the context of other models – such as property graphs – can be defined analogously by allowing variables to replace terms in any position of the model. We provide a formalisation of graph patterns and their evaluation for both directed edge-labelled graphs and property graphs in Appendix B.2.1.

*2.2.2 Complex graph patterns.* A graph pattern transforms an input graph into a table of results (as shown in Figure 5). We may then consider using the relational algebra to combine and/or transform such tables, thus forming more complex queries from one or more graph patterns. Recall that the relational algebra consists of unary operators that accept one input table, and binary operators that accept two input tables. Unary operators include projection ($\pi$) to output a subset of columns,

---

[4]The popularity of these languages is investigated by Seifer et al. [445].

[5]The terms of a directed edge-labelled graph are its nodes and edge-labels. The terms of a property graph are its ids, labels, properties, and values (as used on either edges or nodes).
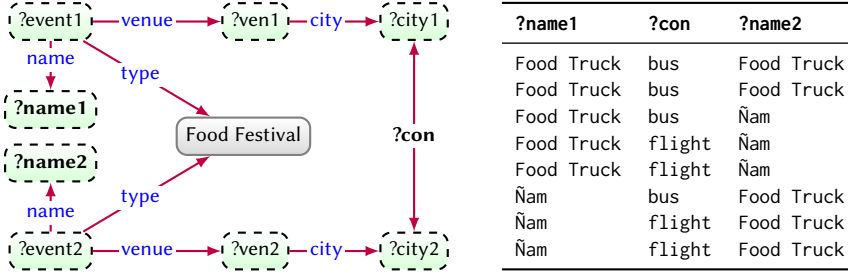
| ?name1 | ?con | ?name2 |
|---|---|---|
| Food Truck | bus | Food Truck |
| Food Truck | bus | Food Truck |
| Food Truck | bus | Ñam |
| Food Truck | flight | Ñam |
| Food Truck | flight | Ñam |
| Ñam | bus | Food Truck |
| Ñam | flight | Food Truck |
| Ñam | flight | Food Truck |

Fig. 6. Conjunctive query (left) with mappings generated over the graph of Figure 1 (right)

selection ($\sigma$) to output a subset of rows matching a given condition, and renaming of columns ($\rho$). Binary operators include union ($\cup$) to merge the rows of two tables into one table, difference ($-$) to remove the rows from the first table present in the second table, and joins ($\bowtie$) to extend the rows of one table with rows from the other table that satisfy a join condition. Selection and join conditions typically include equalities ($=$), inequalities ($\leq$), negation ($\neg$), disjunction ($\vee$), etc. From these operators, we can further define other (syntactic) operators, such as intersection ($\cap$) to output rows in both tables, anti-join ($\rhd$, aka *not exists*) to output rows from the first table for which there are no join-compatible rows in the second table, left-join ($⟕$, aka *optional*) to perform a join but keeping rows from the first table without a compatible row in the second table, etc.

Graph patterns can then be expressed in a subset of relational algebra (namely $\pi$, $\sigma$, $\rho$, $\bowtie$). Assuming, for example, a single ternary relation $G(s, p, o)$ representing a graph – i.e., a table $G$ with three columns $s$, $p$, $o$ – the query of Figure 5 can be expressed in relational algebra as:

$$\pi_{ev, vn1, vn2}(\sigma_{p=\text{type} \wedge o=\text{Food Festival} \wedge p_1=p_2=\text{venue}}(\rho_{s/ev}(G \bowtie \rho_{p/p_1, o/vn1}(G) \bowtie \rho_{p/p_2, o/vn2}(G))))$$

where $\bowtie$ denotes a *natural join*, meaning that equality is checked across pairs of columns with the same name in both tables (here, the join is thus performed on the subject column $s$). The result of this query is a table with a column for each variable: $ev, vn1, vn2$. However, not all queries using $\pi$, $\sigma$, $\rho$ and $\bowtie$ on $G$ can be expressed as graph patterns; for example, we cannot choose which variables to project in a graph pattern, but rather must project all variables not fixed to a constant.

Graph query languages such as SPARQL [207] and Cypher [159] allow the full use of relational operators over the results of graph patterns, giving rise to *complex graph patterns* [14]. Figure 6 presents an example of a complex graph pattern with projected variables in bold, choosing particular variables to appear in the final results. In terms of expressivity, graph patterns with (unrestricted) projection of this form equate to *conjunctive queries* on graphs. In Figure 7, we give another example of a complex graph pattern looking for food festivals or drinks festivals not held in Santiago, optionally returning their name and start date (where available). Such queries – allowing the full use of relational operators on top of graph patterns – equate to *first-order queries* on graphs. In Appendix B.2.2, we formalise complex graph patterns and their evaluation over data graphs.

Complex graph patterns can give rise to duplicate results; for example, the first result in Figure 6 appears twice since ?city1 matches Arica and ?city2 matches Viña del Mar in one result, and vice-versa in the other. Query languages then offer two semantics: *bag semantics* preserves duplicates according to the multiplicity of the underlying mappings, while *set semantics* (typically invoked with a DISTINCT keyword) removes duplicates from the results.

*2.2.3 Navigational graph patterns.* A key feature that distinguishes graph query languages is the ability to include *path expressions* in queries. A path expression $r$ is a regular expression that allows matching arbitrary-length paths between two nodes, which is expressed as a *regular path*
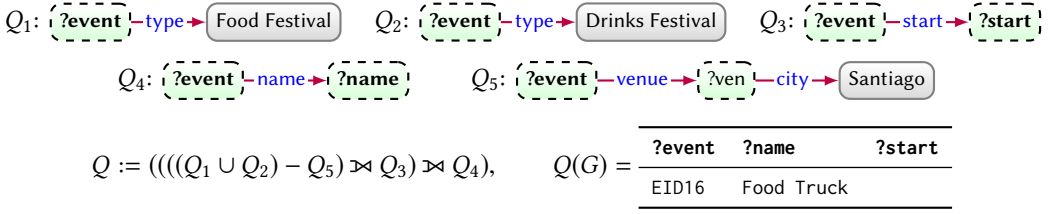
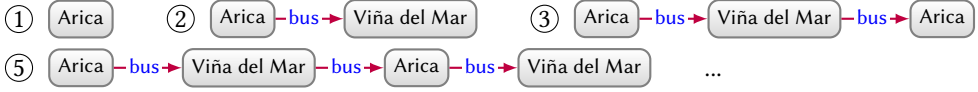Fig. 7. Complex graph pattern ($Q$) with mappings generated over the graph of Figure 1



Fig. 8. Some possible paths matching $(\text{Arica}, \text{bus}*, ?\text{city})$ over the graph of Figure 1 ($Q(G)$)
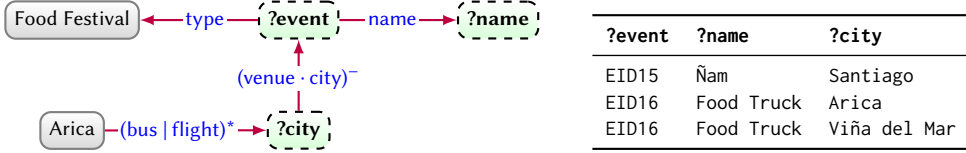


Fig. 9. Navigational graph pattern (left) with mappings generated over the graph of Figure 1 (right)

query $(x, r, y)$, where $x$ and $y$ can be variables or constants (or even the same term). The base path expression is where $r$ is a constant (an edge label). Furthermore if $r$ is a path expression, then $r^-$ (inverse)[6] and $r^*$ (Kleene star: zero-or-more) are also path expressions. Finally, if $r_1$ and $r_2$ are path expressions, then $r_1 \mid r_2$ (disjunction) and $r_1 \cdot r_2$ (concatenation) are also path expressions.

Regular path queries can then be evaluated under a number of different semantics. For example, $(\text{Arica}, \text{bus}*, ?\text{city})$ evaluated against the graph of Figure 1 may match the paths in Figure 8. In fact, since a cycle is present, an infinite number of paths are potentially matched. For this reason, restricted semantics are often applied, returning only the shortest paths, or paths without repeated nodes or edges (as in the case of Cypher).[7] Rather than returning paths, another option is to instead return the (finite) set of pairs of nodes connected by a matching path (as in the case of SPARQL 1.1).

Regular path queries can then be used in graph patterns to express *navigational graph patterns* [14], as shown in Figure 9, which illustrates a query searching for food festivals in cities reachable (recursively) from Arica by bus or flight. Furthermore, when regular path queries and graph patterns are combined with operators such as projection, selection, union, difference, and optional, the result is known as *complex navigational graph patterns* [14]. Appendix B.2.3 provides definitions for (complex) navigational graph patterns and their evaluation.

*2.2.4 Other features.* Thus far we have discussed features that form the practical and theoretical foundation of any query language for graphs [14]. However, specific query languages for graphs may support other practical features, such as aggregation (GROUP BY, COUNT, etc.), more complex filters and datatype operators (e.g., range queries on years extracted from a date), federation for

---

[6]Some authors distinguish *2-way regular path queries* from regular path queries, where only the former supports inverses.
[7]Mapping variables to paths requires special treatment [14]. Cypher [159] returns a string that encodes a path, upon which certain functions such as length($\cdot$) can be applied. G-CORE [13], on the other hand, allows for returning paths, and supports additional operators on them, including projecting them as graphs, applying cost functions, and more besides.

querying remotely hosted graphs over the Web, languages for updating graphs, support for semantic entailment regimes, etc. For more information, we refer to the documentation of the respective query languages (e.g., [13, 207]) and to the survey by Angles et al. [14].

## 3 SCHEMA, IDENTITY, CONTEXT

In this section we describe various enhancements and extensions of the data graph – relating to schema, identity and context – that provide additional structures for accumulating knowledge. Henceforth, we refer to a *data graph* as a collection of data represented as nodes and edges using one of the models discussed in Section 2. We refer to a *knowledge graph* as a data graph potentially enhanced with representations of schema, identity, context, ontologies and/or rules. These additional representations may be embedded in the data graph, or layered above it. Representations for schema, identity and context are discussed herein, while ontologies and rules will be discussed in Section 4.

### 3.1 Schema

One of the benefits of modelling data as graphs – versus, for example, the relational model – is the option to forgo or postpone the definition of a schema. However, when modelling data as graphs, schemata *can* be used to prescribe a high-level structure and/or semantics that the graph follows or should follow. We discuss three types of graph schemata: *semantic*, *validating*, and *emergent*.

*3.1.1 Semantic schema.* A semantic schema allows for defining the meaning of high-level terms (aka *vocabulary* or *terminology*) used in the graph, which facilitates reasoning over graphs using those terms. Looking at Figure 1, for example, we may notice some natural groupings of nodes based on the types of entities to which they refer. We may thus decide to define classes to denote these groupings, such as Event, City, etc. In fact, Figure 1 already illustrates three low-level classes – Open Market, Food Market, Drinks Festival – grouping similar entities with an edge labelled type. We may subsequently observe some natural relations between some of these classes that we would like to capture. In Figure 10, we present a class hierarchy for events where children are defined to be *subclasses* of their parents such that if we find an edge (EID15)—type→(Food Festival) in our graph, we may also *infer* that (EID15)—type→(Festival) and (EID15)—type→(Event).

Aside from classes, we may also wish to define the semantics of edge labels, aka properties. Returning to Figure 1, we may consider that the properties city and venue are *sub-properties* of a more general property location, such that given an edge (Santa Lucía)—city→(Santiago), for example, we may also infer that (Santa Lucía)-location→(Santiago). We may also consider, for example, that bus and flight are both sub-properties of a more general property connects to. As such, properties may also form a hierarchy. We may further define the *domain* of properties, indicating the class(es) of entities for nodes from which edges with that property extend; for example, we may define that the domain of connects to is a class Place, such that given the previous sub-property relations, we could conclude that (Arica)—type→(Place). Conversely, we may define the *range* of properties, indicating the class(es) of entities for nodes to which edges with that property extend; for example, we may define that the range of city is a class City, inferring that (Arica)—type→(City).

A prominent standard for defining a semantic schema for (RDF) graphs is the RDF Schema (RDFS) standard [66], which allows for defining subclasses, subproperties, domains, and ranges amongst the classes and properties used in an RDF graph, where such definitions can be serialised as a graph. We illustrate the semantics of these features in Table 1 and provide a concrete example of definitions in Figure 11 for a sample of terms used in the running example. These definitions can then be embedded into a data graph. More generally, the semantics of terms used in a graph can be defined in much more depth than seen here, as is supported by the *Web Ontology Language* (*OWL*) standard [228] for RDF graphs. We will return to such semantics later in Section 4.

Fig. 10. Example class hierarchy for `Event`

Table 1. Definitions for sub-class, sub-property, domain and range features in semantic schemata

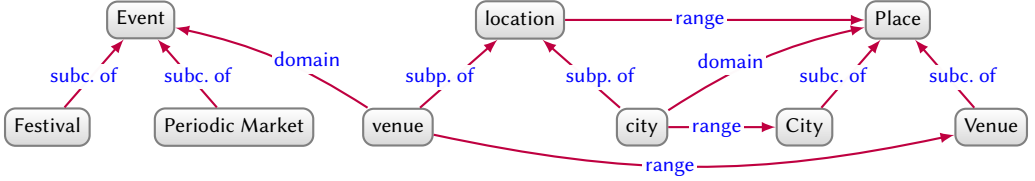| Feature | Definition | Condition | Example |
|---|---|---|---|
| SUBCLASS | $c$ –subc. of→ $d$ | $x$ –type→ $c$ implies $x$ –type→ $d$ | City –subc. of→ Place |
| SUBPROPERTY | $p$ –subp. of→ $q$ | $x$ –$p$→ $y$ implies $x$ –$q$→ $y$ | venue –subp. of→ location |
| DOMAIN | $p$ –domain→ $c$ | $x$ –$p$→ $y$ implies $x$ –type→ $c$ | venue –domain→ Event |
| RANGE | $p$ –range→ $c$ | $x$ –$p$→ $y$ implies $y$ –type→ $c$ | venue –range→ Venue |



Fig. 11. Example schema graph describing sub-classes, sub-properties, domains, and ranges

Semantic schema are typically defined for incomplete graph data, where the absence of an edge between two nodes, such as Viña del Mar –flight→ Arica, does not mean that the relation does not hold in the real world. Therefore, from the graph of Figure 1, we cannot assume that there is no flight between Viña del Mar and Arica. In contrast, if the *Closed World Assumption* (*CWA*) were adopted – as is the case in many classical database systems – it would be assumed that the data graph is a complete description of the world, thus allowing to assert with certainty that no flight exists between the two cities. Systems that do not adopt the CWA are said to adopt the *Open World Assumption* (*OWA*). A consequence of CWA is that the addition of an edge to the data graph may contradict what was previously assumed to be false (due to missing information), whereas with OWA, a statement that is proven false continues to be false with the addition of more edges.

Considering our running example, it would be unreasonable to assume that the tourism organisation has complete knowledge of everything describable in its knowledge graph. However, it is inconvenient if a system is unable to definitely answer "*yes*" or "*no*" to questions such as "*is there a flight between Arica and Viña del Mar?*", especially when the organisation is certain that it has complete knowledge of the flights. A compromise between OWA and CWA is the *Local Closed World Assumption* (*LCWA*), where portions of the data graph are assumed to be complete.

*3.1.2 Validating schema.* When graphs are used to represent diverse, incomplete data at large-scale, the OWA is the most appropriate choice for a *default* semantics. But in some scenarios, we may wish to guarantee that our data graph – or specific parts thereof – are in some sense "complete". Returning to Figure 1, for example, we may wish to ensure that all events have at least a name, a
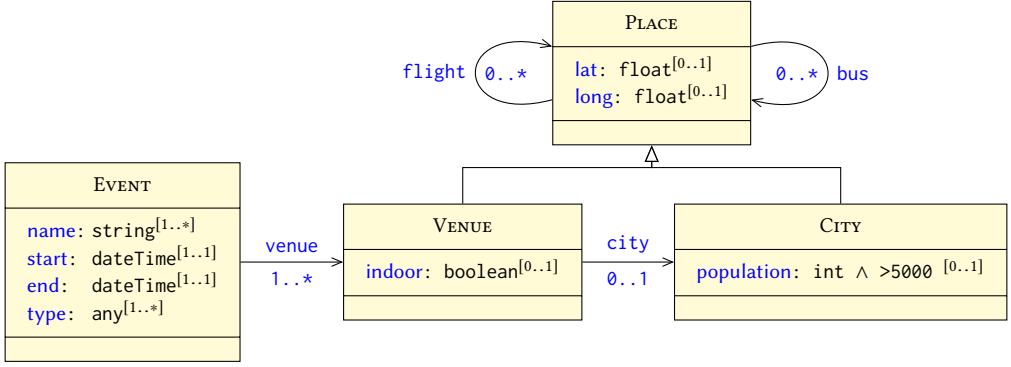
Fig. 12. Example shapes graph depicted as a UML-like diagram

venue, a start date, and an end date, such that applications using the data – e.g., one that sends event notifications to users – can ensure that they have the minimal information required. Furthermore, we may wish to ensure that the city of an event is *stated to be* a city (rather than *inferring* that it is a city). We can define such constraints in a validating schema and validate the data graph with respect to the resulting schema, listing constraint violations (if any). Thus while semantic schemata allow for inferring new graph data, validating schemata allow for validating existing graph data.

A standard way to define a validating schema for graphs is using *shapes* [278, 287, 398]. A shape *targets* a set of nodes in a data graph and specifies *constraints* on those nodes. The shape's target can be defined in many ways, such as targetting all instances of a class, the domain or range of a property, the result of a query, nodes connected to the target of another shape by a given property, etc. Constraints can then be defined on the targetted nodes, such as to restrict the number or types of values taken on a given property. A *shapes graph* is formed from a set of interrelated shapes.

Shapes graphs can be depicted as UML-like class diagrams, where Figure 12 illustrates an example of a shapes graph based on Figure 1, defining constraints on four interrelated shapes. Each shape – denoted with a box like PLACE, EVENT, etc. – is associated with a set of constraints. Nodes conform to a shape if and only if they satisfy all constraints defined on the shape. Inside each shape box are placed constraints on the number (e.g., [1..*] denotes one-to-many, [1..1] denotes precisely one, etc.) and types (e.g., string, dateTime, etc.) of nodes that conforming nodes can relate to with a property (e.g., name, start, etc.). Another option is to place constraints on the number of nodes conforming to a particular shape that the conforming node can relate to with a property (thus generating edges between shapes); for example, EVENT —venue→ VENUE denotes that conforming nodes for EVENT must relate to at least one node with the property venue that conforms to the VENUE shape. Shapes can inherit the constraints of parent shapes – denoted with an △ connector – as in the case of CITY and VENUE, whose conforming nodes must also conform to the PLACE shape.

Given a shape and a targetted node, it is possible to check if the node conforms to that shape or not, which may require checking conformance of other nodes; for example, the node EID15 conforms to the EVENT shape not only based on its local properties, but also based on conformance of Santa Lucía to VENUE and Santiago to CITY. Conformance dependencies may also be recursive, where the conformance of Santiago to CITY requires that it conforms to PLACE, which requires that Viña del Mar and Arica conform to PLACE, and so on. Conversely, EID16 does not conform to EVENT, as it does not have the start and end properties required by the example shapes graph.

When declaring shapes, the data modeller may not know in advance the entire set of properties that some nodes can have. An *open shape* allows the node to have additional properties not specified by the shape, while a *closed shape* does not. For example, if we add the edge (Santiago)–founder→(Pedro de Valdivia) to the graph represented in Figure 1, then (Santiago) only conforms to the [City] shape if that shape is defined as open (since the shape does not mention founder).

Practical languages for shapes often support additional boolean features, such as conjunction (*AND*), disjunction (*OR*), and negation (*NOT*) of shapes; for example, we may say that all the values of venue should conform to the shape [Venue *AND* (*NOT* City)], making explicit that venues in the data graph should not be directly given as cities. However, shape languages that freely combine recursion and negation may lead to semantic problems, depending on how their semantics are defined. To illustrate, consider the following case inspired by the barber paradox [287], involving a shape [Barber] whose conforming nodes shave at least one node conforming to [Person *AND* (*NOT* Barber)]. Now, given (only) (Bob)–shave→(Bob) with (Bob) conforming to [Person], does (Bob) conform to [Barber]? If *yes* – if (Bob) conforms to [Barber] – then (Bob) violates the constraint by not shaving at least one node conforming to [Person *AND* (*NOT* Barber)]. If *no* – if (Bob) does not conform to [Barber] – then (Bob) satisfies the [Barber] constraint by shaving such a node. Semantics to avoid such paradoxical situations have been proposed based on stratification [57], partial assignments [100], and stable models [171].

Although validating schemata and semantic schemata serve different purposes, they can complement each other. In particular, a validating schema can take into consideration a semantic schema, such that, for example, validation is applied on the data graph including inferences. Taking the class hierarchy of Figure 10 and the shapes graph of Figure 12, for example, we may define the target of the [Event] shape as the nodes that are of type Event (the class). If we first apply inferencing with respect to the class hierarchy of the semantic schema, the [Event] shape would now target (EID15) and (EID16). The presence of a semantic schema may, however, require adapting the validating schema. Taking into account, for example, the aforementioned class hierarchy would require defining a relaxed cardinality on the type property. Open shapes may also be preferred in such cases rather than enumerating constraints on all possible properties that may be inferred on a node.

We provide high-level definitions for shapes and related concepts in Appendix B.3.2. Two shape languages have recently emerged for RDF graphs: *Shape Expressions* (*ShEx*), published as a W3C Community Group Report [398]; and *SHACL* (*Shapes Constraint Language*), published as a W3C Recommendation [278]. These languages support the discussed features (and more) and have been adopted for validating graphs in a number of domains relating to health-care [494], scientific literature [206], spatial data [80], amongst others. More details about ShEx and SHACL can be found in the book by Labra Gayo et al. [287]. A recently proposed language that can be used as a common basis for both ShEx and SHACL reveals their similarities and differences [286]. A similar notion of schema has been proposed by Angles [12] for property graphs.

*3.1.3 Emergent schema.* Both semantic and validating schemata require a domain expert to explicitly specify definitions and constraints. However, a data graph will often exhibit latent structures that can be automatically extracted as an *emergent schema* [388] (aka *graph summary* [81, 302, 467]).

A framework often used for defining emergent schema is that of *quotient graphs*, which partition groups of nodes in the data graph according to some equivalence relation while preserving some structural properties of the graph. Taking Figure 1, we can intuitively distinguish different *types* of nodes based on their context, such as event nodes, which link to venue nodes, which in turn link to city nodes, and so forth. In order to describe the structure of the graph, we could consider six partitions of nodes: *event*, *name*, *venue*, *class*, *date-time*, *city*. In practice, these partitions may be computed based on the class or shape of the node. Merging the nodes of each partition into one node while preserving edges leads to the quotient graph shown in Figure 13: the nodes of this
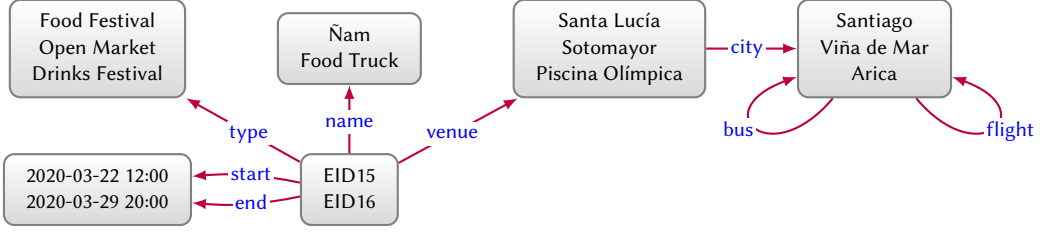
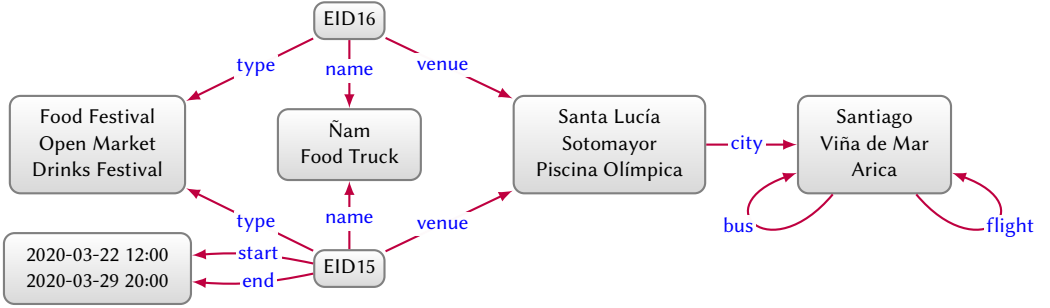Fig. 13. Example quotient graph simulating the data graph in Figure 1



Fig. 14. Example quotient graph bisimilar with the data graph in Figure 1

quotient graph are the partitions of nodes from the data graph and the edge $(X)\!-\!y\!\rightarrow\!(Z)$ is in the quotient graph if and only if there exists $x \in X$ and $z \in Z$ such that $(x)\!-\!y\!\rightarrow\!(z)$ is in the data graph.

There are many ways in which quotient graphs may be defined, depending not only on how nodes are partitioned, but also how the edges are defined. Different quotient graphs may provide different guarantees with respect to the structure they preserve. Formally, we can say that every quotient graph *simulates* its input graph (based on the *simulation relation* of set membership between data nodes and quotient nodes), meaning that for all $x \in X$ with $x$ an input node and $X$ a quotient node, if $(x)\!-\!y\!\rightarrow\!(z)$ is an edge in the data graph, then there must exist an edge $(X)\!-\!y\!\rightarrow\!(Z)$ in the quotient graph such that $z \in Z$; for example, the quotient graph of Figure 13 simulates the data graph of Figure 1. However, this quotient graph seems to suggest (for instance) that (EID16) would have a start and end date in the data graph when this is not the case. A stronger notion of structural preservation is given by *bisimilarity*, which in this case would further require that if $(X)\!-\!y\!\rightarrow\!(Z)$ is an edge in the quotient graph, then for all $x \in X$, there must exist a $z \in Z$ such that $(x)\!-\!y\!\rightarrow\!(z)$ is in the data graph; this is not satisfied by (EID16) in the quotient graph of Figure 13, which does not have an outgoing edge labelled start or end in the original data graph. Figure 14 illustrates a bisimilar version of the quotient graph, splitting the *event* partition into two nodes reflecting their different outgoing edges. An interesting property of bisimilarity is that it preserves forward-directed paths: given a path expression $r$ without inverses and two bisimilar graphs, $r$ will match a path in one graph if and only if it matches a corresponding path in the other bisimilar graph. One can verify, for example, that a path matches $(x)\text{-city}\cdot(\text{flight}|\text{bus})^*\!\rightarrow\!(z)$ in Figure 1 if and only if there is a path matching $(X)\text{-city}\cdot(\text{flight}|\text{bus})^*\!\rightarrow\!(Z)$ in Figure 14 such that $x \in X$ and $z \in Z$.

There are many ways in which quotient graphs may be defined, depending on the equivalence relation that partitions nodes. Furthermore, there are many ways in which other similar or bisimilar graphs can be defined, depending on the (bi)simulation relation that preserves the data graph's structure [81]. We provide formal definitions for the notions of *quotient graphs*, *simulation* and

*bisimulation* in Appendix B.3.3. Such techniques aim to *summarise* the data graph into a higher-level topology. In order to reduce the memory overhead of the quotient graph, in practice, nodes may rather be labelled with the cardinality of the partition and/or a high-level label (e.g., *event*, *city*) for the partition rather than storing the labels of all nodes in the partition.

Various other forms of emergent schema not based on a quotient graph framework have also been proposed; examples include emergent schemata based on relational tables [388], formal concept analysis [189], and so forth. Emergent schemata may be used to provide a human-understandable overview of the data graph, to aid with the definition of a semantic or validating schema, to optimise the indexing and querying of the graph, to guide the integration of data graphs, and so forth. We refer to the survey by Čebirić et al. [81] for further details.

## 3.2 Identity

In Figure 1, we use nodes like (Santiago), but to which Santiago does this node refer? Do we refer to Santiago de Chile, Santiago de Cuba, Santiago de Compostela, or do we perhaps refer to the indie rock band Santiago? Based on edges such as (Santa Lucía)—city→(Santiago), we may deduce that it is one of the three cities mentioned (not the rock band), and based on the fact that the graph describes tourist attractions in Chile, we may further deduce that it refers to Santiago de Chile. Without further details, however, *disambiguating* nodes of this form may rely on heuristics prone to error in more difficult cases. To help avoid such ambiguity, first we may use globally unique identifiers to avoid naming clashes when the knowledge graph is extended with external data, and second we may add external identity links to disambiguate a node with respect to an external source.

*3.2.1 Global identifiers.* Assume we wished to compare tourism in Chile and Cuba, and we have acquired an appropriate knowledge graph for Cuba. Part of the benefit of using graphs to model data is that we can merge two graphs by taking their union. However, as shown in Figure 15, using an ambiguous node like (Santiago) may result in a *naming clash*: the node is referring to two different real-world cities in both graphs, where the merged graph indicates that Santiago is a city in both Chile and Cuba (rather than two different cities).[8] A practical way to avoid such naming clashes would be to use *namespaces* like `chile:` or `cuba:` in the corresponding graphs, such that nodes like (chile:Santiago) and (cuba:Santiago) will not clash so long as distinct namespaces are used.

In the context of the Semantic Web, the RDF data model goes one step further and recommends that global Web identifiers be used for nodes and edge labels. However, rather than adopt the *Uniform Resource Locators (URLs)* used to identify the location of *information resources* such as webpages, RDF 1.1 proposes to use *Internationalised Resource Identifiers (IRIs)* to identify *non-information resources* such as cities or events.[9] Hence, for example, in the RDF representation of the Wikidata [515] – a knowledge graph proposed to complement Wikipedia, discussed in more detail in Section 10 – while the URL (https://www.wikidata.org/wiki/Q2887) refers to a webpage that can be loaded in a browser providing human-readable meta-data about Santiago, the IRI (http://www.wikidata.org/entity/Q2887) refers to the city itself. Distinguishing the identifiers for both resources (the webpage and the city itself) avoids naming clashes; for example, if we use the URL to identify both the webpage and the city, we may end up with an edge in our graph, such as (with readable labels below the edge):

(http://www.wikidata.org/wiki/Q2887)——http://www.wikidata.org/wiki/Property:P112——→(https://www.wikidata.org/wiki/Q203534)

    [Santiago (URL)]                  [founded by (URL)]             [Pedro de Valdivia (URL)]

---

[8]Such a naming clash is not unique to graphs, but could also occur if merging tables, trees, etc.

[9]Uniform Resource Identifiers (URIs) can be Uniform Resource Locators (URLs), used to locate information resources, and Uniform Resource Names (URNs), used to name non-information resources. Internationalised Resource Identifiers (IRIs) are URIs that allow Unicode. For example, `http://example.com/Ñam` is an IRI, but not a URI, due to the use of "Ñ". Percentage encoding – `http://example.com/%C3%91am` – can encode an IRI as a URI (but reduces readability).
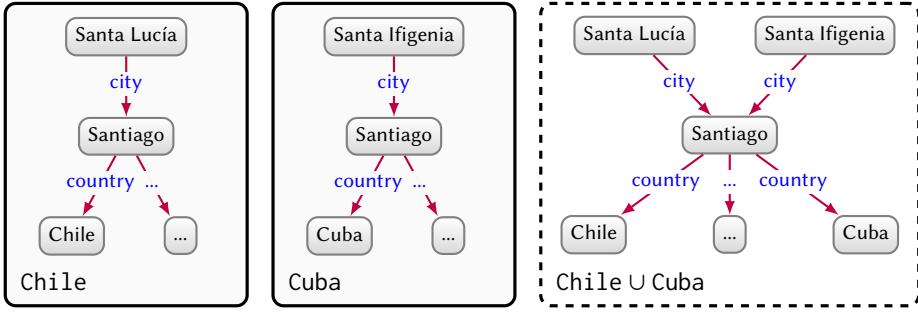
Fig. 15. Result of merging two graphs with ambiguous local identifiers

Such an edge leaves ambiguity: was Pedro de Valdivia the founder of the webpage, or the city? Using IRIs for entities distinct from the URLs for the webpages that describe them avoids such ambiguous cases, where Wikidata thus rather defines the previous edge as follows:



using IRIs for the city, person, and founder of, distinct from the webpages describing them.

If HTTP IRIs are used to identify the graph's entities, when the IRI is looked up (via HTTP), the web-server can return (or redirect to) a description of that entity in formats such as RDF. This further enables RDF graphs to link to related entities described in external RDF graphs over the Web, giving rise to *Linked Data* [37, 216] (discussed in Section 9).

*3.2.2 External identity links.* Assume that the tourist board opts to define the chile: namespace with an IRI such as http://turismo.cl/entity/ on a web-server that they control, allowing nodes such as (chile:Santiago) – a shortcut for the IRI (http://turismo.cl/entity/Santiago) – to be looked up over the Web. While using such a naming scheme helps to avoid naming clashes, the use of IRIs does not necessarily help ground the identity of a resource. For example, an external geographic knowledge graph may assign the same city the IRI (geo:SantiagoDeChile) in their own namespace, where we have no direct way of knowing that the two identifiers refer to the same city. If we merge the two knowledge graphs, we will end up with two distinct nodes for the same city.

There are a number of ways to ground the identity of an entity. The first is to associate the entity with uniquely-identifying information in the graph, such as its geo-coordinates, its postal code, the year it was founded, etc. Each additional piece of information removes ambiguity as to which city is being referred, providing (for example) more options for matching the city with its analogue in external sources. A second option is to use *identity links* to state that a local entity has the same identity as another *coreferent* entity found in an external source; an instantiation of this concept can be found in the OWL standard, which defines the owl:sameAs property relating coreferent entities. Using this property, we could state the edge (chile:Santiago)-owl:sameAs→(geo:SantiagoDeChile) in our RDF graph, thus establishing an identity link between the corresponding nodes in both graphs. The semantics of owl:sameAs defined by the OWL standard then allow us to combine the data for both nodes. Such semantics will be discussed later in Section 4. Ways in which identity links can be computed will also be discussed later in Section 8.

*3.2.3 Datatypes.* Consider the two date-times on the left of Figure 1: how should we assign these nodes global identifiers? Intuitively it would not make sense, for example, to assign IRIs to these nodes since their syntactic form tells us what they refer to: specific dates and times in March 2020.

This syntactic form is further recognisable by machine, meaning that with appropriate software, we could order such values in ascending or descending order, extract the year, and so forth.

Most practical data models for graphs allow for defining nodes that are datatype values. RDF utilises *XML Schema Datatypes* (*XSD*) [386], amongst others, where a datatype node is given as a pair $(l, d)$ where $l$ is a lexical string, such as `"2020-03-29T20:00:00"`, and $d$ is an IRI denoting the datatype, such as `xsd:dateTime`. The node is then denoted `"2020-03-29T20:00:00"^^xsd:dateTime`. Datatype nodes in RDF are called *literals* and are not allowed to have outgoing edges. Other datatypes commonly used in RDF data include `xsd:string`, `xsd:integer`, `xsd:decimal`, `xsd:boolean`, etc. In case that the datatype is omitted, the value is assumed to be of type `xsd:string`. Applications built on top of RDF can then recognise these datatypes, parse them into datatype objects, and apply equality checks, normalisation, ordering, transformations, casting, according to their standard definition. In the context of property graphs, Neo4j [332] also defines a set of internal datatypes on property values that includes numbers, strings, booleans, spatial points, and temporal values.

*3.2.4 Lexicalisation.* Global identifiers for entities will sometimes have a human-interpretable form, such as `chile:Santiago`, but the identifier strings themselves do not carry any formal semantic significance. In other cases, the global identifiers used may not be human-interpretable by design. In Wikidata, for instance, Santiago de Chile is identified as `wd:Q2887`, where such a scheme has the advantage of providing better persistence and of not being biased to a particular human language. For example, the Wikidata identifier for Eswatini (`wd:Q1050`) was not affected when the country changed its name from Swaziland, and does not necessitate choosing between languages for creating (more readable) IRIs such as `wd:Eswatini` (English), `wd:eSwatini` (Swazi), `wd:Esuatini` (Spanish), etc.

Since identifiers can be arbitrary, it is common to add edges that provide a human-interpretable label for nodes, such as `wd:Q2887`-`rdfs:label`→`"Santiago"`, indicating how people may refer to the subject node linguistically. Linguistic information of this form plays an important role in grounding knowledge such that users can more clearly identify which real-world entity a particular node in a knowledge graph actually references [115]; it further permits cross-referencing entity labels with text corpora to find, for example, documents that potentially speak of a given entity [317]. Labels can be complemented with aliases (e.g., `wd:Q2887`-`skos:altLabel`→`"Santiago de Chile"`) or comments (e.g. `wd:Q2887`-`rdfs:comment`→`"Santiago is the capital of Chile"`) to further help ground the node's identity.

Nodes such as `"Santiago"` denote string literals, rather than an identifier. Depending on the specific graph model, such literal nodes may also be defined as a pair $(s, l)$, where $s$ denotes the string and $l$ a language code; in RDF, for example we may state `chile:City`-`rdfs:label`→`"City"@en`, `chile:City`-`rdfs:label`→`"Ciudad"@es`, etc., indicating labels for the node in different languages. In other models, the pertinent language can rather be specified, e.g., via metadata on the edge. Knowledge graphs with human-interpretable labels, aliases, comments, etc., (in various languages) are sometimes called (*multilingual*) *lexicalised knowledge graphs* [53].

*3.2.5 Existential nodes.* When modelling incomplete information, we may in some cases know that there must exist a particular node in the graph with particular relationships to other nodes, but without being able to identify the node in question. For example, we may have two co-located events `chile:EID42` and `chile:EID43` whose venue has yet to be announced. One option is to simply omit the venue edges, in which case we lose the information that these events have a venue and that both events have the same venue. Another option might be to create a fresh IRI representing the venue, but semantically this becomes indistinguishable from there being a known venue. Hence some graph models permit the use of existential nodes, represented here as a blank circle:
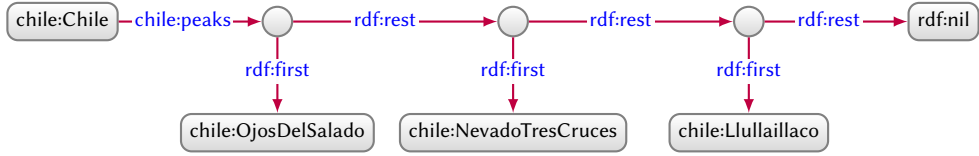
`chile:EID42` —`chile:venue`→ ◯ ←`chile:venue`— `chile:EID43`

Fig. 16. RDF list representing the three largest peaks of Chile, in order

These edges denote that there exists a common venue for (chile:EID42) and (chile:EID42) without identifying it. Existential nodes are supported in RDF as blank nodes [106], which are also commonly used to support modelling complex elements in graphs, such as *RDF lists* [106, 235]. Figure 16 exemplifies an RDF list, which uses blank nodes in a linked-list structure to encode order. Though existential nodes can be convenient, their presence can complicate operations on graphs, such as deciding if two data graphs have the same structure modulo existential nodes [106, 234]. Hence methods for *skolemising* existential nodes in graphs – replacing them with canonical labels – have been proposed [234, 305]. Other authors rather call to minimise the use of such nodes in graph data [216].

## 3.3 Context

Many (arguably *all*) facts presented in the data graph of Figure 1 can be considered true with respect to a certain *context*. With respect to *temporal context*, (Santiago) has only existed as a city since 1541, flights from (Arica) to (Santiago) began in 1956, etc. With respect to *geographic context*, the graph describes events in Chile. With respect to *provenance*, data relating to (EID15) were taken from – and are thus said to be true with respect to – the Ñam webpage on January 4$^{\text{th}}$, 2020. Other forms of context may also be used. We may further combine contexts, such as to indicate that (Arica) is a Chilean city (*geographic*) since 1883 (*temporal*) according to the Treaty of Ancón (*provenance*).

By context we herein refer to the *scope of truth*, and thus talk about the context in which some data are held to be true [198, 324]. The graph of Figure 1 leaves much of its context implicit. However, making context explicit can allow for interpreting the data from different perspectives, such as to understand what held true in 2016, what holds true excluding webpages later found to have spurious data, etc. As seen in the previous examples, context for graph data may be considered at different levels: on individual nodes, individual edges, or sets of edges (sub-graphs). We now discuss various representations by which context can be made explicit at different levels.

*3.3.1 Direct representation.* The first way to represent context is to consider it as data no different from other data. For example, the dates for the event (EID15) in Figure 1 can be seen as representing a form of temporal context, indicating the temporal scope within which edges such as (EID15)—venue→(Santa Lucía) are held true. Another option is to change a relation represented as an edge, such as (Santiago)—flight→(Arica), into a node, such as seen in Figure 3, allowing to assign additional context to the relation. While in these examples context is represented in an ad hoc manner, a number of specifications have been proposed to represent context as data in a more standard way. One example is the *Time Ontology* [103], which specifies how temporal entities, intervals, time instants, etc. – and relations between them such as before, overlaps, etc. – can be described in RDF graphs in an interoperable manner. Another example is the *PROV Data Model* [181], which specifies how provenance can be described in RDF graphs, where entities (e.g., graphs, nodes, physical document) are derived from other entities, are generated and/or used by activities (e.g., extraction, authorship), and are attributed to agents (e.g., people, software, organisations).

*3.3.2 Reification.* Often we may wish to directly define the context of edges themselves; for example, we may wish to state that the edge (Santiago)—flight→(Arica) is valid from 1956. While we
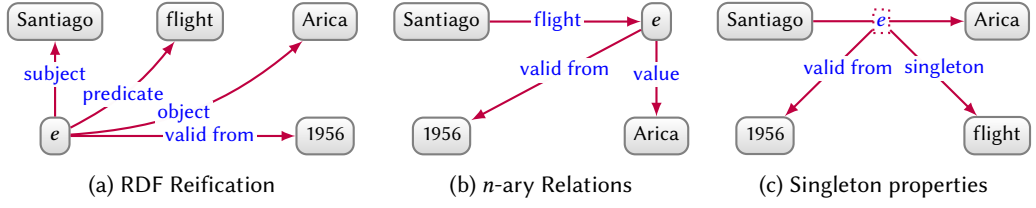
Fig. 17. Three representations of temporal context on an edge in a directed-edge labelled graph

could use the pattern of turning the edge into a node – as illustrated in Figure 3 – to directly represent such context, another option is to use *reification*, which allows for making statements about statements in a generic manner (or in the case of a graph, for defining edges about edges). In Figure 17 we present three forms of reification that can be used for modelling temporal context on the aforementioned edge within a directed edge-labelled graph [224]. We use $e$ to denote an arbitrary identifier representing the edge itself to which the contextual information can be associated. Unlike in a direct representation, $e$ represents an edge, not a flight. RDF reification [106] (Figure 17a) defines a new node $(e)$ to represent the edge and connects it to the source node (via subject), target node (via object), and edge label (via predicate) of the edge. In contrast, $n$-ary relations [106] (Figure 17b) connect the source node of the edge directly to the edge node $(e)$ with the label of the edge; the target node of the edge is then connected to $(e)$ (via value). Finally, singleton properties [361] (Figure 17c) rather use $e$ as an edge label, connecting it to a node indicating the original edge label (via singleton). Other forms of reification have been proposed in the literature, including, for example, NdFluents [183]. In general, a reified edge does not assert the edge it reifies; for example, we may reify an edge to state that it is no longer valid. We refer to the work of Hernández et al. [224] for further comparison of reification alternatives and their relative strengths and weaknesses.

*3.3.3 Higher-arity representation.* As an alternative to reification, we can rather use higher-arity representations for modelling context. Taking again the edge (Santiago)—flight→(Arica), Figure 18 illustrates three higher-arity representations of temporal context. First, we can use a named graph (Figure 18a) to contain the edge and then define the temporal context on the graph name. Second, we can use a property graph (Figure 18b) where the temporal context is defined as an attribute on the edge. Third, we can use *RDF** [208] (Figure 18c): an extension of RDF that allows edges to be defined as nodes. Amongst these options, the most flexible is the named graph representation, where we can assign context to multiple edges at once by placing them in one named graph; for example, we can add more edges to the named graph of Figure 18a that are also valid from 1956. The least flexible option is RDF*, which, in the absence of an edge id, does not permit different groups of contextual values to be assigned to an edge; for example, considering the edge (Chile)—president→(M. Bachelet), if we add four contextual values to this edge to state that it was valid from 2006 until 2010 and valid from 2014 until 2018, we cannot pair the values, but may rather have to create a node to represent different presidencies (in the other models, we could have used two named graphs or edge ids).

*3.3.4 Annotations.* Thus far we have discussed representing context in a graph, but we have not spoken about automated mechanisms for reasoning about context; for example, if there are only seasonal summer flights from (Santiago) to (Arica), we may wish to find other routes from Santiago for winter events taking place in (Arica). While the dates for buses, flights, etc., can be represented directly in the graph, or using reification, writing a query to manually intersect the corresponding temporal contexts will be tedious – or may not even be possible at all. Another alternative is
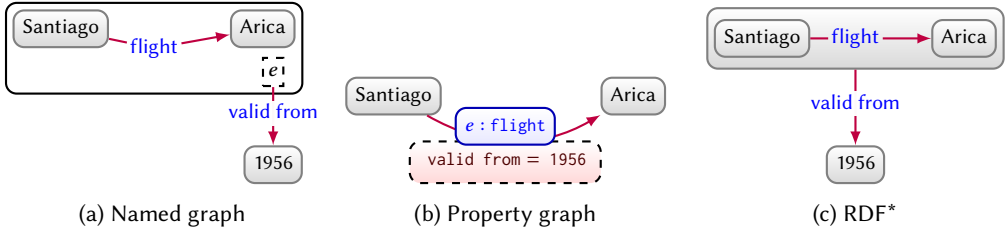
Fig. 18. Three higher-arity representations of temporal context on an edge

to consider *annotations* that provide mathematical definitions of a contextual domain and key operations possible within that domain that can then be applied automatically.

Some annotations model a particular contextual domain; for example, *Temporal RDF* [203] allows for annotating edges with time intervals, such as (Chile)—$\overset{\text{president}}{\underset{[2006,\ 2010]}{}}$→(M. Bachelet), while *Fuzzy RDF* [477] allows for annotating edges with a degree of truth such as (Santiago)—$\overset{\text{climate}}{\underset{0.8}{}}$→(Semi-Arid), indicating that it is more-or-less true – with a degree of 0.8 – that Santiago has a semi-arid climate.

Other forms of annotation are domain-independent; for example, *Annotated RDF* [125, 503, 546] allows for representing various forms of context modelled as *semi-rings*: algebraic structures consisting of domain values (e.g., temporal intervals, fuzzy values, etc.) and two main operators to combine domain values: *meet* and *join*.[10] We provide an example in Figure 19, where $G$ is annotated with values from a simplified temporal domain consisting of sets of integers (1–365) representing days of the year. For brevity we use an interval notation, where, for example, $\{[150, 152]\}$ indicates the set $\{150, 151, 152\}$. Query $Q$ then asks for flights from Santiago to cities with events; this query will check and return an annotation reflecting the temporal validity of each answer. To derive these answers, we first require applying a conjunction of annotations on compatible flight and city edges, applying the *meet operator* to compute the annotation for which both edges hold. The natural way to define meet in our scenario is as the intersection of sets of days, where, for example, applying meet on the event annotation $\{[150, 152]\}$ and the flight annotation $\{[1, 120], [220, 365]\}$ for (Punta Arenas) leads to the empty time interval $\{\}$, which may thus lead to the city being filtered from the results (depending on the query evaluation semantics). However, for (Arica), we find two different non-empty intersections: $\{[123, 125]\}$ for (EID16) and $\{[276, 279]\}$ for (EID17). Given that we are interested in the city (a projected variable), rather than the event, we can thus combine these two annotations for (Arica) using the *join operator*, returning the annotation in which either result holds true. In our scenario, the natural way to define join is as the union of the sets of days, giving $\{[123, 125], [276, 279]\}$. We provide formal definitions in Appendix B.4.1 based on the general framework proposed by Zimmermann et al. [546] for annotations on graphs.

*3.3.5 Other contextual frameworks.* Other frameworks have been proposed for modelling and reasoning about context in graphs. A notable example is that of *contextual knowledge repositories* [240], which allow for assigning individual (sub-)graphs to their own context. Unlike in the case of named graphs, context is explicitly modelled along one or more dimensions, where each (sub-)graph must take a value for each dimension. Each dimension is further associated with a partial order over its values – e.g., (2020-03-22) ≤ (2020-03) ≤ (2020) – allowing to select and combine sub-graphs that are valid within contexts at different levels of granularity. Schuetz et al. [442] similarly propose a form of contextual OnLine Analytic Processing (OLAP), based on a data cube formed by dimensions where

---

[10]The *join* operator for annotations is different from the join operator for relational algebra.
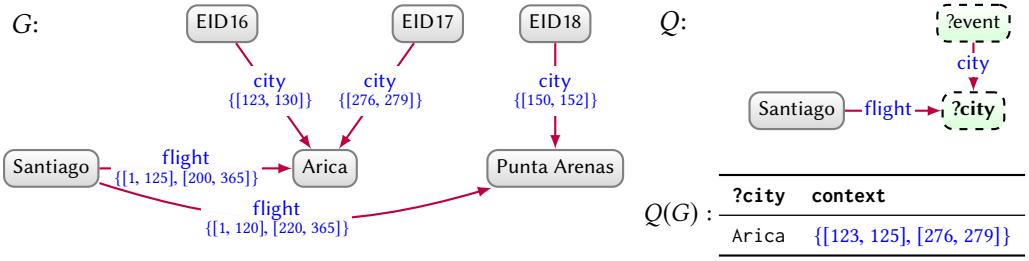
Fig. 19. Example query on a temporally annotated graph

individual cells contain knowledge graphs. Operations such as "*slice-and-dice*" (selecting knowledge according to given dimensions), as well as "*roll-up*" (aggregating knowledge at a higher level) can then be supported. We refer the reader to the respective papers for more details [240, 442].

## 4 DEDUCTIVE KNOWLEDGE

As humans, we can *deduce* more from the data graph of Figure 1 than what the edges explicitly indicate. We may deduce, for example, that the Ñam festival (EID15) will be located in Santiago, even though the graph does not contain an edge EID15—location→Santiago. We may further deduce that the cities connected by flights must have some airport nearby, even though the graph does not contain nodes referring to these airports. In these cases, given the data as premises, and some general rules about the world that we may know *a priori*, we can use a deductive process to derive new data, allowing us to know more than what is explicitly given by the data. These types of general premises and rules, when shared by many people, form part of "*commonsense knowledge*" [323]; conversely, when rather shared by a few experts in an area, they form part of "*domain knowledge*", where, for example, an expert in biology may know that *hemocyanin* is a protein containing copper that carries oxygen in the blood of some species of *Mollusca* and *Arthropoda*.

Machines, in contrast, do not have *a priori* access to such deductive faculties; rather they need to be given formal instructions, in terms of premises and *entailment regimes*, in order to make similar deductions to what a human can make. These entailment regimes formalise the conclusions that logically follow as a consequence of a given set of premises. Once instructed in this manner, machines can (often) apply deductions with a precision, efficiency, and scale beyond human performance. These deductions may serve a range of applications, such as improving query answering, (deductive) classification, finding inconsistencies, etc. As a concrete example involving query answering, assume we are interested in knowing *the festivals located in Santiago*; we may straightforwardly express such a query as per the graph pattern shown in Figure 20. This query returns no results for the graph in Figure 1: there is no node named Festival, and nothing has (directly) the location Santiago. However, an answer (Ñam) could be automatically entailed were we to state that $x$ being a Food Festival *entails* that $x$ is a Festival, or that $x$ having venue $y$ in city $z$ *entails* that $x$ has location $z$. How, then, should such entailments be captured? In Section 3.1.1 we already discussed how the former entailment can be captured with sub-class relations in a semantic schema; the second entailment, however, requires a more expressive entailment regime than seen thus far.

In this section, we discuss ways in which more complex entailments can be expressed and automated. Though we could leverage a number of logical frameworks for these purposes – such as First-Order Logic, Datalog, Prolog, Answer Set Programming, etc. – we focus on *ontologies*, which constitute a formal representation of knowledge that, importantly for us, can be represented as a

Fig. 20. Graph pattern querying for names of festivals in Santiago

graph. We then discuss how these ontologies can be formally defined, how they relate to existing logical frameworks, and how reasoning can be conducted with respect to such ontologies.

## 4.1 Ontologies

To enable entailment, we must be precise about what the terms we use mean. Returning to Figure 1, for example, and examining the node EID16 more closely, we may begin to question how it is modelled, particularly in comparison with EID15. Both nodes – according to the class hierarchy of Figure 10 – are considered to be events. But what if, for example, we wish to define two pairs of start and end dates for EID16 corresponding to the different venues? Should we rather consider what takes place in each venue as a different event? What then if an event has various start and end dates in a single venue: would these also be considered as one (recurring) event, or many events? These questions are facets of a more general question: *what precisely do we mean by an "event"*? Does it happen in one contiguous time interval or can it happen many times? Does it happen in one place or can it happen in multiple? There are no "correct" answers to such questions – we may understand the term "event" in a variety of ways, and thus the answers are a matter of *convention*.

In the context of computing, an *ontology*[11] is then a concrete, formal representation of what terms mean within the scope in which they are used (e.g., a given domain). For example, one event ontology may formally define that if an entity is an "event", then it has precisely one venue and precisely one time instant in which it begins. Conversely, a different event ontology may define that an "event" can have multiple venues and multiple start times, etc. Each such ontology formally captures a particular perspective – a particular *convention*. Under the first ontology, for example, we could not call the Olympics an "event", while under the second ontology we could. Likewise ontologies can guide how graph data are modelled. Under the first ontology we may split EID16 into two events. Under the second, we may elect to keep EID16 as one event with two venues. Ultimately, given that ontologies are formal representations, they can be used to automate entailment.

Like all conventions, the usefulness of an ontology depends on the level of agreement on what that ontology defines, how detailed it is, and how broadly and consistently it is adopted. Adoption of an ontology by the parties involved in one knowledge graph may lead to a consistent use of terms and consistent modelling in that knowledge graph. Agreement over multiple knowledge graphs will, in turn, enhance the interoperability of those knowledge graphs.

Amongst the most popular ontology languages used in practice are the *Web Ontology Language* (*OWL*) [228][12], recommended by the W3C and compatible with RDF graphs; and the *Open Biomedical Ontologies Format* (*OBOF*) [344], used mostly in the biomedical domain. Since OWL is the more widely adopted, we focus on its features, though many similar features are found in both [344]. Before introducing such features, however, we must discuss how graphs are to be *interpreted*.

*4.1.1 Interpretations.* We as humans may *interpret* the node Santiago in the data graph of Figure 1 as referring to the real-world city that is the capital of Chile. We may further *interpret* an edge Arica –flight→ Santiago as stating that there are flights from the city of Arica to this city. We thus

---

[11]The term stems from the philosophical study of *ontology*, concerned with the different kinds of entities that exist, the nature of their existence, what kinds of properties they have, and how they may be identified and categorised.

[12]We could include RDF Schema (RDFS) in this list, but it is largely subsumed by OWL, which builds upon its core.

interpret the data graph as another graph – what we here call the *domain graph* – composed of real-world entities connected by real-world relations. The process of interpretation, here, involves *mapping* the nodes and edges in the data graph to nodes and edges of the domain graph.

Along these lines, we can abstractly define an *interpretation* of a data graph as being composed of two elements: a domain graph, and a mapping from the *terms* (nodes and edge-labels) of the data graph to those of the domain graph. The domain graph follows the same model as the data graph; for example, if the data graph is a directed edge-labelled graph, then so too will be the domain graph. For simplicity, we will speak of directed edge-labelled graphs and refer to the nodes of the domain graph as *entities*, and the edges of the domain graph as *relations*. Given a data graph and an interpretation, while we denote nodes in the data graph by (Santiago), we will denote the entity it refers to in the domain graph by (Santiago) (per the mapping of the given interpretation). Likewise, while we denote an edge by (Arica)—flight→(Santiago), we will denote the relation by (Arica)=flight⟹(Santiago) (again, per the mapping of the given interpretation). In this abstract notion of an interpretation, we do not require that (Santiago) nor (Arica) be the real-world cities, nor even that the domain graph contain real-world entities and relations: an interpretation can have any domain graph and mapping.

Why is such an abstract notion of interpretation useful? The distinction between nodes/edges and entities/relations becomes important when we define the meaning of ontology features and entailment. To illustrate this distinction, if we ask whether there is an edge labelled flight between (Arica) and (Viña del Mar) for the data graph in Figure 1, the answer is *no*. However, if we ask if the entities (Arica) and (Viña del Mar) are connected by the relation flight, then the answer depends on what assumptions we make when interpreting the graph. Under the Closed World Assumption (CWA), if we do not have additional knowledge, then the answer is a definite *no* – since what is not known is assumed to be false. Conversely, under the Open World Assumption (OWA), we cannot be certain that this relation does not exist as this could be part of some knowledge not (yet) described by the graph. Likewise under the Unique Name Assumption (UNA), we can say that the data graph describes *at least two* flights (since (Santiago), (Viña del Mar) and (Arica) are assumed to be different entities and therefore, (Arica)=flight⟹(Santiago) and (Viña del Mar)=flight⟹(Santiago) must be different edges). Conversely, under No Unique Name Assumption (NUNA), we can only say that there is *at least one* such flight since (Viña del Mar) and (Arica), for instance, may be the same entity.

These assumptions (or lack thereof) define which interpretations are valid, and which interpretations *satisfy* which data graphs. The UNA forbids interpretations that map two data terms to the same domain term. The NUNA allows such interpretations. Under CWA, an interpretation that contains an edge (x)=p⟹(y) in its domain graph can only satisfy a data graph from which we can entail (x)—p→(y). Under OWA, an interpretation containing the edge (x)=p⟹(y) can satisfy a data graph not entailing (x)—p→(y) so long it does not contradict that edge.[13] In the case of OWL, the NUNA and OWA are adopted, thus representing the most general case, whereby multiple nodes/edge-labels in the graph may refer to the same entity/relation-type (NUNA), and where anything not entailed by the data graph is *not* assumed to be false as a consequence (OWA).

Beyond our base assumptions, we can associate certain patterns in the data graph with *semantic conditions* that define which interpretations satisfy it; for example, we can add a semantic condition to enforce that if our data graph contains the edge (p)-subp. of→(q), then any edge (x)=p⟹(y) in the domain graph of the interpretation must also have a corresponding edge (x)=q⟹(y) to satisfy the data graph. These semantic conditions then form the features of an ontology language. In what follows, to aid readability, we will introduce the features of OWL using an abstract graphical notation

---

[13]Variations of the CWA can provide a middle ground between a completely open world that makes no assumption about completeness, falsehood of unknown statements, or unicity of names. One example of such variation is Local Closed World Assumption, already mentioned in Section 3.1.1.

Table 2. Ontology features for individuals

| Feature | Axiom | Condition | Example |
|---|---|---|---|
| ASSERTION | $(x)—y\rightarrow(z)$ | $(x)=y\Rightarrow(z)$ | Chile—capital→Santiago |
| NEGATION | $(n)$ type→Neg, sub→$(x)$, pre→$(y)$, obj→$(z)$ | not $(x)=y\Rightarrow(z)$ | $(n)$ type→Neg, sub→Chile, pre→capital, obj→Arica |
| SAME AS | $(x_1)$—same as→$(x_2)$ | $(x_1) = (x_2)$ | Región V—same as→Región de Valparaíso |
| DIFFERENT FROM | $(x_1)$—diff. from→$(x_2)$ | $(x_1) \neq (x_2)$ | Valparaíso—diff. from→Región de Valparaíso |

with abbreviated terms. For details of concrete syntaxes, we rather refer to the OWL and OBOF standards [228, 344]. Likewise we present semantic conditions for interpretations associated with each feature in the same graphical format;[14] further details of these conditions will be described later in Section 4.2, with formal definitions rather provided in Appendix B.5.

*4.1.2 Individuals.* In Table 2, we list the main features supported by OWL for describing *individuals* (e.g., Santiago, EID16), sometimes distinguished from classes and properties. First, we can *assert* (binary) relations between individuals using edges such as Santa Lucía—city→Santiago. In the condition column, when we write $(x)=y\Rightarrow(z)$, for example, we refer to the condition that the given relation holds in the interpretation; if so, the interpretation *satisfies* the axiom. OWL further allows for defining relations to explicitly state that two terms refer to the *same* entity, where, e.g., Región V—same as→Región de Valparaíso states that both refer to the same region (per Section 3.2); or that two terms refer to *different* entities, where, e.g., Valparaíso—diff. from→Región de Valparaíso distinguishes the city from the region of the same name. We may also state that a relation does not hold using *negation*, which can be serialised as a graph using a form of reification (see Figure 17a).
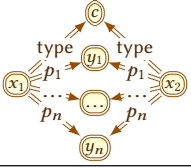
*4.1.3 Properties.* In Section 3.1.1, we already discussed how *subproperties*, *domains* and *ranges* may be defined for properties. OWL allows such definitions, and further includes other features, as listed in Table 3. We may define a pair of properties to be *equivalent*, *inverses*, or *disjoint*. We can further define a particular property to denote a *transitive*, *symmetric*, *asymmetric*, *reflexive*, or *irreflexive* relation. We can also define the multiplicity of the relation denoted by properties, based on being *functional* (many-to-one) or *inverse-functional* (one-to-many). We may further define a *key* for a class, denoting the set of properties whose values uniquely identify the entities of that class. Without adopting a Unique Name Assumption (UNA), from these latter three features we may conclude that two or more terms refer to the same entity. Finally, we can relate a property to a *chain* (a path expression only allowing concatenation of properties) such that pairs of entities related by the chain are also related by the given property. Note that for the latter two features in Table 3 we require representing a list, denoted with a vertical notation ⋮; while such a list may be serialised as a graph in a number of concrete ways, OWL uses RDF lists (see Figure 16).

*4.1.4 Classes.* In Section 3.1.1, we discussed how class hierarchies can be modelled using a *sub-class* relation. OWL supports sub-classes, and many additional features, for defining and making claims about classes; these additional features are summarised in Table 4. Given a pair of classes, OWL allows for defining that they are *equivalent*, or *disjoint*. Thereafter, OWL provides a variety of

---

[14]We use "iff" as an abbreviation for "if and only if" whereby "$\phi$ iff $\psi$" can be read as "if $\phi$ then $\psi$" and "if $\psi$ then $\phi$".

Table 3. Ontology features for property axioms

| Feature | Axiom | Condition (for all $x_*, y_*, z_*$) | Example |
|---|---|---|---|
| SUBPROPERTY | $p$ –subp. of→ $q$ | $x = p → y$ implies $x = q → y$ | venue –subp. of→ location |
| DOMAIN | $p$ –domain→ $c$ | $x = p → y$ implies $x$ –type→ $c$ | venue –domain→ Event |
| RANGE | $p$ –range→ $c$ | $x = p → y$ implies $y$ –type→ $c$ | venue –range→ Venue |
| EQUIVALENCE | $p$ –equiv. p.→ $q$ | $x = p → y$ iff $x = q → y$ | start –equiv. p.→ begins |
| INVERSE | $p$ –inv. of→ $q$ | $x = p → y$ iff $y = q → x$ | venue –inv. of→ hosts |
| DISJOINT | $p$ –disj. p.→ $q$ | not $x \overset{q}{\underset{p}{\rightrightarrows}} y$ | venue –disj. p.→ hosts |
| TRANSITIVE | $p$ –type→ Transitive | $x = p → y = p → z$ implies $x = p → z$ | part of –type→ Transitive |
| SYMMETRIC | $p$ –type→ Symmetric | $x = p → y$ iff $y = p → x$ | nearby –type→ Symmetric |
| ASYMMETRIC | $p$ –type→ Asymmetric | not $x \overset{p}{\underset{p}{\rightrightarrows}} y$ | capital –type→ Asymmetric |
| REFLEXIVE | $p$ –type→ Reflexive | $x \circlearrowleft p$ | part of –type→ Reflexive |
| IRREFLEXIVE | $p$ –type→ Irreflexive | not $x \circlearrowleft p$ | flight –type→ Irreflexive |
| FUNCTIONAL | $p$ –type→ Functional | $y_1 ← p = x = p → y_2$ implies $y_1 = y_2$ | population –type→ Functional |
| INV. FUNCTIONAL | $p$ –type→ Inv. Functional | $x_1 = p → y ← p = x_2$ implies $x_1 = x_2$ | capital –type→ Inv. Functional |
| KEY | $c$ –key→ $\begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$ | $x_1 \overset{p_1}{\underset{p_n}{\cdots}} \{y_1 \dots y_n, c\} \overset{p_1}{\underset{p_n}{\cdots}} x_2$ implies $x_1 = x_2$ | City –key→ lat long |
| CHAIN | $p$ –chain→ $\begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$ | $x = q_1 → y_1 → \dots → y_{n-1} = q_n → z$ implies $x = p → z$ | location –chain→ location part of |

features for defining novel classes by applying set operators on other classes, or based on conditions that the properties of its instances satisfy. First, using set operators, one can define a novel class as the *complement* of another class, the *union* or *intersection* of a list (of arbitrary length) of other classes, or as an *enumeration* of all of its instances. Second, by placing restrictions on a particular property $p$, one can define classes whose instances are all of the entities that have: *some value* from a given class on $p$; *all values* from a given class on $p$;[15] have a specific individual as a value on $p$ (*has value*); have themselves as a reflexive value on $p$ (*has self*); have at least, at most or exactly some number of values on $p$ (*cardinality*); and have at least, at most or exactly some number of values on $p$ from a given class (*qualified cardinality*). For the latter two cases, in Table 4, we use the notation "#$\{a \mid \phi\}$" to count distinct entities satisfying $\phi$ in the interpretation. These features can then be combined to create more complex classes, where combining the examples for INTERSECTION and HAS SELF in Table 4 gives the definition: *self-driving taxis are taxis having themselves as a driver*.

---

[15]While something like flight ← prop– DomesticAirport –all→ NationalFlight might appear to be a more natural example for ALL VALUES, this would be a modelling mistake, as the corresponding *for all* condition is satisfied when no such node exists. In other words, with this example definition, we could infer anything known not to have any flights to be a domestic airport. (We could, however, define the intersection of this class and airport as being a domestic airport.)

Table 4. Ontology features for class axioms and definitions

| Feature | Axiom | Condition (for all $x_*, y_*, z_*$) | Example |
|---|---|---|---|
| SUBCLASS | $c$ —subc. of→ $d$ | $x$ =type→ $c$ implies $x$ =type→ $d$ | City —subc. of→ Place |
| EQUIVALENCE | $c$ —equiv. c.→ $d$ | $x$ =type→ $c$ iff $x$ =type→ $d$ | Human —equiv. c.→ Person |
| DISJOINT | $c$ —disj. c.→ $d$ | not $c$ ←type= $x$ =type→ $d$ | City —disj. c.→ Region |
| COMPLEMENT | $c$ —comp.→ $d$ | $x$ =type→ $c$ iff not $x$ =type→ $d$ | Dead —comp.→ Alive |
| UNION | $c$ —union→ $\begin{matrix} d_1 \\ \vdots \\ d_n \end{matrix}$ | $x$ =type→ $c$ iff $x$ =type→ $d_1$ or $x$ =type→ $\cdots$ or $x$ =type→ $d_n$ | Flight —union→ DomesticFlight InternationalFlight |
| INTERSECTION | $c$ —inter.→ $\begin{matrix} d_1 \\ \vdots \\ d_n \end{matrix}$ | $x$ =type→ $c$ iff $x$ type→ $d_1$ type→ $\cdots$ type→ $d_n$ | SelfDrivingTaxi —inter.→ Taxi SelfDriving |
| ENUMERATION | $c$ —one of→ $\begin{matrix} x_1 \\ \vdots \\ x_n \end{matrix}$ | $x$ =type→ $c$ iff $x$ $\in \{x_1, \ldots, x_n\}$ | EUState —one of→ Austria $\vdots$ Sweden |
| SOME VALUES | $c$ ⟨prop→ $p$, some→ $d$⟩ | $x$ =type→ $c$ iff there exists $a$ such that $x$ = $p$ → $a$ =type→ $d$ | EUCitizen ⟨prop→ nationality, some→ EUState⟩ |
| ALL VALUES | $c$ ⟨prop→ $p$, all→ $d$⟩ | $x$ =type→ $c$ iff for all $a$ with $x$ = $p$ → $a$ it holds that $a$ =type→ $d$ | Weightless ⟨prop→ has part, all→ Weightless⟩ |
| HAS VALUE | $c$ ⟨prop→ $p$, value→ $y$⟩ | $x$ =type→ $c$ iff $x$ = $p$ → $y$ | ChileanCitizen ⟨prop→ nationality, value→ Chile⟩ |
| HAS SELF | $c$ ⟨prop→ $p$, self→ true⟩ | $x$ =type→ $c$ iff $x$ = $p$ → $x$ | SelfDriving ⟨prop→ driver, self→ true⟩ |
| CARDINALITY $\star \in \{=, \leq, \geq\}$ | $c$ ⟨prop→ $p$, $\star$→ $n$⟩ | $x$ =type→ $c$ iff $\#\{a \mid x = p \to a\} \star n$ | Polyglot ⟨prop→ fluent, $\geq$→ 2⟩ |
| QUALIFIED CARDINALITY $\star \in \{=, \leq, \geq\}$ | $c$ ⟨prop→ $p$, class→ $d$, $\star$→ $n$⟩ | $x$ =type→ $c$ iff $\#\{a \mid x = p \to a =$type→ $d\} \star n$ | BinaryStarSystem ⟨prop→ body, class→ Star, $=$→ 2⟩ |

*4.1.5 Other features.* OWL supports other language features not previously discussed, including: *annotation properties*, which provide metadata about ontologies, such as versioning info; *datatype vs. object properties*, which distinguish properties that take datatype values from those that do not; and *datatype facets*, which allow for defining new datatypes by applying restrictions to existing datatypes, such as to define that places in Chile must have a *float between -66.0 and -110.0* as their value for the (datatype) property latitude. For more details we refer to the OWL 2 standard [228]. We will further discuss methodologies for the creation of ontologies in Section 6.5.

## 4.2 Semantics and Entailment

The conditions listed in the previous tables indicate how each feature should be interpreted. These conditions give rise to *entailments*, where, for example, in reference to the SYMMETRIC feature

of Table 3, the definition (nearby)—type→(Symmetric) and edge (Santiago)–nearby→(Santiago Airport) entail the edge (Santiago Airport)–nearby→(Santiago) according to the condition given for that feature. We now describe how these conditions lead to entailments.

*4.2.1 Model-theoretic semantics.* Each axiom described by the previous tables, when added to a graph, enforces some condition(s) on the interpretations that *satisfy* the graph. The interpretations that satisfy a graph are called *models* of the graph. Were we to consider only the base condition of the ASSERTION feature in Table 2, for example, then the models of a graph would be any interpretation such that for every edge (x)—y→(z) in the graph, there exists a relation (x)–y→(z) in the model. Given that there may be other relations in the model (under the OWA), the number of models of any such graph is infinite. Furthermore, given that we can map multiple nodes in the graph to one entity in the model (under the NUNA), any interpretation with (for example) the relation (a)–a→(a) is a model of any graph so long as for every edge (x)—y→(z) in the graph, it holds that (x) = (y) = (z) = (a) in the interpretation (in other words, the interpretation maps everything to (a)). As we add axioms with their associated conditions to the graph, we restrict models for the graph; for example, considering a graph with two edges – (x)—y→(z) and (y)–type→(Irreflexive) – the interpretation with (a)–a→(a), (x) = (y) = ... = (a) is no longer a model as it breaks the condition for the irreflexive axiom.

*4.2.2 Entailment.* We say that one graph *entails* another if and only if any model for the former graph is also a model of the latter graph. Intuitively this means that the latter graph says nothing new over the former graph and thus holds as a logical consequence of the former graph. For example, consider the graph (Santiago)–type→(City)–subc. of→(Place) and the graph (Santiago)–type→(Place). All models of the latter must have that (Santiago)–type→(Place), but so must all models of the former, which must have (Santiago)–type→(City)–subc. of→(Place) and further must satisfy the condition for SUBCLASS, which requires that (Santiago)–type→(Place) also hold. Hence we conclude that any model of the latter graph must be a model of the former graph, or, in other words, the former graph entails the latter graph.

*4.2.3 If–then vs. if-and-only-if semantics.* Consider the graph (nearby)–type→(Symmetric) and the graph (nearby)–inv. of→(nearby). They result in the same semantic conditions being applied in the domain graph, but does one entail the other? The answer depends on the semantics applied. Considering the axioms and conditions of Tables 2, we can consider two semantics. Under if–then semantics – IF **Axiom** matches data graph THEN **Condition** holds in domain graph – the graphs do not entail each other: though both graphs give rise to the same condition, this condition is not translated back into the axioms that describe it.[16] Conversely, under if-and-only-if semantics – **Axiom** matches data graph IF-AND-ONLY-IF **Condition** holds in domain graph – the graphs entail each other: both graphs give rise to the same condition, which is translated back into all possible axioms that describe it. Hence if-and-only-if semantics allows for entailing more axioms in the ontology language than if–then semantics. OWL generally applies an if-and-only-if semantics [228].

## 4.3 Reasoning

Unfortunately, given two graphs, deciding if the first entails the second – per the notion of entailment we have defined and for all of the ontological features listed in Tables 2–4 – is *undecidable*: no (finite) algorithm for such entailment can exist that halts on all inputs with the correct true/false answer [229]. However, we can provide practical reasoning algorithms for ontologies that (1) halt on any input ontology but may miss entailments, returning false instead of true, (2) always halt with the correct answer but only accept input ontologies with restricted features, or (3) only return

---

[16]Observe that (nearby)–type→(Symmetric) is a model of the first graph but not the second, while (nearby)–inv. of→(nearby) is a model of the second graph but not the first. Hence neither graph entails the other.

correct answers for any input ontology but may never halt on certain inputs. Though option (3) has been explored using, e.g., theorem provers for First Order Logic [441], options (1) and (2) are more commonly pursued using rules and/or Description Logics. Option (1) generally allows for more efficient and scalable reasoning algorithms and is useful where data are incomplete and having some entailments is valuable. Option (2) may be a better choice in domains – such as medical ontologies – where missing entailments may have undesirable outcomes.

*4.3.1 Rules.* One of the most straightforward ways to provide automated access to deductive knowledge is through *inference rules* (or simply *rules*) encoding IF–THEN-style consequences. A rule is composed of a *body* (IF) and a *head* (THEN). Both the body and head are given as graph patterns. A rule indicates that if we can replace the variables of the body with terms from the data graph and form a subgraph of a given data graph, then using the same replacement of variables in the head will yield a valid entailment. The head must typically use a subset of the variables appearing in the body to ensure that the conclusion leaves no variables unreplaced. Rules of this form correspond to (positive) Datalog [82] in databases, Horn clauses [303] in logic programming, etc.

Rules can be used to capture entailments under ontological conditions. In Table 5, we list some example rules for sub-class, sub-property, domain and range features [346]; these rules may be considered incomplete, not capturing, for example, that every class is a sub-class of itself, that every property is a sub-property of itself, etc. A more comprehensive set of rules for the OWL features of Tables 2–4 have been defined as OWL 2 RL/RDF [341]; these rules are likewise incomplete as such rules cannot fully capture negation (e.g., COMPLEMENT), existentials (e.g., SOME VALUES), universals (e.g., ALL VALUES), or counting (e.g., CARDINALITY and QUALIFIED CARDINALITY). Other rule languages have, however, been proposed to support additional such features, including existentials (see, e.g., Datalog$^\pm$ [33]), disjunction (see, e.g., Disjunctive Datalog [424]), etc.

Rules can be leveraged for reasoning in a number of ways. *Materialisation* refers to the idea of applying rules recursively to a graph, adding the conclusions generated back to the graph until a fixpoint is reached and nothing more can be added. The materialised graph can then be treated as any other graph. Although the efficiency and scalability of materialisation can be enhanced through optimisations like Rete networks [158], or using distributed frameworks like MapReduce [504], depending on the rules and the data, the materialised graph may become unfeasibly large to manage. Another strategy is to use rules for *query rewriting*, which given a query, will automatically extend the query in order to find solutions entailed by a set of rules; for example, taking the schema graph in Figure 11 and the rules in Table 5, the (sub-)pattern $\boxed{?x} \!-\! type \!\rightarrow\! \boxed{Event}$ in a given input query would be rewritten to the following disjunctive pattern evaluated on the original graph:

$$\boxed{?x}\!-\!type\!\rightarrow\!\boxed{Event} \cup \boxed{?x}\!-\!type\!\rightarrow\!\boxed{Festival} \cup \boxed{?x}\!-\!type\!\rightarrow\!\boxed{Periodic\ Market} \cup \boxed{?x}\!-\!venue\!\rightarrow\!\boxed{?y}$$

OWL 2 QL [341] is then a subset of OWL designed for query rewriting of this form. [19]
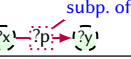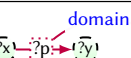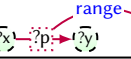
While rules can be used to (partially) capture ontological entailments, they can also be defined independently of an ontology language, capturing entailments for a given domain. In fact, some rules – such as the following – cannot be captured by the ontology features previously seen, as they do not support ways to infer relations from cyclical graph patterns (for computability reasons):

$$\boxed{?x}\!-\!flight\!\rightarrow\!\boxed{?y}\!-\!country\!\rightarrow\!\boxed{?z} \Rightarrow \boxed{?x}\!-\!domestic\ flight\!\rightarrow\!\boxed{?y}$$
$$country$$

Various languages allow for expressing rules over graphs – independently or alongside of an ontology language – including: Notation3 (N3) [38], Rule Interchange Format (RIF) [270], Semantic Web Rule Language (SWRL) [242], and SPARQL Inferencing Notation (SPIN) [277].

*4.3.2 Description Logics.* Description Logics (DLs) were initially introduced as a way to formalise the meaning of *frames* [333] and *semantic networks* [401]. Considering that semantic networks

Table 5. Example rules for sub-class, sub-property, domain, and range features

| Feature | Body | ⇒ | Head |
|---|---|---|---|
| SUBCLASS (I) | ?x —type→ ?c —subc. of→ ?d | ⇒ | ?x —type→ ?d |
| SUBCLASS (II) | ?c —subc. of→ ?d —subc. of→ ?e | ⇒ | ?c —subc. of→ ?e |
| SUBPROPERTY (I) | ?x —?p→ ?y, ?p —subp. of→ ?q | ⇒ | ?x —?q→ ?y |
| SUBPROPERTY (II) | ?p —subp. of→ ?q —subp. of→ ?r | ⇒ | ?p —subp. of→ ?r |
| DOMAIN | ?x —?p→ ?y, ?p —domain→ ?c | ⇒ | ?x —type→ ?c |
| RANGE | ?x —?p→ ?y, ?p —range→ ?c | ⇒ | ?y —type→ ?c |

are an early version of knowledge graphs, and the fact that DLs have heavily influenced the Web Ontology Language, DLs thus hold an important place in the logical formalisation of knowledge graphs. DLs form a family of logics rather than a particular logic. Initially, DLs were restricted fragments of First Order Logic (FOL) that permit decidable reasoning tasks, such as entailment checking [21]. Different DLs strike different balances between expressive power and computational complexity of reasoning. DLs would later be extended with features that go beyond FOL but are useful in the context of modelling graph data, such as transitive closure, datatypes, etc.

Description Logics are based on three types of elements: *individuals*, such as Santiago; *classes* (aka *concepts*) such as City; and *properties* (aka *roles*) such as flight. DLs then allow for making claims, known as *axioms*, about these elements. *Assertional axioms* can be either unary class relations on individuals, such as City(Santiago), or binary property relations on individuals, such as flight(Santiago,Arica). Such axioms form the *Assertional Box* (*A-Box*). DLs further introduce logical symbols to allow for defining *class axioms* (forming the *Terminology Box*, or *T-Box* for short), and *property axioms* (forming the *Role Box*, *R-Box*); for example, the class axiom City ⊑ Place states that the former class is a subclass of the latter one, while the property axiom flight ⊑ connectsTo states that the former property is a subproperty of the latter one. DLs may then introduce a rich set of logical symbols, not only for defining class and property axioms, but also defining new classes based on existing terms; as an example of the latter, we can define a class ∃nearby.Airport as the class of individuals that have some airport nearby. Noting that the symbol ⊤ is used in DLs to denote the class of all individuals, we can then add a class axiom ∃flight.⊤ ⊑ ∃nearby.Airport to state that individuals with an outgoing flight must have some airport nearby. Noting that the symbol ⊔ can be used in DL to define that a class is the union of other classes, we can further define that Airport ⊑ DomesticAirport ⊔ InternationalAirport, i.e., that an airport is either a domestic airport or an international airport (or both).

The similarities between these DL features and the OWL features previously outlined in Tables 2–4 are not coincidental: the OWL standard was heavily influenced by DLs, where, for example, the OWL 2 DL language is a fragment of OWL restricted so that entailment becomes decidable. As an example of a restriction, with DomesticAirport ⊑ = 1 destination ∘ country.⊤, we can define in DL syntax that domestic airports have flights destined to precisely one country (where p ∘ q denotes a chain of properties). However, counting chains is often disallowed in DLs to ensure decidability. In Appendix B.5.3, we present formal definitions for DL syntax and semantics, as well as notions of entailment. For further reading, we also refer to the textbook by Baader et al. [21].

Expressive DLs support complex entailments involving existentials, universals, counting, etc. A common strategy for deciding such entailments is to reduce entailment to *satisfiability*, which decides if an ontology is consistent or not [241].[17] Thereafter methods such as *tableau* can be used to check satisfiability, cautiously constructing models by completing them along similar lines to the materialisation strategy previously described, but additionally branching models in the case of disjunction, introducing new elements to represent existentials, etc. If any model is successfully "completed", the process concludes that the original definitions are satisfiable (see, e.g., [342]). Due to their prohibitive computational complexity [341] – where for example, disjunction may lead to an exponential number of branching possibilities – such reasoning strategies are not typically applied in the case of large-scale data, though they may be useful when modelling complex domains.

## 5 INDUCTIVE KNOWLEDGE

While deductive knowledge is characterised by precise logical consequences, inductively acquiring knowledge involves generalising patterns from a given set of input observations, which can then be used to generate novel but potentially imprecise predictions. For example, from a large data graph with geographical and flight information, we may observe the pattern that almost all capital cities of countries have international airports serving them, and hence predict that if Santiago is a capital city, it *likely* has an international airport serving it; however, the predictions drawn from this pattern do not hold for certain, where (e.g.) Vaduz, the capital city of Liechtenstein, has no (international) airport serving it. Hence predictions will often be associated with a level of confidence; for example, we may say that a capital has an international airport in $\frac{187}{195}$ of cases, offering a confidence of 0.959 for predictions made with that pattern. We then refer to knowledge acquired inductively as *inductive knowledge*, which includes both the models used to encode patterns, as well as the predictions made by those models. Though fallible, inductive knowledge can be highly valuable.

Several approaches can be employed for acquiring knowledge inductively from graphs, based on *supervised* or *unsupervised* methods. Supervised methods learn a function (aka *model*) to map a given set of example inputs to their labelled outputs; the model can then be applied to unlabelled inputs to predict their output. Given that manual labelling can be costly, some supervised methods rather find ways to generate the input–output pairs automatically from the input, which are then fed into a supervised process to learn a model; herein we refer to this approach as *self-supervision*. Alternatively, unsupervised processes do not require labelled input–output pairs, but rather apply a predefined function (typically statistical in nature) to map inputs to outputs.

In the case of unsupervised methods, there is a rich body of work on *graph analytics*, which uses well-known functions/algorithms to detect communities or clusters, find central nodes and edges, discover interesting paths, etc., in graphs/networks. Alternatively, *knowledge graph embeddings* can be used to learn a low-dimensional numeric model of a knowledge graph that (typically) maps input edges to an output *plausibility score* indicating the likelihood of the edge being true; such embeddings can be learnt in a self-supervised manner using positive edges and negative edges generated automatically from the knowledge graph itself. Rather than encoding graphs in a flat numerical representation, the structure of graphs can be directly leveraged for supervised learning, as explored in the context of *graph neural networks*, which learn models that are applied (possibly recursively) over neighbourhoods of the graph, typically to generate outputs for nodes. Finally, while the aforementioned techniques learn numerical models, *symbolic learning* can learn symbolic models – i.e., logical formulae – from a graph that can be combined with deductive reasoning to make predictions. We now discuss each of the four aforementioned techniques in turn.

---

[17]$G$ entails $G'$ if and only if $G \cup \mathrm{not}(G)$ is not satisfiable.
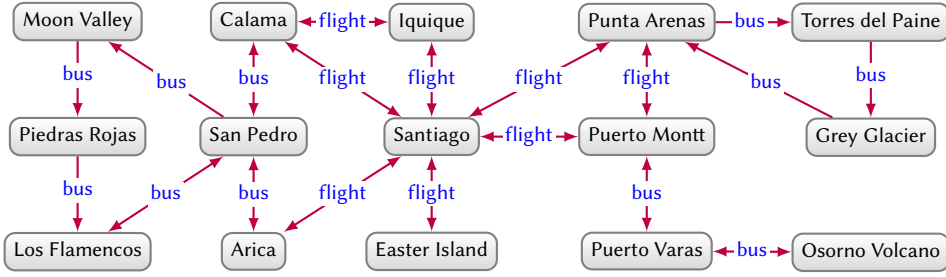
Fig. 21. Data graph representing transport routes in Chile

## 5.1 Graph Analytics

Analytics is the process of discovering, interpreting, and communicating meaningful patterns inherent to (typically large) data collections. Graph analytics is then the application of analytical processes to (typically large) graph data. The nature of graphs naturally lends itself to certain types of analytics that derive conclusions about nodes and edges based on the *topology* of the graph, i.e., how the nodes of the graph are connected. Graph analytics hence draws many of its techniques from related areas such as graph theory and network analysis, which have been used to study graphs that represent social networks, the Web, internet routing, transportation networks, ecosystems, protein–protein interactions, linguistic cooccurrences, and more besides [142].

Returning to the domain of our running example, the tourism board could use graph analytics to extract knowledge about, for instance: key transport hubs that serve many tourist attractions (centrality); groupings of attractions visited by the same tourists (community detection); attractions that may become unreachable in the event of strikes or other route failures (connectivity), or pairs of attractions that are similar to each other (node similarity). Given that such analytics will require a complex, large-scale graph, for the purposes of illustration, in Figure 21 we present a more concise example of some transportation connections in Chile directed towards popular touristic destinations. We first introduce a selection of key techniques that can be applied for graph analytics. We then discuss frameworks and languages that can be used to compute such analytics in practice. Given that many traditional graph algorithms are defined for unlabelled graphs, we then describe ways in which analytics can be applied over directed edge-labelled graphs. Finally we discuss the potential connections between graph analytics and querying and reasoning.

*5.1.1 Techniques.* A wide variety of techniques can be applied for graph analytics. In the following we will enumerate some of the main techniques – as recognised, for example, by the survey of Iosup et al. [250] – that can be invoked in this setting.

- *Centrality:* aims to identify the most important (aka *central*) nodes or edges of a graph. Specific node centrality measures include *degree*, *betweenness*, *closeness*, *Eigenvector*, *PageRank*, *HITS*, *Katz*, among others. Betweenness centrality can also be applied to edges. A node centrality measure would allow, e.g., to predict the transport hubs in Figure 21, while edge centrality would allow us to find the edges on which many shortest routes depend for predicting traffic.
- *Community detection:* aims to identify *communities* in a graph, i.e., sub-graphs that are more densely connected internally than to the rest of the graph. Community detection algorithms, such as *minimum-cut algorithms*, *label propagation*, *Louvain modularity*, etc. enable discovering such communities. Community detection applied to Figure 21 may, for example, detect a community to the left (referring to the north of Chile), to the right (referring to the south of Chile), and perhaps also the centre (referring to cities with airports).

- *Connectivity:* aims to estimate how well-connected the graph is, revealing, for instance, the resilience and (un)reachability of elements of the graph. Specific techniques include measuring *graph density* or *$k$-connectivity*, detecting *strongly connected components* and *weakly connected components*, computing *spanning trees* or *minimum cuts*, etc. In the context of Figure 21, such analysis may tell us that routes to (Osorno Volcano) and (Piedras Rojas) are the most "brittle", becoming disconnected from the main hubs if one of two bus routes fail.
- *Node similarity:* aims to find nodes that are similar to other nodes by virtue of how they are connected within their neighbourhood. Node similarity metrics may be computed using *structural equivalence*, *random walks*, *diffusion kernels*, etc. These methods provide an understanding of what connects nodes, and, thereafter, in what ways they are similar. In the context of Figure 21, such analysis may tell us that (Calama) and (Arica) are similar nodes based on both having return flights to (Santiago) and return buses to (San Pedro).

While the previous techniques accept a graph alone as input,[18] other forms of graph analytics may further accept a node, a pair of nodes, etc., along with the graph.

- *Path finding:* aims to find paths in a graph, typically between pairs of nodes given as input. Various technical definitions exist that restrict the set of valid paths between such nodes, including simple paths that do not visit the same node twice, shortest paths that visit the fewest number of edges, or – as previously discussed in Section 2.2 – regular path queries that restrict the labels of edges that can be traversed by the path [14]. We could use such algorithms to find, for example, the shortest path(s) in Figure 21 from (Torres del Paine) to (Moon Valley).

Most such techniques have been proposed and studied for simple graphs or directed graphs without edge labels. We will discuss their application to more complex graph models – and how they can be combined with other techniques such as reasoning and querying – later in Section 5.1.3.

*5.1.2 Frameworks.* Various frameworks have been proposed for large-scale graph analytics, often in a distributed (cluster) setting. Amongst these we can mention Apache Spark (GraphX) [114, 529], GraphLab [306], Pregel [314], Signal–Collect [478], Shark [530], etc. These *graph parallel frameworks* apply a *systolic abstraction* [285] based on a directed graph, where nodes are processors that can send messages to other nodes along edges. Computation is then iterative, where in each iteration, each node reads messages received through inward edges (and possibly its own previous state), performs a computation, and then sends messages through outward edges based on the result. These frameworks then define the systolic computational abstraction on top of the data graph being processed: nodes and edges in the data graph become nodes and edges in the systolic graph. We refer to Appendix B.6.1 for more formal details on graph parallel frameworks.

To take an example, assume we wish to compute the places that are most (or least) easily reached by the routes shown in the graph of Figure 21. A good way to measure this is using centrality, where we choose PageRank [369], which computes the probability of a tourist randomly following the routes shown in the graph being at a particular place after a given number of "hops". We can implement PageRank on large graphs using a graph parallel framework. In Figure 22, we provide an example of an iteration of PageRank for an illustrative sub-graph of Figure 21. The nodes are initialised with a score of $\frac{1}{|V|} = \frac{1}{6}$, where we assume the tourist to have an equal chance of starting at any point. In the *message phase* (Msg), each node $v$ passes a score of $\frac{d\mathrm{R}_i(v)}{|E(v)|}$ on each of its outgoing edges, where we denote by $d$ a constant damping factor used to ensure convergence (typically $d = 0.85$, indicating the probability that a tourist randomly "jumps" to any place), by $\mathrm{R}_i(v)$ the score of node $v$ in iteration $i$ (the probability of the tourist being at node $v$ after $i$ hops), and by $|E(v)|$

---

[18]Node similarity can be run over an entire graph to find the $k$ most similar nodes for each node, or can also be run for a specific node to find its most similar nodes.
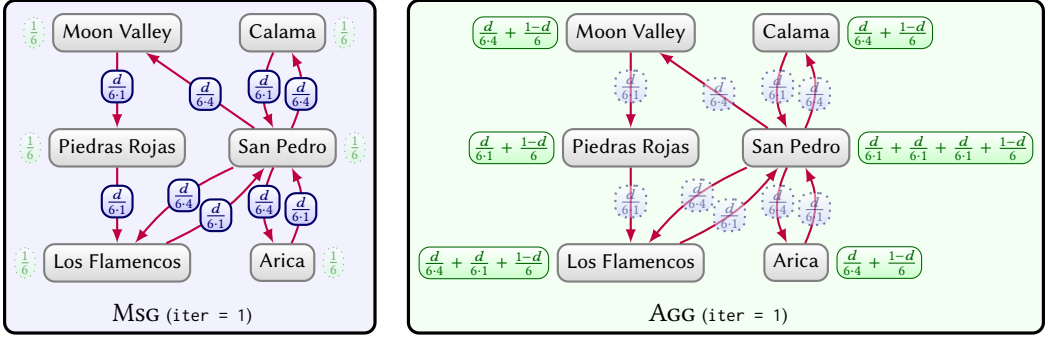
Fig. 22. Example of a systolic iteration of PageRank for a sample sub-graph of Figure 21

the number of outgoing edges of $v$. The aggregation phase (AGG) for $v$ then sums all incoming messages received along with its constant share of the damping factor ($\frac{1-d}{|V|}$) to compute $R_{i+1}(v)$. We then proceed to the message phase of the next iteration, continuing until some termination criterion is reached (e.g., iteration count or residual threshold, etc.) and final scores are output.

While the given example is for PageRank, the systolic abstraction is general enough to support a wide variety of graph analytics, including those previously mentioned. An algorithm in this framework consists of the functions to compute message values in the *message phase* (MSG), and to accumulate the messages in the aggregation phase (AGG). The framework will take care of distribution, message passing, fault tolerance, etc. However, such frameworks – based on message passing between neighbours – have limitations: not all types of analytics can be expressed in such frameworks [531].[19] Hence frameworks may allow additional features, such as a *global step* that performs a global computation on all nodes, making the result available to each node [314]; or a *mutation step* that allows for adding or removing nodes and edges during processing [314].

*5.1.3 Analytics on data graphs.* As aforementioned, most analytics presented thus far are, in their "native" form, applicable for undirected or directed graphs without the *edge meta-data* – i.e., edge labels or property–value pairs – typical of graph data models.[20] A number of strategies can be applied to make data graphs subject to analytics of this form:

- *Projection* involves simply "projecting" an undirected or directed graph by optionally selecting a sub-graph from the data graph from which all edge meta-data are dropped; for example, Figure 21 may be the result of extracting the sub-graph induced by the edge labels bus and flight from a larger data graph, where the labels are then dropped to create a directed graph.
- *Weighting* involves converting edge meta-data into numerical values according to some function. Many of the aforementioned techniques are easily adapted to the case of weighted (directed) graphs; for example, we could consider weights on the graph of Figure 21 denoting trip duration (or price, traffic, etc.), and then compute the shortest paths adding the duration of each leg.[21] In the absence of external weights, we may rather map edge labels to weights, assigning the same weight to all flight edges, to all bus edges, etc., based on some criteria.

---

[19]Formally Xu et al. [531] have shown that such frameworks are as powerful as the (incomplete) Weisfeiler–Lehman (WL) graph isomorphism test – based on recursively hashing neighbouring hashes – for distinguishing graph structures.

[20]We remark that in the case of property graphs, property–value pairs on nodes can be converted by mapping values to nodes and properties to edges with the corresponding label.

[21]Other forms of analytics are possible if we assume the graph is weighted; for example, if we annotated the graph of Figure 21 with probabilities of tourists moving from one place to the next, we could leverage *Markov processes* to understand features such as reducibility, periodicity, transience, recurrence, ergodicity, steady-states, etc., of the routes [133].
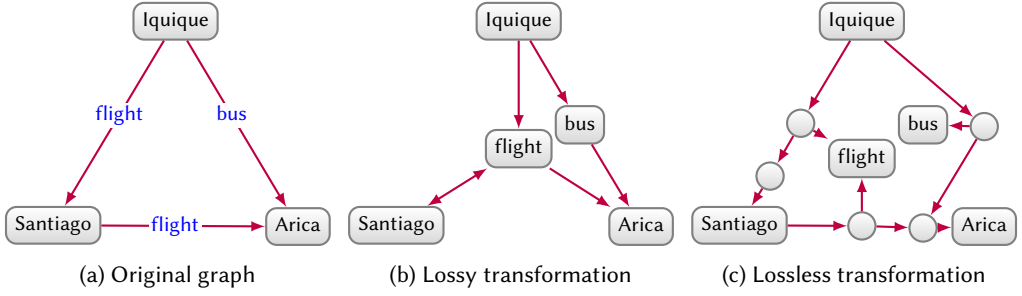
Fig. 23. Transformations from a directed edge-labelled graph to a directed graph

- *Transformation* involves transforming the graph to a lower arity model. A transformation may be *lossy*, meaning that the original graph cannot be recovered; or *lossless*, meaning that the original graph can be recovered. Figure 23 provides an example of a lossy and lossless transformation from a directed edge-labelled graph to directed graphs. In the lossy transformation, we cannot tell, for example, if the original graph contained the edge Iquique –flight→ Santiago or Iquique –flight→ Arica, etc. The lossless transformation must introduce new nodes (similar to reification) to maintain information about directed labelled edges. Both transformed graphs further attempt to preserve the directionality of the original graph.
- *Customisation* involves changing the analytical procedure to incorporate edge meta-data, such as was the case for path finding based on path expressions. Other examples might include structural measures for node similarity that not only consider common neighbours, but also common neighbours connected by edges with the same label, or aggregate centrality measures that capture the importance of edges grouped by label, etc.

The results of an analytical process may change drastically depending on which of the previous strategies are chosen to prepare the data for analysis. This choice may be a non-trivial one to make *a priori* and may require empirical validation. More study is required to more generally understand the effects of such strategies on the results of different analytical techniques.

*5.1.4   Analytics with queries.* As discussed in Section 2.2, various languages for querying graphs have been proposed [14]. One may consider a variety of ways in which query languages and analytics can complement each other. First, we may consider using query languages to project or transform a graph suitable for a particular analytical task, such as to extract the graph of Figure 21 from a larger data graph. Query languages such as SPARQL [207], Cypher [159], and G-CORE [13] allow for outputting graphs, where such queries can be used to select sub-graphs for analysis. These languages can also express some limited (non-recursive) analytics, where aggregations can be used to compute degree centrality, for example; they may also have some built-in analytical support, where, for example, Cypher [159] allows for finding shortest paths. In the other direction, analytics can contribute to the querying process in terms of *optimisations*, where, for example, analysis of connectivity may suggest how to better distribute a large data graph over multiple machines for querying using, e.g., *minimum cuts* [6, 253]. Analytics have also been used to *rank* query results over large graphs [146, 516], selecting the most important results for presentation to the user.

In some use-cases we may further wish to interleave querying and analytical processes. For example, from the full data graph collected by the tourist board, consider an upcoming airline strike where the board wishes to find *the events during the strike with venues in cities unreachable from Santiago by public transport due to the strike*. Hypothetically, we could use a query to extract the transport network excluding the airline's routes (assuming, per Figure 3 that the airline information

is available), use analytics to extract the strongly connected component containing Santiago, and finally use a query to find events in cities not in the Santiago component on the given dates.[22] While one could solve this task using an imperative language such as Gremlin [420], GraphX [529], or R [491], more declarative languages are also being explored to more easily express such tasks, with proposals including the extension of graph query languages with recursive capabilities [43, 414],[23] combining linear algebra with relational (query) algebra [244], and so forth.

*5.1.5  Analytics with entailment.* Knowledge graphs are often associated with a semantic schema or ontology that defines the semantics of domain terms, giving rise to entailments (per Section 4). Applying analytics with or without such entailments – e.g., before or after materialisation – may yield radically different results. For example, observe that an edge (Santa Lucía)–hosts►(EID15) is semantically equivalent to an edge (EID15)-venue►(Santa Lucía) once the inverse axiom (hosts)-inv. of►(venue) is invoked; however, these edges are far from equivalent from the perspective of analytical techniques that consider edge direction, for which including one type of edge, or the other, or both, may have a major bearing on the final results. To the best of our knowledge, the combination of analytics and entailment has not been well-explored, leaving open interesting research questions. Along these lines, it may be of interest to explore *semantically-invariant analytics* that yield the same results over semantically-equivalent graphs (i.e., graphs that entail one another), thus analysing the semantic content of the knowledge graph rather than simply the topological features of the data graph; for example, semantically-invariant analytics would yield the same results over a graph containing the inverse axiom (hosts)-inv. of►(venue) and a number of hosts edges, the same graph but where every hosts edge is replaced by an inverse venue edge, and the union of both graphs.

## 5.2  Knowledge Graph Embeddings

Methods for machine learning have gained significant attention in recent years. In the context of knowledge graphs, machine learning can either be used for *refining* the knowledge graph itself, which involves predicting new edges and/or identifying erroneous edges (discussed further in Section 8); or for *downstream tasks*, where the knowledge graph is used to train models for classification, recommendation, regression, etc., in the application domain (discussed further in Section 10). However, many traditional machine learning techniques assume dense numeric input representations in the form of vectors, which is quite distinct from how graphs are usually expressed. So how can graphs – or nodes, edges, etc., thereof – be encoded as numeric vectors?

A first attempt to represent a graph using vectors would be to use a *one-hot encoding*, generating a vector for each node of length $|L| \cdot |V|$ – with $|V|$ the number of nodes in the input graph and $|L|$ the number of edge labels – placing a one at the corresponding index to indicate the existence of the respective edge in the graph, or zero otherwise. Such a representation will, however, typically result in large and sparse vectors, which will be detrimental for most machine learning models.

The main goal of knowledge graph embedding techniques is to create a dense representation of the graph (i.e., *embed* the graph) in a continuous, low-dimensional vector space that can then be used for machine learning tasks. The dimensionality $d$ of the embedding is fixed and typically low (often, e.g., $50 \geq d \geq 1000$). Typically the graph embedding is composed of an *entity embedding* for each node: a vector with $d$ dimensions that we denote by $\mathbf{e}$; and a *relation embedding* for each edge label: (typically) a vector with $d$ dimensions that we denote by $\mathbf{r}$. The overall goal of these vectors is to abstract and preserve latent structures in the graph. There are many ways in which this notion of an embedding can be instantiated. Most commonly, given an edge (s)-p►(o), a specific

---

[22]Such a task could not be solved in a single query using regular path queries as such expressions would not be capable of filtering edges representing flights of a particular airline.

[23]Recursive query languages become Turing complete assuming one can also express operations on binary arrays.

embedding approach defines a *scoring function* that accepts $\mathbf{e}_s$ (the entity embedding of node ⓢ), $\mathbf{r}_p$ (the entity embedding of edge label ⓟ) and $\mathbf{e}_o$ (the entity embedding of node ⓞ) and computes the *plausibility* of the edge: how likely it is to be true. Given a data graph, the goal is then to compute the embeddings of dimension $d$ that maximise the plausibility of positive edges (typically edges in the graph) and minimise the plausibility of negative examples (typically edges in the graph with a node or edge label changed such that they are no longer in the graph) according to the given scoring function. The resulting embeddings can then be seen as models learnt through self-supervision that encode (latent) features of the graph, mapping input edges to output plausibility scores.

Embeddings can then be used for a number of low-level tasks involving the nodes and edge-labels of the graph from which they were computed. First, we can use the plausibility scoring function to assign a confidence to edges that may, for example, have been extracted from an external source (discussed later in Section 6). Second, the plausibility scoring function can be used to complete edges with missing nodes/edge labels for the purposes of link prediction (discussed later in Section 8); for example, in Figure 21, we might ask which nodes in the graph are likely to complete the edge (Grey Glacier)–bus→(?), where – aside from (Punta Arenas), which is already given – we might intuitively expect (Torres del Paine) to be a plausible candidate. Third, embedding models will typically assign similar vectors to similar nodes and similar edge-labels, and thus they can be used as the basis of similarity measures, which may be useful for finding duplicate nodes that refer to the same entity, or for the purposes of providing recommendations (discussed later in Section 10).

A wide range of knowledge graph embedding techniques have been proposed [519], where our goal here is to provide a high-level introduction to some of the most popular techniques proposed thus far. First we discuss *translational models* that adopt a geometric perspective whereby relation embeddings translate subject entities to object entities in the low-dimensional space. We then describe *tensor decomposition models* that extract latent factors approximating the graph's structure. Thereafter we discuss *neural models* that use neural networks to train embeddings that provide accurate plausibility scores. Finally, we discuss *language models* that leverage existing word embedding techniques, proposing ways of generating graph-like analogues for their expected (textual) inputs. A more formal treatment of these models is provided in Appendix B.6.2.

*5.2.1 Translational models. Translational models* interpret edge labels as transformations from subject nodes (aka the *source* or *head*) to object nodes (aka the *target* or *tail*); for example, in the edge (San Pedro)–bus→(Moon Valley), the edge label bus is seen as transforming (San Pedro) to (Moon Valley), and likewise for other bus edges. The most elementary approach in this family is TransE [59]. Over all positive edges ⓢ–ⓟ→ⓞ, TransE learns vectors $\mathbf{e}_s$, $\mathbf{r}_p$, and $\mathbf{e}_o$ aiming to make $\mathbf{e}_s + \mathbf{r}_p$ as close as possible to $\mathbf{e}_o$. Conversely, if the edge is a negative example, TransE attempts to learn a representation that keeps $\mathbf{e}_s + \mathbf{r}_p$ away from $\mathbf{e}_o$. To illustrate, Figure 24 provides a toy example of two-dimensional ($d = 2$) entity and relation embeddings computed by TransE. We keep the orientation of the vectors similar to the original graph for clarity. For any edge ⓢ–ⓟ→ⓞ in the original graph, adding the vectors $\mathbf{e}_s + \mathbf{r}_p$ should approximate $\mathbf{e}_o$. In this toy example, the vectors correspond precisely where, for instance, adding the vectors for (Licantén) ($\mathbf{e}_L$) and west of ($\mathbf{r}_{wo}$) gives a vector corresponding to (Curicó) ($\mathbf{e}_C$). We can use these embeddings to predict edges (among other tasks); for example, in order to predict which node in the graph is most likely to be west of (Antofagasta) ($\mathbf{A}$), by computing $\mathbf{e}_A + \mathbf{r}_{wo}$ we find that the resulting vector (dotted in Figure 24c) is closest to $\mathbf{e}_T$, thus predicting (Toconao) ($\mathbf{T}$) to be the most *plausible* such node.

Aside from this toy example, TransE can be too simplistic; for example, in Figure 21, bus not only transforms (San Pedro) to (Moon Valley), but also to (Arica), (Calama), and so forth. TransE will, in this case, aim to give similar vectors to all such target locations, which may not be feasible given other edges. TransE will also tend to assign cyclical relations a zero vector, as the directional components

(a) Original graph      (b) Relation embeddings      (c) Entity embeddings
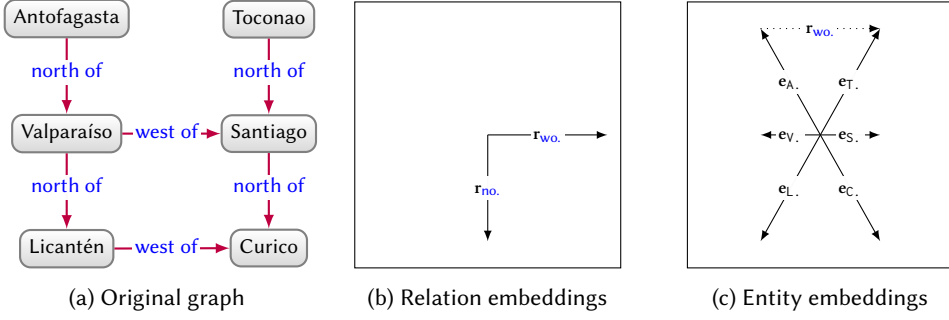
Fig. 24. Toy example of two-dimensional relation and entity embeddings learnt by TransE; the entity embeddings use abbreviations and include an example of vector addition to predict what is west of Antofagasta

will tend to cancel each other out. To resolve such issues, many variants of TransE have been investigated. Amongst these, for example, TransH [521] represents different relations using distinct hyperplanes, where for the edge ⓢ–ⓟ→ⓞ, ⓢ is first projected onto the hyperplane of ⓟ before the translation to ⓞ is learnt (uninfluenced by edges with other labels for ⓢ and for ⓞ). TransR [256] generalises this approach by projecting ⓢ and ⓞ into a vector space specific to ⓟ, which involves multiplying the entity embeddings for ⓢ and ⓞ by a projection matrix specific to ⓟ. TransD [256] simplifies TransR by associating entities and relations with a second vector, where these secondary vectors are used to project the entity into a relation-specific vector space. For discussion of other translational models, we refer to the survey by Wang et al. [519].

*5.2.2 Tensor decomposition models.* A second approach to derive graph embeddings is to apply methods based on *tensor decomposition*. A *tensor* is a multidimensional numeric field that generalises scalars (0-order tensors), vectors (1-order tensors) and matrices (2-order tensors) towards arbitrary dimension/order. Tensors have become a widely used abstraction for machine learning [402]. Tensor decomposition involves decomposing a tensor into more "elemental" tensors (e.g., of lower order) from which the original tensor can be recomposed (or approximated) by a fixed sequence of basic operations. These elemental tensors can be viewed as capturing *latent factors* underlying the information contained in the original tensor. There are many approaches to tensor decomposition, where we will now briefly introduce the main ideas behind *rank decompositions* [402].

Leaving aside graphs momentarily, consider an $(a, b)$-matrix (i.e., an order-2 tensor) $\mathbf{C}$, where $a$ is the number of cities in Chile, $b$ is the number of months in a year, and each element $(\mathbf{C})_{ij}$ denotes the average temperature of the $i^{\text{th}}$ city in the $j^{\text{th}}$ month. Noting that Chile is a long, thin country – ranging from subpolar climates in the south, to a desert climate in the north – we may find a decomposition of $\mathbf{C}$ into two vectors representing latent factors – specifically $\mathbf{x}$ (with $m$ elements) giving lower values for cities with lower latitude, and $\mathbf{y}$ (with $n$ elements), giving lower values for months with lower temperatures – such that computing the outer product[24] of the two vectors approximates $\mathbf{C}$ reasonably well: $\mathbf{x} \otimes \mathbf{y} \approx \mathbf{C}$. In the (unlikely) case that there exist vectors $\mathbf{x}$ and $\mathbf{y}$ such that that $\mathbf{C}$ is precisely the outer product of two vectors ($\mathbf{x} \otimes \mathbf{y} = \mathbf{C}$) we call $\mathbf{C}$ a rank-1 matrix; we can then precisely encode $\mathbf{C}$ using $m + n$ values rather than $m \times n$ values. Most times, however, to get precisely $\mathbf{C}$, we will need to sum multiple rank-1 matrices, where the rank $r$ of $\mathbf{C}$ is the minimum number of rank-1 matrices that need to be summed to derive precisely $\mathbf{C}$, such that

---

[24]The outer product of two (column) vectors $\mathbf{x}$ of length $m$ and $\mathbf{y}$ of length $n$, denoted $\mathbf{x} \otimes \mathbf{y}$, is defined as $\mathbf{xy}^{\mathrm{T}}$, yielding an $(m, n)$-matrix $\mathbf{M}$ such that $(\mathbf{M})_{ij} = (\mathbf{x})_i \cdot (\mathbf{y})_j$. Analogously, the outer product of $k$ vectors is a $k$-order tensor.
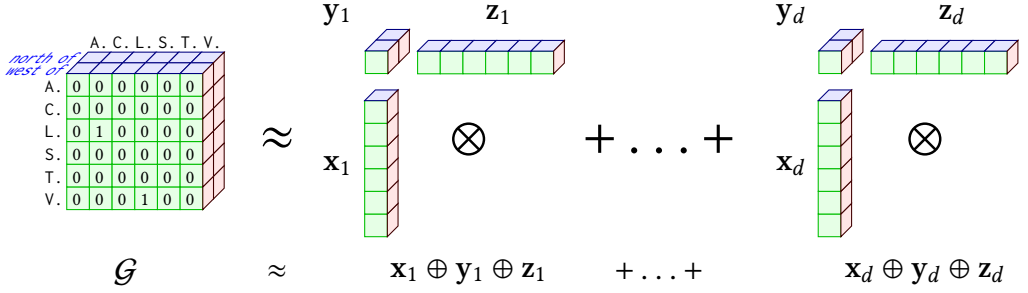
Fig. 25. Abstract illustration of a CP $d$-rank decomposition of a tensor representing the graph of Figure 24a

$\mathbf{x}_1 \otimes \mathbf{y}_1 + \ldots \mathbf{x}_r \otimes \mathbf{y}_r = \mathbf{C}$. In the temperature example, $\mathbf{x}_2 \otimes \mathbf{y}_2$ might correspond to a correction for altitude, $\mathbf{x}_3 \otimes \mathbf{y}_3$ for higher temperature variance further south, etc. A (low) rank decomposition of a matrix then sets a limit $d$ on the rank and computes the vectors $(\mathbf{x}_1, \mathbf{y}_1, \ldots, \mathbf{x}_d, \mathbf{y}_d)$ such that $\mathbf{x}_1 \otimes \mathbf{y}_1 + \ldots + \mathbf{x}_d \otimes \mathbf{y}_d$ gives the best $d$-rank approximation of $\mathbf{C}$. Noting that to generate $n$-order tensors we need to compute the outer product of $n$ vectors, we can generalise this idea towards low rank decomposition of tensors; this method is called Canonical Polyadic (CP) decomposition [225]. For example, we might have a 3-order tensor $\mathcal{C}$ containing monthly temperatures for Chilean cities *at four different times of day*, which could be approximated with $\mathbf{x}_1 \otimes \mathbf{y}_1 \otimes \mathbf{z}_1 + \ldots \mathbf{x}_d \otimes \mathbf{y}_d \otimes \mathbf{z}_d$ (e.g., $\mathbf{x}_1$ might be a latitude factor, $\mathbf{y}_1$ a monthly variation factor, and $\mathbf{z}_1$ a daily variation factor, and so on). Various algorithms then exist to compute (approximate) CP decompositions, including Alternating Least Squares, Jennrich's Algorithm, and the Tensor Power method [402].

Returning to graphs, similar principles can be used to decompose a graph into vectors, thus yielding embeddings. In particular, a graph can be encoded as a one-hot order-3 tensor $\mathcal{G}$ with $|V| \times |L| \times |V|$ elements, where the element $(\mathcal{G})_{ijk}$ is set to one if the $i^{\text{th}}$ node links to the $k^{\text{th}}$ node with an edge having the $j^{\text{th}}$ label, or zero otherwise. As previously mentioned, such a tensor will typically be very large and sparse, where rank decompositions are thus applicable. A CP decomposition [225] would compute a sequence of vectors $(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1, \ldots, \mathbf{x}_d, \mathbf{y}_d, \mathbf{z}_d)$ such that $\mathbf{x}_1 \otimes \mathbf{y}_1 \otimes \mathbf{z}_1 + \ldots + \mathbf{x}_d \otimes \mathbf{y}_d \otimes \mathbf{z}_d \approx \mathcal{G}$. We illustrate this scheme in Figure 25. Letting $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ denote the matrices formed by $\begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_d \end{bmatrix}, \begin{bmatrix} \mathbf{y}_1 & \cdots & \mathbf{y}_d \end{bmatrix}, \begin{bmatrix} \mathbf{z}_1 & \cdots & \mathbf{z}_d \end{bmatrix}$, respectively, with each vector forming a column of the corresponding matrix, then we could then extract the $i^{\text{th}}$ row of $\mathbf{Y}$ as an embedding for the $i^{\text{th}}$ relation, and the $j^{\text{th}}$ rows of $\mathbf{X}$ and $\mathbf{Z}$ as *two* embeddings for the $j^{\text{th}}$ entity. However, knowledge graph embeddings typically aim to assign *one* vector to each entity.

DistMult [533] is a seminal method for computing knowledge graph embeddings based on rank decompositions, where each entity and relation is associated with a vector of dimension $d$, such that for an edge ⓢ–ₚ→ⓞ, a plausibility scoring function $\sum_{i=1}^{d} (\mathbf{e}_s)_i (\mathbf{r}_p)_i (\mathbf{e}_o)_i$ is defined, where $(\mathbf{e}_s)_i$, $(\mathbf{r}_p)_i$ and $(\mathbf{e}_o)_i$ denote the $i^{\text{th}}$ elements of vectors $\mathbf{e}_s, \mathbf{r}_p, \mathbf{e}_o$, respectively. The goal, then, is to learn vectors for each node and edge label that maximise the plausibility of positive edges and minimise the plausibility of negative edges. This approach equates to a CP decomposition of the graph tensor $\mathcal{G}$, but where entities have one vector that is used twice: $\mathbf{x}_1 \otimes \mathbf{y}_1 \otimes \mathbf{x}_1 + \ldots + \mathbf{x}_d \otimes \mathbf{y}_d \otimes \mathbf{x}_d \approx \mathcal{G}$. A weakness of this approach is that per the scoring function, the plausibility of ⓢ–ₚ→ⓞ will always be equal to that of ⓞ–ₚ→ⓢ; in other words, DistMult does not consider edge direction.

Rather than use a vector as a relation embedding, RESCAL [364] uses a matrix, which allows for combining values from $\mathbf{e}_s$ and $\mathbf{e}_o$ across all dimensions, and thus can capture (e.g.) edge direction. However, RESCAL incurs a higher cost in terms of space and time than DistMult. HolE [363] uses vectors for relation and entity embeddings, but proposes to use the *circular correlation operator* –

which takes sums along the diagonals of the outer product of two vectors – to combine them. This operator is not commutative, and can thus consider edge direction. ComplEx [499], on the other hand, uses a complex vector (i.e., a vector containing complex numbers) as a relational embedding, which similarly allows for breaking the aforementioned symmetry of DistMult's scoring function while keeping the number of parameters low. SimplE [266] rather proposes to compute a standard CP decomposition computing two initial vectors for entities from $\mathbf{X}$ and $\mathbf{Z}$ and then averaging terms across $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ to compute the final plausibility scores. TuckER [27] employs a different type of decomposition – called a Tucker Decomposition [501], which computes a smaller "core" tensor $\mathcal{T}$ and a sequence of three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, such that $\mathcal{G} \approx \mathcal{T} \otimes \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}$ – where entity embeddings are taken from $\mathbf{A}$ and $\mathbf{C}$, while relation embeddings are taken from $\mathbf{B}$. Of these approaches, TuckER [27] currently provides state-of-the-art results on standard benchmarks.

*5.2.3 Neural models.* A limitation of the previously discussed approaches is that they assume either linear (preserving addition and scalar multiplication) or bilinear (e.g., matrix multiplication) operations over embeddings to compute plausibility scores. A number of approaches rather use neural networks to learn embeddings with non-linear scoring functions for plausibility.

One of the earliest proposals of a neural model was Semantic Matching Energy (SME) [185], which learns parameters (aka weights: $\mathbf{w}$, $\mathbf{w}'$) for two functions – $f_{\mathbf{w}}(\mathbf{e}_s, \mathbf{r}_p)$ and $g_{\mathbf{w}'}(\mathbf{e}_o, \mathbf{r}_p)$ – such that the dot product of the result of both functions – $f_{\mathbf{w}}(\mathbf{e}_s, \mathbf{r}_p) \cdot g_{\mathbf{w}'}(\mathbf{e}_o, \mathbf{r}_p)$ – gives the plausibility score. Both linear and bilinear variants of $f_{\mathbf{w}}$ and $g_{\mathbf{w}'}$ are proposed. Another early proposal was Neural Tensor Networks (NTN) [463], which rather proposes to maintain a tensor $\mathcal{W}$ of internal weights, such that the plausibility score is computed by a complex function that combines the outer product $\mathbf{e}_s \otimes \mathcal{W} \otimes \mathbf{e}_o$ with a standard neural layer over $\mathbf{e}_s$ and $\mathbf{e}_o$, which in turn is combined with $\mathbf{r}_p$, to produce a plausibility score. The use of the tensor $\mathcal{W}$ results in a high number of parameters, which limits scalability [519]. Multi Layer Perceptron (MLP) [126] is a simpler model, where $\mathbf{e}_s$, $\mathbf{r}_p$ and $\mathbf{e}_o$ are concatenated and fed into a hidden layer to compute the plausibility score.

A number of more recent approaches have proposed using convolutional kernels in their models. ConvE [122] proposes to generate a matrix from $\mathbf{e}_s$ and $\mathbf{r}_p$ by "wrapping" each vector over several rows and concatenating both matrices. The concatenated matrix serves as the input for a set of (2D) convolutional layers, which returns a feature map tensor. The feature map tensor is vectorised and projected into $d$ dimensions using a parametrised linear transformation. The plausibility score is then computed based on the dot product of this vector and $\mathbf{e}_o$. A disadvantage of ConvE is that by wrapping vectors into matrices, it imposes an artificial two-dimensional structure on the embeddings. HypER [26] is a similar model using convolutions, but avoids the need to wrap vectors into matrices. Instead, a fully connected layer (called the "hypernetwork") is applied to $\mathbf{r}_p$ and used to generate a matrix of relation-specific convolutional filters. These filters are applied directly to $\mathbf{e}_s$ to give a feature map, which is vectorised. The same process is then applied as in ConvE: the resulting vector is projected into $d$ dimensions, and a dot product applied with $\mathbf{e}_o$ to produce the plausibility score. The resulting model is shown to outperform ConvE on standard benchmarks [26].

The presented approaches strike different balances in terms of expressivity and the number of parameters than need to be trained. While more expressive models, such as NTN, may better fit more complex plausibility functions over lower dimensional embeddings by using more hidden parameters, simpler models, such as that proposed by Dong et al. [126], and convolutional networks [26, 122] that enable parameter sharing by applying the same (typically small) kernels over different regions of a matrix, require handling fewer parameters overall and are more scalable.

*5.2.4 Language models.* Embedding techniques were first explored as a way to represent natural language within machine learning frameworks, with word2vec [330] and GloVe [383] being two seminal approaches. Both approaches compute embeddings for words based on large corpora of text

such that words used in similar contexts (e.g., "frog", "toad") have similar vectors. Word2vec uses neural networks trained either to predict the current word from surrounding words (*continuous bag of words*), or to predict the surrounding words given the current word (*continuous skip-gram*). GloVe rather applies a regression model over a matrix of co-occurrence probabilities of word pairs. Embeddings generated by both approaches are widely used in natural language processing tasks.

Another approach for graph embeddings is thus to leverage proven approaches for language embeddings. However, while a graph consists of an unordered set of sequences of three terms (i.e., a set of edges), text in natural language consists of arbitrary-length sequences of terms (i.e., sentences of words). Along these lines, RDF2Vec [416] performs (biased [91]) random walks on the graph and records the paths (the sequence of nodes and edge labels traversed) as "sentences", which are then fed as input into the word2vec [330] model. An example of such a path extracted from Figure 21 might be, for example, San Pedro –bus→ Calama –flight→ Iquique –flight→ Santiago, where the paper experiments with 500 paths of length 8 per entity. RDF2Vec also proposes a second mode where sequences are generated for nodes from canonically-labelled sub-trees of which they are a root node, where the paper experiments with sub-trees of depth 1 and 2. Conversely, KGloVe [92] is based on the GloVe model. Much like how the original GloVe model [383] considers words that co-occur frequently in windows of text to be more related, KGloVe uses personalised PageRank[25] to determine the most related nodes to a given node, whose results are then fed into the GloVe model.

*5.2.5    Entailment-aware models.* The embeddings thus far consider the data graph alone. But what if an ontology or set of rules is provided? Such deductive knowledge could be used to improve the embeddings. One approach is to use constraint rules to refine the predictions made by embeddings; for example, Wang et al. [520] use functional and inverse-functional definitions as constraints (under UNA) such that, for example, if we define that an event can have at most one value for venue, this is used to lower the plausibility of edges that would assign multiple venues to an event.

More recent approaches rather propose joint embeddings that consider both the data graph and rules when computing embeddings. KALE [200] computes entity and relation embeddings using a translational model (specifically TransE) that is adapted to further consider rules using *t-norm fuzzy logics*. With reference to Figure 21, consider a simple rule $\boxed{?x}$ –bus→ $\boxed{?y}$ ⟹ $\boxed{?x}$ –connects to→ $\boxed{?y}$. We can use embeddings to assign plausibility scores to new edges, such as $e_1$: Piedras Rojas –bus→ Moon Valley. We can further apply the previous rule to generate a new edge $e_2$: Piedras Rojas –connects to→ Moon Valley from the predicted edge $e_1$. But what plausibility should we assign to this second edge? Letting $p_1$ and $p_2$ be the current plausibility scores of $e_1$ and $e_2$ (initialised using the standard embedding), then t-norm fuzzy logics suggests that the plausibility be updated as $p_1 p_2 - p_1 + 1$. Embeddings are then trained to jointly assign larger plausibility scores to positive examples versus negative examples of both edges and *ground rules*. An example of a positive ground rule based on Figure 21 would be Arica –bus→ San Pedro ⟹ Arica –connects to→ San Pedro. Negative ground rules randomly replace the relation in the head of the rule; for example, Arica –bus→ San Pedro ⟹ Arica –flight→ San Pedro. Guo et al. [201] later propose RUGE, which uses a joint model over ground rules (possibly soft rules with confidence scores) and plausibility scores to align both forms of scoring for unseen edges.

Generating ground rules can be costly. An alternative approach, called FSL [120], observes that in the case of a simple rule, such as $\boxed{?x}$ –bus→ $\boxed{?y}$ ⟹ $\boxed{?x}$ –connects to→ $\boxed{?y}$, the relation embedding bus should always return a lower plausibility than connects to. Thus, for all such rules, FSL proposes to train relation embeddings while avoiding violations of such inequalities. While relatively straightforward, FSL only supports simple rules, while KALE also supports more complex rules.

---

[25]Intuitively speaking, personalised PageRank starts at a given node and then determines the probability of a random walk being at a particular node after a given number of steps. A higher number of steps converges towards standard PageRank emphasising global node centrality, while a lower number emphasises proximity/relatedness to the starting node.

These works are interesting examples of how deductive and inductive forms of knowledge – in this case rules and embeddings – can interplay and complement each other.

## 5.3 Graph Neural Networks

While embeddings aim to provide a dense numerical representation of graphs suitable for use within existing machine learning models, another approach is to build custom machine learning models adapted for graph-structured data. Most custom learning models for graphs are based on (artificial) neural networks [527], exploiting a natural correspondence between both: a neural network already corresponds to a weighted, directed graph, where nodes serve as artificial neurons, and edges serve as weighted connections (axons). However, the typical topology of a traditional neural network – more specifically, a fully-connected feed-forward neural network – is quite homogeneous, being defined in terms of sequential layers of nodes where each node in one layer is connected to all nodes in the next layer. Conversely, the topology of a data graph is quite heterogeneous, being determined by the relations between entities that its edges represent.

A *graph neural network* (GNN) [437] builds a neural network based on the topology of the data graph; i.e., nodes are connected to their neighbours per the data graph. Typically a model is then learnt to map input features for nodes to output features in a supervised manner; output features for example nodes may be manually labelled, or may be taken from the knowledge graph. We now discuss the main ideas underlying two key flavours of GNN – *recursive GNNs* and *convolutional GNNs* – where we refer to Appendix B.6.3 for more formal definitions relating to GNNs.

*5.3.1 Recursive graph neural networks.* Recursive graph neural networks (RecGNNs) are the seminal approach to graph neural networks [437, 468]. The approach is conceptually similar to the systolic abstraction illustrated in Figure 22, where messages are passed between neighbours towards recursively computing some result. However, rather than define the functions used to decide the messages to pass, we rather label the output of a training set of nodes and let the framework learn the functions that generate the expected output, thereafter applying them to label other examples.

In a seminal paper, Scarselli et al. [437] proposed what they generically call a graph neural network (GNN), which takes as input a directed graph where nodes and edges are associated with *feature vectors* that can capture node and edge labels, weights, etc. These feature vectors remain fixed throughout the process. Each node in the graph is also associated with a *state vector*, which is recursively updated based on information from the node's neighbours – i.e., the feature and state vectors of the neighbouring nodes and the feature vectors of the edges extending to/from them – using a parametric function, called the *transition function*. A second parametric function, called the *output function*, is used to compute the final output for a node based on its own feature and state vector. These functions are applied recursively up to a fixpoint. Both parametric functions can be implemented using neural networks where, given a partial set of *supervised nodes* in the graph – i.e., nodes labelled with their desired output – parameters for the transition and output functions can be learnt that best approximate the supervised outputs. The result can thus be seen as a recursive neural network architecture.[26] To ensure convergence up to a fixpoint, certain restrictions are applied, namely that the transition function be a *contractor*, meaning that upon each application of the function, points in the numeric space are brought closer together (intuitively, in this case, the numeric space "shrinks" upon each application, ensuring a unique fixpoint).

To illustrate, consider, for example, that we wish to find priority locations for creating new tourist information offices. A good strategy would be to install them in hubs from which many tourists visit popular destinations. Along these lines, in Figure 26 we illustrate the GNN architecture proposed

---

[26]Some authors refer to such architectures as *recurrent graph neural networks*, observing that the internal state maintained for nodes can be viewed as a form of recurrence over a sequence of transitions.
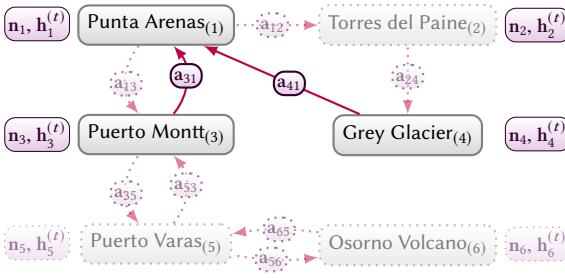
Fig. 26. On the left a sub-graph of Figure 21 highlighting the neighbourhood of Punta Arenas, where nodes are annotated with feature vectors ($\mathbf{n}_x$) and hidden states at step $t$ ($\mathbf{h}_x^{(t)}$), and edges are annotated with feature vectors ($\mathbf{a}_{xy}$); on the right, the GNN transition and output functions proposed by Scarselli et al. [437] and an example for Punta Arenas ($x = 1$), where $N(x)$ denotes the neighbouring nodes of $x$, $f_\mathbf{w}(\cdot)$ denotes the transition function with parameters $\mathbf{w}$ and $g_{\mathbf{w}'}(\cdot)$ denotes the output function with parameters $\mathbf{w}'$

by Scarselli et al. [437] for a sub-graph of Figure 21, where we highlight the neighbourhood of Punta Arenas. In this graph, nodes are annotated with feature vectors ($\mathbf{n}_x$) and hidden states at step $t$ ($\mathbf{h}_x^{(t)}$), while edges are annotated with feature vectors ($\mathbf{a}_{xy}$). Feature vectors for nodes may, for example, one-hot encode the type of node (*City*, *Attraction*, etc.), directly encode statistics such as the number of tourists visiting per year, etc. Feature vectors for nodes may, for example, one-hot encode the edge label (the type of transport), directly encode statistics such as the distance or number of tickets sold per year, etc. Hidden states can be randomly initialised. The right-hand side of Figure 26 provides the GNN transition and output functions, where $f_\mathbf{w}(\cdot)$ denotes the transition function with parameters $\mathbf{w}$ and $g_{\mathbf{w}'}(\cdot)$ denotes the output function with parameters $\mathbf{w}'$. An example is also provided for Punta Arenas ($x = 1$), where $N(x)$ denotes the neighbouring nodes of $x$. These functions will be recursively applied until a fixpoint is reached. To train the network, we can label examples of places that already have (or should have) tourist offices and places that do (or should) not have tourist offices. These labels may be taken from the knowledge graph, or may be added manually. The GNN can then learn parameters $\mathbf{w}$ and $\mathbf{w}'$ that give the expected output for the labelled examples, which can subsequently be used to label other nodes.

This GNN model is flexible and can be adapted in various ways [437]: we may define neighbouring nodes differently, for example to include nodes for outgoing edges, or nodes one or two hops away; we may allow pairs of nodes to be connected by multiple edges with different vectors; we may consider transition and output functions with distinct parameters for each node; we may add states and outputs for edges; we may change the sum to another aggregation function; etc.

*5.3.2 Convolutional graph neural networks.* Convolutional neural networks (CNNs) have gained a lot of attention, in particular, for machine learning tasks involving images [283]. The core idea in the image setting is to apply small kernels (aka filters) over localised regions of an image using a convolution operator to extract features from that local region. When applied to all local regions, the convolution outputs a feature map of the image. Typically multiple kernels are applied, forming multiple convolutional layers. These kernels can be learnt, given sufficient labelled examples.

One may note an analogy between GNNs as previously discussed, and CNNs as applied to images: in both cases, operators are applied over local regions of the input data. In the case of GNNs, the transition function is applied over a node and its neighbours in the graph. In the case of CNNs, the convolution is applied on a pixel and its neighbours in the image. Following this intuition, a number of *convolutional graph neural networks* (*ConvGNNs*) [68, 271, 527] have been proposed, where the

transition function is implemented by means of convolutions. A key consideration for ConvGNNs is how regions of a graph are defined. Unlike the pixels of an image, nodes in a graph may have varying numbers of neighbours. This creates a challenge: a benefit of CNNs is that the same kernel can be applied over all the regions of an image, but this requires more careful consideration in the case of ConvGNNs since neighbourhoods of different nodes can be diverse. Approaches to address these challenges involve working with spectral (e.g. [68, 271]) or spatial (e.g., [336]) representations of graphs that induce a more regular structure from the graph. An alternative is to use an attention mechanism [507] to *learn* the nodes whose features are most important to the current node.

Aside from architectural considerations, there are two main differences between RecGNNs and ConvGNNs. First, RecGNNs aggregate information from neighbours recursively up to a fixpoint, whereas ConvGNNs typically apply a fixed number of convolutional layers. Second, RecGNNs typically use the same function/parameters in uniform steps, while different convolutional layers of a ConvGNN can apply different kernels/weights at each distinct step.

## 5.4 Symbolic Learning

The supervised techniques discussed thus far – namely knowledge graph embeddings and graph neural networks – learn numerical models over graphs. However, such models are often difficult to explain or understand. For example, taking the graph of Figure 27, knowledge graph embeddings might predict the edge (SCL)–flight→(ARI) as being highly plausible, but they will not provide an interpretable model to help understand why this is the case: the reason for the result may lie in a matrix of parameters learnt to fit a plausibility score on training data. Such approaches also suffer from the *out-of-vocabulary* problem, where they are unable to provide results for edges involving previously unseen nodes or edges; for example, if we add an edge (SCL)–flight→(CDG), where (CDG) is new to the graph, a knowledge graph embedding will not have the entity embedding for (CDG) and would need to be retrained in order to estimate the plausibility of an edge (CDG)–flight→(SCL).

An alternative (sometimes complementary) approach is to adopt *symbolic learning* in order to learn *hypotheses* in a symbolic (logical) language that "explain" a given set of positive and negative edges. These edges are typically generated from the knowledge graph in an automatic manner (similar to the case of knowledge graph embeddings). The hypotheses then serve as interpretable models that can be used for further deductive reasoning. Given the graph of Figure 27, we may, for example, learn the rule '(x)–flight►(y)' ⇒ '(y)–flight►(x)' from observing that flight routes tend to be return routes. Alternatively, we might learn a DL axiom stating that airports are either domestic, international, or both: Airport ⊑ DomesticAirport ⊔ InternationalAirport. Such rules and axioms can then be used for deductive reasoning, and offer an interpretable model for new knowledge that is entailed/predicted; for example, from the aforementioned rule for return flights, one can interpret why a novel edge (SCL)–flight→(ARI) is predicted. This further offers domain experts the opportunity to verify the models – e.g., the rules and axioms – derived by such processes. Finally, rules/axioms are quantified (*all* flights have a return flight, *all* airports are domestic or international, etc.), so they can be applied to unseen examples (e.g., with the aforementioned rule, we can derive (CDG)–flight→(SCL) from a new edge (SCL)–flight→(CDG) with the unseen node (CDG)).

In this section, we discuss two forms of symbolic learning: *rule mining*, which learns rules from a knowledge graph, and *axiom mining*, which learns other forms of logical axioms. We refer to Appendix B.6.4 for a more formal treatment of these two tasks.

*5.4.1 Rule mining.* Rule mining, in the general sense, refers to discovering meaningful patterns in the form of rules from large collections of background knowledge. In the context of knowledge graphs, we assume a set of positive and negative edges as given. Typically positive edges are observed edges (i.e., those given or entailed by a knowledge graph) while negative edges are defined
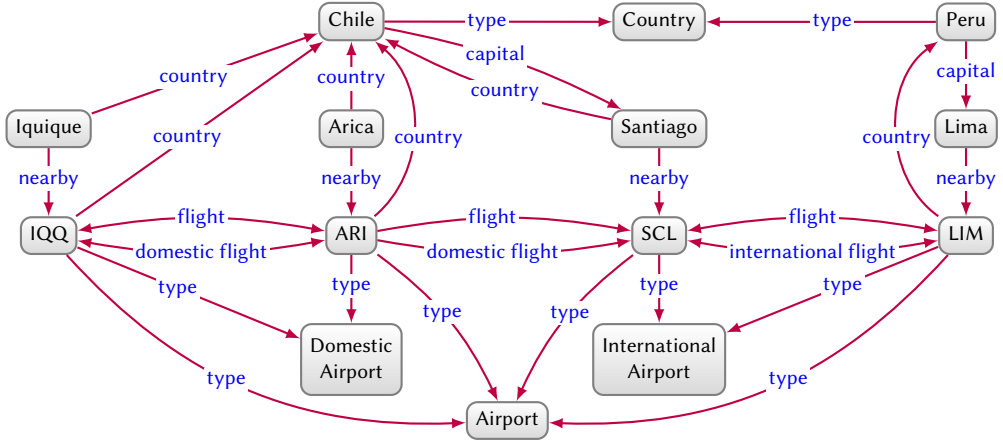
Fig. 27. An incomplete directed edge-labelled graph describing flights between airports

according to a given assumption of completeness (discussed later). The goal of rule mining is to identify new rules that entail a high ratio of positive edges from other positive edges, but entail a low ratio of negative edges from positive edges. The types of rules considered may vary from more simple cases, such as $?x$–flight►$?y$ ⟹ $?y$–flight►$?x$ mentioned previously, to more complex rules, such as $?x$–capital→$?y$–nearby→$?z$–type→(Airport) ⟹ $?z$–type►(International Airport), indicating that airports near capitals tend to be international airports; or $?x$—flight→$?y$—country→$?z$ ⟹ $?x$–domestic flight→$?y$, indicating that flights within the same country denote domestic flights (as seen in Section 4.3.1).

   Per the international airport example, rules are not assumed to hold in all cases, but rather are associated with measures of how well they conform to the positive and negative edges. In more detail, we call the edges entailed by a rule and the set of positive edges (not including the entailed edge itself), the *positive entailments* of that rule. The number of entailments that are positive is called the *support* for the rule, while the ratio of a rules' entailments that are positive is called the *confidence* for the rule [483]. As such, support and confidence indicate, respectively, the number and ratio of entailments "confirmed" to be true for the rule, where the goal is to identify rules that have both high support and high confidence. In fact, techniques for rule mining in relational settings have long been explored in the context of *Inductive Logic Programming* (*ILP*) [405]. However, knowledge graphs present novel challenges due to the scale of the data and the frequent assumption of incomplete data (OWA), where dedicated techniques have been proposed to address these issues [164].

   When dealing with an incomplete knowledge graph, it is not immediately clear how to define negative edges. A common heuristic – also used for knowledge graph embeddings – is to adopt a Partial Completeness Assumption (PCA) [164], which considers the set of positive edges to be those contained in the data graph, and the set of negative examples to be the set of all edges $x$-$p$►$y'$ not in the graph but where there exists a node $y$ such that $x$-$p$►$y$ is in the graph. Taking Figure 27, an example of a negative edge under PCA would be (SCL)—flight→(ARI) (given the presence of (SCL)—flight→(LIM)); conversely, (SCL)-domestic flight→(ARI) is neither positive nor negative. The PCA confidence measure is then the ratio of the support to all entailments in the positive or negative set [164]. For example, the support for the rule $?x$-domestic flight→$?y$ ⟹ $?y$-domestic flight→$?x$ is 2 (since it entails (IQQ)-domestic flight→(ARI) and (ARI)-domestic flight→(IQQ) in the graph), while the confidence is $\frac{2}{2} = 1$ (noting that (SCL)-domestic flight→(ARI), though entailed, is neither positive nor

negative, and is thus ignored by the measure). The support for the rule $\boxed{?x}$—flight→$\boxed{?y}$ $\Rightarrow$ $\boxed{?y}$—flight→$\boxed{?x}$ is analogously 4, while the confidence is $\frac{4}{5} = 0.8$ (noting that (SCL)—flight→(ARI) is negative).

The goal then, is to find rules satisfying given support and confidence thresholds. An influential rule-mining system for graphs is AMIE [163, 164], which adopts the PCA measure of confidence, and builds rules in a top-down fashion [483] starting with rule heads like $\Rightarrow$ $\boxed{?x}$—country→$\boxed{?y}$. For each rule head of this form (one for each edge label), three types of *refinements* are considered, each of which adds a new edge to the body of the rule. This new edge takes an edge label from the graph and may otherwise use *fresh variables* not appearing previously in the rule, *existing variables* that already appear in the rule, or nodes from the graph. The three refinements may then:

(1) add an edge with one existing variable and one fresh variable; for example, refining the aforementioned rule head might give: $\boxed{?z}$—flight→$\boxed{?x}$ $\Rightarrow$ $\boxed{?x}$—country→$\boxed{?y}$;

(2) add an edge with an existing variable and a node from the graph; for example, refining the above rule might give: (Domestic Airport)←type—$\boxed{?z}$—flight→$\boxed{?x}$ $\Rightarrow$ $\boxed{?x}$—country→$\boxed{?y}$;

(3) add an edge with two existing variables; for example, refining the above rule might give: (Domestic Airport)←type—$\boxed{?z}$—flight→$\boxed{?x}$, $\boxed{?y}$ country $\Rightarrow$ $\boxed{?x}$—country→$\boxed{?y}$.

These refinements can be combined arbitrarily, which gives rise to a potentially exponential search space, where rules meeting given thresholds for support and confidence are maintained. To improve efficiency, the search space can be pruned; for example, these three refinements always decrease support, so if a rule does not meet the support threshold, there is no need to explore its refinements. Further restrictions are imposed on the types of rules generated. First, only rules up to a certain fixed size are considered. Second, a rule must be *closed*, meaning that each variable appears in at least two edges of the rule, which ensures that rules are *safe*, meaning that each variable in the head appears in the body; for example, the rules produced previously by the first and second refinements are neither closed (variable $?y$ appears once) nor safe (variable $?y$ appears only in the head).[27] To ensure closed rules, the third refinement is applied until a rule is closed. For further discussion of possible optimisations based on pruning and indexing, we refer to the paper on AMIE+ [163].

Later works have built on these techniques for mining rules from knowledge graphs. Gad-Elrab et al. [162] propose a method to learn non-monotonic rules – rules with negated edges in the body – in order to capture exceptions to base rules; for example, the approach may learn a rule $\boxed{International\ Airport}$←¬ type··$\boxed{?z}$—flight→$\boxed{?x}$, $\boxed{?y}$ country $\Rightarrow$ $\boxed{?x}$—country→$\boxed{?y}$, indicating that flights are within the same country *except* when the (departure) airport is international, where the exception is shown dotted and we use ¬ to negate an edge. The RuLES system [230] – which is also capable of learning non-monotonic rules – proposes to mitigate the limitations of the PCA heuristic by extending the confidence measure to consider the plausibility scores of knowledge graph embeddings for entailed edges not appearing in the graph. Where available, explicit statements about the completeness of the knowledge graph (such as expressed in shapes; see Section 3.1.2) can be used in lieu of PCA for identifying negative edges. Along these lines, CARL [381] exploits additional knowledge about the cardinalities of relations to refine the set of negative examples and the confidence measure for candidate rules. Alternatively, where available, ontologies can be used to derive logically-certain negative edges under OWA through, for example, disjointness axioms. The system proposed by d'Amato et al. [109, 110] leverages ontologically-entailed negative edges for determining the confidence of rules generated through an evolutionary algorithm.

While the previous works involve discrete expansions of candidate rules for which a fixed confidence scoring function is applied, another line of research is on a technique called *differentiable*

---

[27]Safe rules like $\boxed{?x}$—capital→$\boxed{?y}$—nearby→$\boxed{?z}$—type→(Airport) $\Rightarrow$ $\boxed{?z}$—type→(International Airport) are not closed as $?x$ appears only in one edge. Hence the condition that rules are closed is strictly stronger than the condition that they are safe.

*rule mining* [419, 430, 534], which allows end-to-end learning of rules. The core idea is that the joins in rule bodies can be represented as matrix multiplication. More specifically, we can represent the relations of an edge label $p$ by the adjacency matrix $\mathbf{A}_p$ (of size $|V| \times |V|$) such that the value on the $i^{\text{th}}$ row of the $j^{\text{th}}$ column is 1 if there is an edge labelled $p$ from the $i^{\text{th}}$ entity to the $j^{\text{th}}$ entity; otherwise the value is 0. Now we can represent a join in a rule body as matrix multiplication; for example, given $?x$—domestic flight→$?y$—country→$?z$ $\Rightarrow$ $?x$—country→$?z$, we can denote the body by the matrix multiplication $\mathbf{A}_{\text{df.}}\mathbf{A}_{\text{c.}}$, which gives an adjacency matrix representing entailed country edges, where we should expect the 1's in $\mathbf{A}_{\text{df.}}\mathbf{A}_{\text{c.}}$ to be covered by the head's adjacency matrix $\mathbf{A}_{\text{c.}}$. Since we are given adjacency matrices for all edge labels, we are left to learn confidence scores for individual rules, and to learn rules (of varying length) with a threshold confidence. Along these lines, NeuralLP [534] uses an *attention mechanism* to select a variable-length sequence of edge labels for path-like rules of the form $?x$—$p_1$→$?y_1$—$p_2$→ $\cdots$ —$p_n$→$?y_n$—$p_{n+1}$→$?z$ $\Rightarrow$ $?x$—$p$→$?z$, for which confidences are likewise learnt. DRUM [430] also learns path-like rules, where, observing that some edge labels are more/less likely to follow others in the rules – for example, flight will not be followed by capital in the graph of Figure 21 as the join will be empty – the system uses bidirectional recurrent neural networks (a popular technique for learning over sequential data) to learn sequences of relations for rules, and their confidences. These differentiable rule mining techniques are, however, currently limited to learning path-like rules.

*5.4.2 Axiom mining.* Aside from rules, more general forms of axioms – expressed in logical languages such as DLs (see Section 4.3.2) – can be mined from a knowledge graph. We can divide these approaches into two categories: those mining specific axioms and more general axioms.

Among systems mining specific types of axioms, disjointness axioms are a popular target; for example, the disjointness axiom `DomesticAirport ⊓ InternationalAirport ≡ ⊥` states that the intersection of the two classes is equivalent to the empty class, or in simpler terms, no node can be simultaneously of type Domestic Airport and International Airport. The system proposed by [512] extracts disjointness axioms based on (negative) *association rule mining* [4], which finds pairs of classes where each has many instances in the knowledge graph but there are relatively few (or no) instances of both classes. Töpper et al. [497] rather extract disjointness for pairs of classes that have a cosine similarity below a fixed threshold. For computing this cosine similarity, class vectors are computed using a TF–IDF analogy, where the "document" of each class is constructed from all of its instances, and the "terms" of this document are the properties used on the class' instances (preserving multiplicities). While the previous two approaches find disjointness constraints between named classes (e.g., *city* is disjoint with *airport*), Rizzo et al. [418] propose an approach that can capture disjointness constraints between class descriptions (e.g., *city without an airport nearby* is disjoint with *city that is the capital of a country*). The approach first clusters similar nodes of the knowledge base. Next, a *terminological cluster tree* is extracted, where each leaf node indicates a cluster extracted previously, and each internal (non-leaf) node is a class definition (e.g., *cities*) where the left child is either a cluster having all nodes in that class or a sub-class description (e.g., *cities without airports*) and the right child is either a cluster having no nodes in that class or a disjoint-class description (e.g., *non-cities with events*). Finally, candidate disjointness axioms are proposed for pairs of class descriptions in the tree that are not entailed to have a subclass relation.

Other systems propose methods to learn more general axioms. A prominent such system is DL-Learner [70], which is based on algorithms for *class learning* (aka *concept learning*), whereby given a set of positive nodes and negative nodes, the goal is to find a logical class description that divides the positive and negative sets. For example, given {Iquique, Arica} as the positive set and {Santiago} as the negative set, we may learn a (DL) class description $\exists$nearby.Airport $\sqcap \neg(\exists$capital$^-.\top)$, denoting entities near to an airport that are not capitals, of which all positive nodes are instances

and no negative nodes are instances. Such class descriptions are learnt in an analogous manner to how aforementioned systems like AMIE learn rules, with a refinement operator used to move from more general classes to more specific classes (and vice-versa), a confidence scoring function, and a search strategy. The system further supports learning more general axioms through a scoring function that uses count queries to determine what ratio of expected edges – edges that would be entailed were the axiom true – are indeed found in the graph; for example, to score the axiom $\exists\texttt{flight}^-.\texttt{DomesticAirport} \sqsubseteq \texttt{InternationalAirport}$ over Figure 27, we can use a graph query to count how many nodes have incoming flights from a domestic airport (there are 3), and how many nodes have incoming flights from a domestic airport *and* are international airports (there is 1), where the greater the difference between both counts, the weaker the evidence for the axiom.

## 6 CREATION AND ENRICHMENT

In this section, we discuss the principal techniques by which knowledge graphs can be created and subsequently enriched from diverse sources of legacy data that may range from plain text to structured formats (and anything in between). The appropriate methodology to follow when creating a knowledge graph depends on the actors involved, the domain, the envisaged applications, the available data sources, etc. Generally speaking, however, the flexibility of knowledge graphs lends itself to starting with an initial core that can be incrementally enriched from other sources as required (typically following an Agile [243] or "pay-as-you-go" [448] methodology). For our running example, we assume that the tourism board decides to build a knowledge graph from scratch, aiming to initially describe the main tourist attractions – places, events, etc. – in Chile in order to help visiting tourists identify those that most interest them. The board decides to postpone adding further data, like transport routes, reports of crime, etc., for a later date.

### 6.1 Human Collaboration

One approach for creating and enriching knowledge graphs is to solicit direct contributions from human editors. Such editors may be found in-house (e.g., employees of the tourist board), using crowd-sourcing platforms, through feedback mechanisms (e.g., tourists adding comments on attractions), through collaborative-editing platforms (e.g., an attractions wiki open to public edits), etc. Though human involvement incurs high costs [376], some prominent knowledge graphs have been primarily based on direct contributions from human editors [214, 515]. Depending on how the contributions are solicited, however, the approach has a number of key drawbacks, due primarily to human error [382], disagreement [535], bias [255], vandalism [218], etc. Successful collaborative creation further raises challenges concerning licensing, tooling, and culture [382]. Humans are sometimes rather employed to verify and curate additions to a knowledge graph extracted by other means [382] (through, e.g., video games with a purpose [259]), to define high-quality mappings from other sources [113], to define appropriate high-level schema [267, 287], and so forth.

### 6.2 Text Sources

Text corpora – such as sourced from newspapers, books, scientific articles, social media, emails, web crawls, etc. – are an abundant source of rich information [220, 422]. However, extracting such information with high precision and recall for the purposes of creating or enriching a knowledge graph is a non-trivial challenge. To address this, techniques from Natural Language Processing (NLP) [258, 322] and Information Extraction (IE) [192, 317, 522] can be applied. Though processes vary considerably across text extraction frameworks, in Figure 28 we illustrate four core tasks for text extraction on a sample sentence. We will discuss these tasks in turn.
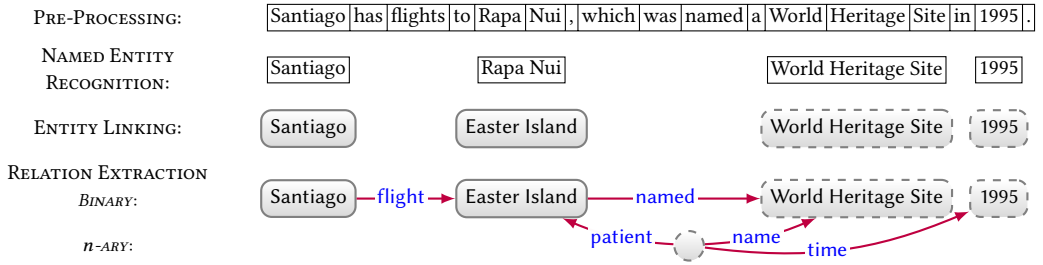
Fig. 28. Text extraction example; nodes new to the knowledge graph are shown dashed

*6.2.1 Pre-processing.* The pre-processing task may involve applying various techniques to the input text, where Figure 28 illustrates *Tokenisation*, which parses the text into atomic terms and symbols. Other pre-processing tasks applied to a text corpus may include: *Part-of-Speech* (*POS*) *tagging* [258, 322] to identify terms representing verbs, nouns, adjectives, etc.; *Dependency Parsing*, which extracts a grammatical tree structure for a sentence where leaf nodes indicate individual words that together form phrases (e.g., noun phrases, verb phrases) and eventually clauses and sentences [258, 322]; and *Word Sense Disambiguation* (*WSD*) [350] to identify the meaning (aka *sense*) in which a word is used, linking words with a lexicon of senses (e.g., WordNet [331] or BabelNet [351]), where, for instance, the term flights may be linked with the WordNet sense "an instance of travelling by air" rather than "a stairway between one floor and the next'. The appropriate type of pre-processing to apply often depends on the requirements of later tasks in the pipeline.

*6.2.2 Named Entity Recognition (NER).* The NER task identifies mentions of named entities in a text [347, 409], typically targeting mentions of people, organisations, locations, and potentially other types [300, 348, 537]. A variety of NER techniques exist, with many modern approaches based on learning frameworks that leverage lexical features (e.g., POS tags, dependency parse trees, etc.) and gazetteers (e.g., lists of common first names, last names, countries, prominent businesses, etc.). Supervised methods [42, 154, 288] require manually labelling all entity mentions in a training corpus, whereas *bootstrapping*-based approaches [94, 143, 202, 348] rather require a small set of *seed examples* of entity mentions from which patterns can be learnt and applied to unlabelled text. *Distant supervision* [300, 412, 537] uses known entities in a knowledge graph as seed examples through which similar entities can be detected. Aside from learning-based frameworks, manually-crafted rules [84, 275] are still sometimes used due to their more controllable and predictable behaviour [85]. The named entities identified by NER may be used to generate new candidate nodes for the knowledge graph (known as *emerging entities*, shown dashed in Figure 28), or may be linked to existing nodes per the Entity Linking task described in the following.

*6.2.3 Entity Linking (EL).* The EL task associates mentions of entities in a text with the existing nodes of a target knowledge graph, which may be the nucleus of a knowledge graph under creation, or an external knowledge graph [526]. In Figure 28, we assume that the nodes (Santiago) and (Easter Island) already exist in the knowledge graph (possibly extracted from other sources). EL may then link the given mentions to these nodes. The EL task presents two main challenges. First, there may be multiple ways to mention the same entity, as in the case of Rapa Nui and Easter Island ; if we created a node (Rapa Nui) to represent that mention, we would split the information available under both mentions across different nodes, where it is thus important for the target knowledge graph to capture the various aliases and multilingual labels by which one can refer to an entity [340]. Secondly, the same mention in different contexts can refer to distinct entities; for instance, Santiago can refer

to cities in Chile, Cuba, Spain, among others. The EL task thus considers a *disambiguation phase* wherein mentions are associated to candidate nodes in the knowledge graph, the candidates are ranked, and the most likely node being mentioned is chosen [526]. Context can be used in this phase; for example, if (Easter Island) is a likely candidate for the corresponding mention alongside [Santiago], we may boost the probability that this mention refers to the Chilean capital as both candidates are located in Chile. Other heuristics for disambiguation consider a prior probability, where for example, [Santiago] most often refers to the Chilean capital (being, e.g., the largest city with that name); centrality measures on the knowledge graph can be used for such purposes [526].

*6.2.4 Relation Extraction (RE).* The RE task extracts relations between entities in the text [22, 545]. The simplest case is that of extracting binary relations in a *closed setting* wherein a fixed set of relation types are considered. While traditional approaches often used manually-crafted patterns [215], modern approaches rather tend to use learning-based frameworks [421], including supervised methods over manually-labelled examples [74, 545]. Other learning-based approaches again use bootstrapping [75, 143] and distant supervision [233, 334, 415, 462, 485, 532] to forgo the need for manual labelling; the former requires a subset of manually-labelled seed examples, while the latter finds sentences in a large corpus of text mentioning pairs of entities with a known relation/edge, which are used to learn patterns for that relation. Binary RE can also be applied using unsupervised methods in an open setting – often referred to as *Open Information Extraction* (*OIE*) [28, 144, 145, 320, 321, 335] – whereby the set of target relations is not pre-defined but rather extracted from text based on, for example, dependency parse trees from which relations are taken.

A variety of RE methods have been proposed to extract $n$-ary relations that capture further context for how entities are related. In Figure 28, we see how an $n$-ary relation captures additional temporal context, denoting when Rapa Nui was named a World Heritage site; in this case, an anonymous node is created to represent the higher-arity relation in the directed-labelled graph. Various methods for $n$-ary RE are based on *frame semantics* [153], which, for a given verb (e.g., "*named*"), captures the entities involved and how they may be interrelated. Resources such as FrameNet [24] then define frames for words, such as to identify that the semantic frame for "*named*" includes a *speaker* (the person naming something), an *entity* (the thing named) and a *name*. Optional frame elements are an *explanation*, a *purpose*, a *place*, a *time*, etc., that may add context to the relation. Other RE methods are rather based on *Discourse Representation Theory* (*DRT*) [262], which considers a logical representation of text based on existential events. Under this theory, for example, the naming of Easter Island as a World Heritage Site is considered to be an (existential) event where Easter Island is the *patient* (the entity affected), leading to the logical (neo-Davidsonian) formula:

$$\exists e : \big(\mathrm{naming}(e), \mathrm{patient}(e, \boxed{\text{Easter Island}}), \mathrm{name}(e, \boxed{\text{World Heritage Site}})\big)$$

Such a formula is analogous to the idea of reification, as discussed previously in Section 3.3.

Finally, while relations extracted in a closed setting are typically mapped directly to a knowledge graph, relations that are extracted in an open setting may need to be aligned with the knowledge graph; for example, if an OIE process extracts a binary relation (Santiago)-has flights to→(Easter Island), it may be the case that the knowledge graph does not have other edges labelled has flights to, where alignment may rather map such a relation to the edge (Santiago)-flight→(Easter Island) assuming flight is used in the knowledge graph. A variety of methods have been applied for such purposes, including mappings [98, 169] and rules [423] for aligning $n$-ary relations; distributional and dependency-based similarities [339], association rule mining [130], Markov clustering [131] and linguistic techniques [318] for aligning OIE relations; amongst others.

*6.2.5 Joint tasks.* Having presented the four main tasks for building knowledge graphs from text, it is important to note that frameworks do not always follow this particular sequence of tasks.

```
<html>
 <head><title>UNESCO World Heritage Sites</title></head>
 <body>
  <h1>World Heritage Sites</h1>
  <h2>Chile</h2>
  <p>Chile has 6 UNESCO World Heritage Sites.</p>
  <table border="1">
   <tr><th>Place</th><th>Year</th><th>Criteria</th></tr>
   <tr><td>Rapa Nui</td><td>1995</td>
       <td rowspan="6">Cultural</td></tr>
   <tr><td>Churches of Chiloé</td><td>2000</td></tr>
   <tr><td>Historical Valparaíso</td><td>2003</td></tr>
   <tr><td>Saltpeter Works</td><td>2005</td></tr>
   <tr><td>Sewell Mining Town</td><td>2006</td></tr>
   <tr><td>Qhapaq Ñan</td><td>2014</td></tr>
  </table>
 </body>
</html>
```

⚲ UNESCO World Heritage Sites ✕

**World Heritage Sites**
**Chile**

Chile has 6 UNESCO World Heritage Sites.

| Place | Year | Criteria |
|---|---|---|
| Rapa Nui | 1995 | Cultural |
| Churches of Chiloé | 2000 | |
| Historical Valparaíso | 2003 | |
| Saltpeter Works | 2005 | |
| Sewell Mining Town | 2006 | |
| Qhapaq Ñan | 2014 | |

Fig. 29. Example markup document (HTML) with source-code (left) and formatted document (right)

A common trend, for example, is to combine interdependent tasks, jointly performing WSD and EL [340], or NER and EL [310, 360], or NER and RE [413, 542], etc., in order to mutually improve the performance of multiple tasks. For further details on extracting knowledge from text we refer to the book by Maynard et al. [322] and the recent survey by Martínez-Rodríguez et al. [317].

## 6.3 Markup Sources

The Web was founded on interlinking *markup documents* wherein markers (aka *tags*) are used to separate elements of the document (typically for formatting purposes). Most documents on the Web use the HyperText Markup Language (HTML). Figure 29 presents an example HTML webpage about World Heritage Sites in Chile. Other formats of markup include Wikitext used by Wikipedia, TeX for typesetting, Markdown used by Content Management Systems, etc. One approach for extracting information from markup documents – in order to create and/or enrich a knowledge graph – is to strip the markers (e.g., HTML tags), leaving only plain text upon which the techniques from the previous section can be applied. However, markup can be useful for extraction purposes, where variations of the aforementioned tasks for text extraction have been adapted to exploit such markup [304, 307, 317]. We can divide extraction techniques for markup documents into three main categories: general approaches that work independently of the markup used in a particular format, often based on *wrappers* that map elements of the document to the output; focussed approaches that target specific forms of markup in a document, most typically *web tables* (but sometimes also lists, links, etc.); and form-based approaches that extract the data underlying a webpage, per the notion of the *Deep Web*. These approaches can often benefit from the regularities shared by webpages of a given website, be it due to informal conventions on how information is published across webpages, or due to the re-use of templates to automatically generate content across webpages; for example, intuitively speaking, while the webpage of Figure 29 is about Chile, we will likely find pages for other countries following the same structure on the same website.

*6.3.1 Wrapper-based extraction.* Many general approaches are based on *wrappers* that locate and extract the useful information directly from the markup document. While the traditional approach was to define such wrappers manually – a task for which a variety of declarative languages and tools have been defined – such approaches are brittle to changes in a websites layout [152]. Hence other approaches allow for (semi-)automatically *inducing* wrappers [155]. A modern such approach – used to enrich knowledge graphs in systems such as LODIE [174] – is to apply distant supervision,

**Report**

| crime | claimant | station | date |
|---|---|---|---|
| Pickpocketing | XY12SDA | Viña del Mar | 2019-04-12 |
| Assault | AB9123N | Arica | 2019-04-12 |
| Pickpocketing | XY12SDA | Rapa Nui | 2019-04-12 |
| Fraud | FI92HAS | Arica | 2019-04-13 |

**Claimant**

| id | name | country |
|---|---|---|
| XY12SDA | John Smith | U.S. |
| AB9123N | Joan Dubois | France |
| XI92HAS | Jorge Hernández | Chile |

Fig. 30. Example relational database instance with two tables describing crime data

whereby EL is used to identify and link entities in the webpage to nodes in the knowledge graph such that paths in the markup that connect pairs of nodes for known edges can be extracted, ranked, and applied to other examples. Taking Figure 29, for example, distant supervision may link ⌞Rapa Nui⌝ and ⌞World Heritage Sites⌝ to the nodes (Easter Island) and (World Heritage Site) in the knowledge graph using EL, and given the edge (Easter Island)–named→(World Heritage Site) in the knowledge graph (extracted per Figure 28), identify the candidate path $(x, \texttt{td}[1]^- \cdot \texttt{tr}^- \cdot \texttt{table}^- \cdot \texttt{h1}, y)$ as reflecting edges of the form (x)–named→(y), where $t[n]$ indicates the $n^{\text{th}}$ child of tag $t$, $t^-$ its inverse, and $t_1 \cdot t_2$ concatenation. Finally, paths with high confidence (e.g., ones "witnessed" by many known edges in the knowledge graph) can then be used to extract novel edges, such as (Qhapaq Ñan)–named→(World Heritage Site), both on this page and on related pages of the website with similar structure (e.g., for other countries).

*6.3.2 Web table extraction.* Other approaches target specific types of markup, most commonly *web tables*, i.e., tables embedded in HTML webpages. However, web tables are designed to enhance human readability, which often conflicts with machine readability. Many web tables are used for layout and page structure (e.g., navigation bars), while those that do contain data may follow different formats such as relational tables, listings, attribute-value tables, matrices, etc. [78, 104]. Hence a first step is to classify tables to find ones appropriate for the given extraction mechanism(s) [104, 134]. Next, web tables may contain column spans, row spans, inner tables, or may be split vertically to improve human aesthetics. Hence a table normalisation phase is required to identify headers, merge split tables, un-nest tables, transpose tables, etc. [78, 104, 121, 140, 292, 393]. Subsequently, approaches may need to identify the *protagonist* [104, 345] – the main entity that the table describes – which is rather found elsewhere in the webpages; for example, though ⌞World Heritage Sites⌝ is the protagonist of the table of Figure 28, it is not mentioned by the table. Finally, extraction processes can be applied, potentially associating cells with entities [297, 343], columns with types [121, 297, 343], and column pairs with relations [297, 345]. For the purposes of enriching knowledge graphs, more recent approaches again apply distant supervision, first linking table cells to knowledge graph nodes, which are used to generate candidates for type and relation extraction [297, 343, 345]. Statistical distributions can also aid in linking numerical columns [354]. Specialised extraction frameworks have also been designed for tables on specific websites, where prominent knowledge graphs, such as DBpedia [291] and YAGO [482] focus on extraction from info-box tables in Wikipedia.

*6.3.3 Deep Web crawling.* The *Deep Web* presents a rich source of information accessible only through searches on web forms, thus requiring *Deep Web crawling* techniques to access [312]. Systems have been proposed to extract knowledge graphs from Deep Web sources [93, 172, 290]. Approaches typically attempt to generate sensible form inputs – which may be based on a user query or generated from reference knowledge – and then extract data from the generated responses (markup documents) using the aforementioned techniques [93, 172, 290].
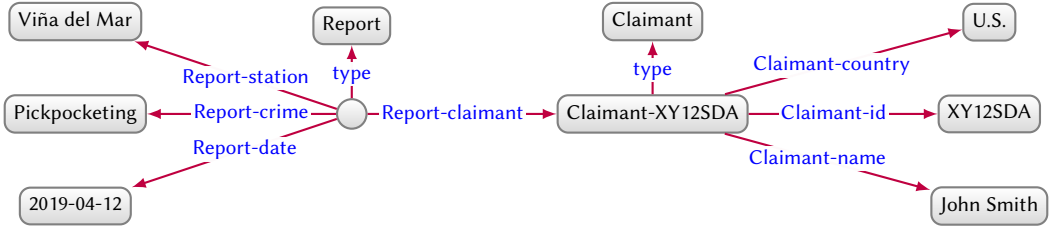
Fig. 31. Possible result of applying a direct mapping to the first rows of both tables in Figure 30

## 6.4 Structured Sources

Much of the legacy data available within organisations and on the Web is represented in structured formats, primarily tables – in the form of relational databases, CSV files, etc. – but also tree-structured formats such as JSON, XML etc. Unlike text and markup documents, structured sources can often be *mapped* to knowledge graphs whereby the structure is (precisely) transformed according to a mapping rather than (imprecisely) extracted. The mapping process involves two steps: 1) create a mapping from the source to a graph, and 2) use the mapping in order to materialise the source data as a graph or to virtualise the source (creating a graph view over the legacy data).

*6.4.1 Mapping from tables.* Tabular sources of data are prevalent, where, for example, the structured content underlying many organisations, websites, etc., are housed in relational databases. In Figure 30 we present an example of a relational database instance that we would like to integrate into our knowledge graph under construction. There are then two approaches for mapping content from tables to knowledge graphs: a *direct mapping*, and a *custom mapping*.

A direct mapping automatically generates a graph from a table. We present in Figure 31 the result of a standard direct mapping [18], which creates an edge (x)–y→(z) for each (non-header, non-empty, non-NULL) cell of the table, such that (x) represents the row of the cell, y the column name of the cell, and (z) the value of the cell. In particular, (x) typically encodes the values of the primary key for a row (e.g., **Claimant.id**); otherwise, if no primary key is defined (e.g., per the **Report** table), (x) can be an anonymous node or a node based on the row number. The node (x) and edge label y further encode the name of the table to avoid clashes across tables that have the same column names used with different meanings. For each row (x), we may add a type edge based on the name of its table. The value (z) may be mapped to datatype values in the corresponding graph model based on the source domain (e.g., a value in an SQL column of type Date can be mapped to xsd:date in the RDF data model). If the value is NULL (or empty), typically the corresponding edge will be omitted.[28] With respect to Figure 31, we highlight the difference between the nodes (Claimant-XY12SDA) and (XY12SDA), where the former denotes the row (or entity) identified by the latter primary key value. In case of a foreign key between two tables – such as **Report.claimant** referencing **Claimant.id** – we can link, for example, to (Claimant-XY12SDA) rather than (XY12SDA), where the former node also has the name and country of the claimant. A direct mapping along these lines has been standardised for mapping relational databases to RDF [18], where Stoica et al. [475] have recently proposed an analogous direct mapping for property graphs. Another direct mapping has been defined for CSV and other tabular data [489] that further allows for specifying column names, primary/foreign keys, and data types – which are often missing in such data formats – as part of the mapping itself.

---

[28]One might consider representing NULLs with anonymous nodes. However, NULLs in SQL can be used to mean that there is no such value, which conflicts with the existential semantics of anonymous nodes in models such as RDF (i.e., blank nodes).

Although a direct mapping can be applied automatically on tabular sources of data and preserve the information of the original source – i.e., allowing a deterministic inverse mapping that reconstructs the tabular source from the output graph [446] – in many cases it is desirable to customise a mapping, such as to align edge labels or nodes with a knowledge graph under enrichment, etc. Along these lines, declarative mapping languages allow for manually defining custom mappings from tabular sources to graphs. A standard language along these lines is the RDB2RDF Mapping Language (R2RML) [113], which allows for mapping from individual rows of a table to one or more custom edges, with nodes and edges defined either as constants, as individual cell values, or using templates that concatenate multiple cell values from a row and static substrings into a single term; for example, a template {id}-{country} may produce nodes such as (XY12SDA-U.S.) from the **Claimant** table. In case that the desired output edges cannot be defined from a single row, R2RML allows for (SQL) queries to generate tables from which edges can be extracted where, for example, edges such as (U.S.)-crimes→(2) can be generated by defining the mapping with respect to a query that joins the **Report** and **Claimant** tables on **claimant**=**id**, grouping by **country**, and applying a count. A mapping can then be defined on the results table such that the source node denotes the value of **country**, the edge label is the constant crimes, and the target node is the count value. An analogous standard also exists for mapping CSV and other tabular data to RDF graphs, again allowing keys, column names, and datatypes to be chosen as part of the mapping [490].

Once the mappings have been defined, one option is to use them to *materialise* graph data following an *Extract-Transform-Load* (*ETL*) approach, whereby the tabular data are transformed and explicitly serialised as graph data using the mapping. A second option is to use *virtualisation* through a *Query Rewriting* (*QR*) approach, whereby queries on the graph (using, e.g., SPARQL, Cypher, etc.) are translated to queries over the tabular data (typically using SQL). Comparing these two options, ETL allows the graph data to be used as if they were any other data in the knowledge graph. However, ETL requires updates to the underlying tabular data to be explicitly propagated to the knowledge graph, whereas a QR approach only maintains one copy of data to be updated. The area of *Ontology-Based Data Access* (*OBDA*) [528] is then concerned with QR approaches that support ontological entailments as discussed in Section 4. Although most QR approaches only support non-recursive entailments expressible as a single (non-recursive) query, some QR approaches support recursive entailments through rewritings to recursive queries [447].

*6.4.2 Mapping from trees.* A number of popular data formats are based on trees, including XML and JSON. While one could imagine – leaving aside issues such as the ordering of children in a tree – a trivial direct mapping from trees to graphs by simply creating edges of the form (x)-child→(y) for each node $y$ that is a child of $x$ in the source tree, such an approach is not typically used, as it represents the literal structure of the source data. Instead, the content of tree-structured data can be more naturally represented as a graph using a custom mapping. Along these lines, the GRDLL standard [95] allows for mapping from XML to (RDF) graphs, while the JSON-LD standard [469] allows for mapping from JSON to (RDF) graphs. In contrast, hybrid query languages such as XSPARQL [43] allow for querying XML and RDF in an integrated fashion, thus supporting both materialisation and virtualisation of graphs over tree-structured sources of legacy data.

*6.4.3 Mapping from other knowledge graphs.* Another route to construct or enrich knowledge graphs is to leverage existing knowledge graphs as a source. In our scenario, for instance, a large number of points of interest for the Chilean tourist board may be available in existing knowledge graphs such as DBpedia [291], LinkedGeoData [472], Wikidata [515], YAGO [232], BabelNet [351], etc. However, depending on the knowledge graph under construction, not all entities and/or relations may be of interest. A standard option to extract a relevant sub-graph of data is to use SPARQL construct-queries that generate graphs as output [353]. Entity and schema alignment
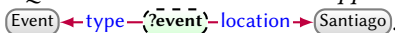
between the knowledge graphs may be further necessary to better integrate (parts of) external knowledge graphs; this may be done using linking tools for graphs [356, 513], based on the use of external identifiers [382], or indeed may be done manually [382]. For instance, Wikidata [515] uses Freebase [50, 382] as a source; Gottschalk and Demidova [190] extract an event-centric knowledge graph from Wikidata, DBpedia and YAGO; while Neumaier and Polleres [353] construct a spatio-temporal knowledge graph from Geonames, Wikidata, and PeriodO [186] (as well as tabular data).

## 6.5  Schema/Ontology Creation

The discussion thus far has focussed on extracting *data* from external sources in order to create and enrich a knowledge graph. In this section, we discuss some of the principal methods for generating a schema based on external sources of data, including human knowledge. For discussion on extracting a schema from the knowledge graph itself, we refer back to Section 3.1.3. In general, much of the work in this area has focussed on the creation of ontologies using either ontology engineering methodologies, and/or ontology learning. We discuss these two approaches in turn.

*6.5.1  Ontology engineering.* Ontology engineering refers to the development and application of methodologies for building ontologies, proposing principled processes by which better quality ontologies can be constructed and maintained with less effort. Early methodologies [151, 195, 366] were often based on a waterfall-like process, where requirements and conceptualisation were fixed before starting to implement the ontology in a logical language, using, for example, an ontology engineering tool [187, 267, 269]. However, for situations involving large or ever-evolving ontologies, more iterative and agile ways of building and maintaining ontologies have been proposed.

DILIGENT [389] was an early example of an agile methodology, proposing a complete process for ontology life-cycle management and knowledge evolution, as well as separating local changes (local views on knowledge) from global updates of the core part of the ontology, using a review process to authorise the propagation of changes from the local to the global level. This methodology is similar to how, for instance, the large clinical reference terminology SNOMED CT [249] (also available as an ontology) is maintained and evolved, where the (international) core terminology is maintained based on global requirements, while national or local extensions to SNOMED CT are maintained based on local requirements. A group of authors then decides which national or local extensions to propagate to the core terminology. More modern agile methodologies include eXtreme Design (XD) [46, 395], Modular Ontology Modelling (MOM) [227, 282], Simplified Agile Methodology for Ontology Development (SAMOD) [384], etc. Such methodologies typically include two key elements: *ontology requirements* and (more recently) *ontology design patterns*.

Ontology requirements specify the intended task of the resulting ontology – or indeed the knowledge graph itself – based on the ontology as its schema. A common way to express ontology requirements is through Competency Questions (CQ) [196], which are natural language questions illustrating the typical knowledge that one would require the ontology (or the knowledge graph) to provide. Such CQs can then be complemented with additional restrictions, and reasoning requirements, in case that the ontology should also contain restrictions and general axioms for inferring new knowledge or checking data consistency. A common way of testing ontologies (or knowledge graphs based on them) is then to formalise the CQs as queries over some test set of data, and make sure the expected results are entailed [49, 268]. We may, for example, consider the CQ "*What are all the events happening in Santiago?*", which can be represented as a graph query Event ←type– ?event –location→ Santiago . Taking the data graph of Figure 1 and the axioms of Figure 11, we can check to see if the expected result EID15 is entailed by the ontology and the data, and since it is not, we may consider expanding the axioms to assert that location –type→ Transitive .

Ontology Design Patterns (ODPs) are another common feature of modern methodologies [48, 167], specifying generalisable ontology modelling patterns that can be used as inspiration for modelling similar patterns, as modelling templates [135, 460], or as directly reusable components [157, 454]. Several pattern libraries have been made available online, ranging from carefully curated ones [17, 454] to open and community moderated ones [157]. As an example, in modelling an ontology for our scenario, we may decide to follow the Core Event ontology pattern proposed by Krisnadhi and Hitzler [281], which specifies a spatio-temporal extent, sub-events, and participants of an event, further suggesting competency questions, formal definitions, etc., to support this pattern.

*6.5.2 Ontology learning.* The previous methodologies outline methods by which ontologies can be built and maintained manually. Ontology learning, in contrast, can be used to (semi-)automatically extract information from text that is useful for the ontology engineering process [73, 89]. Early methods focussed on extracting terminology from text that may represent the relevant domain's classes; for example, from a collection of text documents about tourism, a terminology extraction tool – using measures of *unithood* that determine how cohesive an *n*-gram is as a unitary phrase, and *termhood* that determine how relevant the phrase is to a domain [318] – may identify *n*-grams such as "visitor visa", "World Heritage Site", "off-peak rate", etc., as terminology of particular importance to the tourist domain, and that thus may merit inclusion in such an ontology. Axioms may also be extracted from text, where subclass axioms are commonly targeted, based on modifying nouns and adjectives that incrementally specialise concepts (e.g., extracting Visitor Visa -subc. of→ Visa from the noun phrase "visitor visa" and isolated appearances of "visa" elsewhere), or using Hearst patterns [215] (e.g., extracting Off-Peak Rate -subc. of→ Discount from "many discounts, such as off-peak rates, are available" based on the pattern "X, such as Y"). Textual definitions can also be harvested from large texts to extract hypernym relations and induce a taxonomy from scratch [506]. More recent works aim to extract more expressive axioms from text, including disjointness axioms [512]; and axioms involving the union and intersection of classes, along with existential, universal, and qualified-cardinality restrictions [387]. The results of an ontology learning process can then serve as input to a more general ontology engineering methodology, allowing us to validate the terminological coverage of an ontology, to identify new classes and axioms, etc.

# 7 QUALITY ASSESSMENT

Independently of the (kinds of) source(s) from which a knowledge graph is created, data extracted for the initial knowledge graph will usually be incomplete, and will contain duplicate, contradictory or even incorrect statements – especially when taken from multiple sources. After the initial creation and enrichment of a knowledge graph from external sources, a crucial step is thus to assess the *quality* of the resulting knowledge graph. By quality, we here refer to *fitness for purpose*. Quality assessment then helps to ascertain for which purposes a knowledge graph can be reliability used.

In the following we discuss *quality dimensions* that capture aspects of multifaceted data quality which evolves from the traditional domain of databases to the domain of knowledge graphs [30], some of which are general, others of which are more particular to knowledge graphs [538]. While quality dimensions aim to capture qualitative aspects of the data, we also discuss *quality metrics* that provide ways to measure quantitative aspects of these dimensions. We discuss groupings of dimensions and metrics as inspired by Batini and Scannapieco [31].

## 7.1 Accuracy

Accuracy refers to the extent to which entities and relations – encoded by nodes and edges in the graph – correctly represent real-life phenomena. Accuracy can be further sub-divided into three dimensions: *syntactic accuracy*, *semantic accuracy*, and *timeliness*.

*7.1.1    Syntactic accuracy* is the degree to which the data are accurate with respect to the grammatical rules defined for the domain and/or data model. A prevalent example of syntactic inaccuracies occurs with datatype nodes, which may be incompatible with a defined range or be malformed. For example, assuming that a property start is defined with the range xsd:dateTime, taking a value such as `"March 29, 2019, 20:00"^^xsd:string` would be incompatible with the defined range, while a value `"March 29, 2019, 20:00"^^xsd:dateTime` would be malformed (a value such as `"2019-11-12T20:00:00"^^xsd:dateTime` is rather expected). A corresponding metric for syntactic accuracy is the ratio between the number of incorrect values of a given property and the total number of values for the same property [538]. Such forms of syntactic accuracy can typically be assessed using validation tools [161, 236].

*7.1.2    Semantic accuracy* is the degree to which data values correctly represent real world phenomena, which may be affected by imprecise extraction results, imprecise entailments, vandalism, etc. For instance, given that the National Congress of Chile is located in Valparaíso, this may give rise to the edge Chile —capital→ Valparaiso (through entailment, extraction, completion, etc.), which is in fact semantically inaccurate: the Chilean capital is Santiago. Assessing the level of semantic inaccuracies is challenging. While one option is to apply manual verification, an automatic option may be to check the stated relation against several sources [141, 293]. Another option is to rather validate the quality of individual processes used to generate the knowledge graph, based on measures such as precision, possibly with the help of human experts or gold standards [317].

*7.1.3    Timeliness* is the degree to which the knowledge graph is currently up-to-date with the real world state [260]; in other words, a knowledge graph may be semantically accurate now, but may quickly become inaccurate (outdated) if no procedures are in place to keep it up-to-date in a timely manner. For example, consider a user checking the tourist knowledge graph for flights from one city to another. Suppose that the flight timetable is updated every minute with current flight statuses, but the knowledge graph is only updated every hour. In this case, we see that there is a quality issue regarding timeliness in the knowledge graph. Timeliness can be assessed based on how frequently the knowledge graph is updated with respect to underlying sources [260, 427], which can be done using temporal annotations of changes in the knowledge graph [425, 426], as well as contextual representations that capture the temporal validity of data (see Section 3.3).

## 7.2    Coverage

Coverage refers to avoiding the omission of domain-relevant elements, which otherwise may yield incomplete query results or entailments, biased models, etc.

*7.2.1    Completeness* refers to the degree to which all required information is present in a particular dataset. Completeness comprises the following aspects: (i) *schema completeness* refers to the degree to which the classes and properties of a schema are represented in the data graph, (ii) *property completeness* refers to the ratio of missing values for a specific property, (iii) *population completeness* provides the percentage of all real-world entities of a particular type that are represented in the datasets, and (iv) *linkability completeness* refers to the degree to which instances in the data set are interlinked. Measuring completeness directly is non-trivial as it requires knowledge of a hypothetical *ideal knowledge graph* [111] that contains all the elements that the knowledge graph in question should "*ideally*" represent. Concrete strategies involve comparison with gold standards that provide samples of the ideal knowledge graph (possibly based on *completeness statements* [111]), or measuring the recall of extraction methods from complete sources [317], and so forth.

*7.2.2    Representativeness* is a related dimension that, instead of focusing on the ratio of domain-relevant elements that are missing, rather focuses on assessing high-level *biases* in what is included/excluded from the knowledge graph [23]. As such, this dimension assumes that the knowledge

graph is incomplete – i.e., that it is a sample of the ideal knowledge graph – and asks how biased this sample is. Biases may occur in the data, in the schema, or during reasoning [255]. Examples of data biases include geographic biases that under-represent entities/relations from certain parts of the world [255], linguistic biases that under-represent multilingual resources (e.g., labels and descriptions) for certain languages [261], social biases that under-represent people of particular genders or races [517], and so forth. In contrast, schema biases may result from high-level definitions extracted from biased data [255], semantic definitions that do not cover uncommon cases, etc. Unrecognised biases may lead to adverse effects; for example, if our tourism knowledge graph has a geographic bias towards events and attractions close to Santiago city – due perhaps to the sources used for creation, the employment of curators from the city, etc. – then this may lead to tourism in and around Santiago being disproportionally promoted (potentially compounding future biases). Measures of representativeness involve comparison of known statistical distributions with those of the knowledge graph, for example, comparing geolocated entities with known population densities [255], linguistic distributions with known distributions of speakers [261], etc. Another option is to compare the knowledge graph with general statistical laws, where Soulet et al. [464] use (non-)conformance with Benford's law[29] to measure representativeness in knowledge graphs.

## 7.3 Coherency

Coherency refers to how well the knowledge graph conforms to – or is coherent with – the formal semantics and constraints defined at the schema-level.

*7.3.1 Consistency* means that a knowledge graph is free of (logical/formal) contradictions with respect to the particular logical entailment considered. For example, in the ontology of our knowledge graph, we may define that (flight)–range→(Airport)–disj. c.→(City), which when combined with the edges (Arica)–flight→(Santiago)–type→(City), gives rise to an inconsistency, entailing that (Santiago) is a member of the disjoint classes (City) and (Airport). More generally, any semantic feature in Tables 2–4 with a "not" condition can give rise to inconsistencies if the negated condition is entailed. A measure of consistency can be the number of inconsistencies found in a knowledge graph, possibly sub-divided into the number of such inconsistencies identified by each semantic feature [54].

*7.3.2 Validity* means that the knowledge graph is free of constraint violations, such as captured by shape expressions [494] (see Section 3.1.2). We may, for example, specify a shape CITY whose target nodes have at most one country. Then, given the edges (Chile)◄country–(Santiago)–country►(Cuba), and assuming that (Santiago) becomes a target of CITY, we have a constraint violation. Conversely, even if we defined analogous cardinality restrictions in an ontology, this would not necessarily cause an inconsistency since, without UNA, we would first infer that (Chile) and (Cuba) refer to the same entity. A straightforward measure of validity is to count the number of violations per constraint.

## 7.4 Succinctness

Succinctness refers to the inclusion only of relevant content (avoiding "information overload") that is represented in a concise and intelligible manner.

*7.4.1 Conciseness* refers to avoiding the inclusion of schema and data elements that are irrelevant to the domain. Mendes et al. [328] distinguish *intensional conciseness* (schema level), which refers to the case when the data does not contain redundant schema elements (properties, classes, shapes, etc.), and *extensional conciseness* (data level), when the data does not contain redundant entities and relations. For example, including events in (Santiago de Cuba) in our knowledge graph dedicated to tourism in Chile may affect the extensional conciseness of the knowledge graph, potentially

---

[29]Benford's law states that the leading significant digit in many collections of numbers is more likely to be small.

returning irrelevant results for the given domain. In general, conciseness can be measured in terms of the ratio of properties, classes, shapes, entities, relations, etc., of relevance to the domain, which may in turn require a gold standard, or techniques to assess domain-relevance.

*7.4.2 Representational-conciseness* refers to the extent to which content is compactly represented in the knowledge graph, which may again be intensional or extensional [538]. For example, having two properties flight and flies to serving the same purpose would negatively affect the intensional form of representational conciseness, while having two nodes (Santiago) and (Santiago de Chile) representing the capital of Chile (with neither linked to the other) would affect the extensional form of representational conciseness. Another example of representational conciseness is the unnecessary use of complex modelling constructs, such as using reification unnecessarily, or using linked lists when the order of elements is not important [238]. Though representational conciseness is challenging to assess, measures such as the number of redundant nodes can be used [161].

*7.4.3 Understandability* refers to the ease with which data can be interpreted without ambiguity by human users, which involves – at least – the provision of human-readable labels and descriptions (preferably in different languages [261]) that allow them to understand what is being spoken about [238]. Referring back to Figure 1, though the nodes (EID15) and (EID16) are used to ensure unique identifiers for events, they should also be associated with labels such as (Ñam) and (Food Truck). Ideally the human readable information is sufficient to disambiguate a particular node, such as associating a description ("Santiago, the capital of Chile"@en) with (Santiago) to disambiguate the city from synonymous ones. Measures of understandability may include the ratio of nodes with human-readable labels and descriptions, the uniqueness of such labels and descriptions, the languages supported, etc.

## 7.5 Other Quality Dimensions

We have discussed some key quality dimensions that have been discussed for – and apply generally to – knowledge graphs. Further dimensions may be pertinent in the context of specific domains, specific applications, or specific graph data models. For further details, we refer to the survey by Zaveri et al. [538] and to the book by Batini and Scannapieco [31].

## 8 REFINEMENT

Beyond assessing the quality of a knowledge graph, there exist techniques to refine the knowledge graph, in particular to (semi-)automatically complete and correct the knowledge graph [375], aka *knowledge graph completion* and *knowledge graph correction*, respectively. As distinguished from the creation and enrichment tasks outlined in Section 6, refinement typically does not involve applying extraction or mapping techniques over external sources in order to ingest their content into the local knowledge graph. Instead, refinement typically targets improvement of the local knowledge graph as given (but potentially using external sources to verify local content [375]).

## 8.1 Completion

Knowledge graphs are characterised by incompleteness [523]. As such, knowledge graph completion aims at filling in the *missing edges* (aka *missing links*) of a knowledge graph, i.e., edges that are deemed correct but are neither given nor entailed by the knowledge graph. This task is often addressed with *link prediction* techniques proposed in the area of *Statistical Relational Learning* [177], which predict the existence – or sometimes more generally, predict the probability of correctness – of missing edges. For instance, one might predict that the edge (Moon Valley)— bus →(San Pedro) is a probable missing edge for the graph of Figure 21, given that most bus routes observed are return services (i.e., bus is typically symmetric). Link prediction may target three settings: *general links* involving edges with arbitrary labels, e.g., bus, flight, type, etc.; *type links* involving edges with

59

label type, indicating the type of an entity; and *identity links* involving edges with label same as, indicating that two nodes refer to the same entity (cf. Section 3.2.2). While type and identity links can be addressed using general link prediction techniques, the particular semantics of type and identity links can be addressed with custom techniques. (The related task of generating links across knowledge graphs – referred to as *link discovery* [352] – will be discussed later in Section 9.1.)

*8.1.1 General link prediction.* Link prediction, in the general case, is often addressed with inductive techniques as discussed in Section 5, and in particular, knowledge graph embeddings and rule/axiom mining. For example, given Figure 21, using knowledge graph embeddings, we may detect that given an edge of the form (x)—bus→(y), a (missing) edge (y)—bus→(x) has high plausibility, while using symbol-based approaches, we may learn the high-level rule ⟨x̃⟩—bus→⟨ỹ⟩ ⟹ ⟨ỹ⟩—bus→⟨x̃⟩. Either such approach would help us to predict the missing link (Moon Valley)—bus→(San Pedro).

*8.1.2 Type-link prediction.* Type links are of particular importance to a knowledge graph, where dedicated techniques can be leveraged taking into account the specific semantics of such links. In the case of type prediction, there is only one edge label (type) and typically fewer distinct values (classes) than in other cases, such that the task can be reduced to a traditional classification task [375], training models to identify each semantic class based on features such as outgoing and/or incoming edge labels on their instances in the knowledge graph [377, 461]. For example, assume that in Figure 21 we also know that (Arica), (Calama), (Puerto Montt), (Punta Arenas) and (Santiago) are of type (City). We may then predict that (Iquique) and (Easter Island) are also of type (City) based on the presence of edges labelled flight to/from these nodes, which (we assume) are learnt to be a good feature for prediction of that class (the former prediction is correct, while the latter is incorrect). Graph neural networks (see Section 5.3) can also be used for node classification/type prediction.

*8.1.3 Identity-link prediction.* Predicting identity links involves searching for nodes that refer to the same entity; this is analogous to the task of *entity matching* (aka record linkage, deduplication, etc.) considered in more general data integration settings [279]. Such techniques are generally based on two types of *matchers*: *value matchers* determine how similar the values of two entities on a given property are, which may involve similarity metrics on strings, numbers, dates, etc.; while *context matchers* consider the similarity of entities based on various nodes and edges [279]. An illustrative example is given in Figure 32, where value matchers will compute similarity between values such as (7400) and (7500), while context matchers will compute similarity between (Easter Island) and (Rapa Ñui) based on their surrounding information, such as their having similar latitudes, longitudes, populations, and the same seat (by way of comparison, a value matcher on this pair of nodes would measure string similarity between "Easter Island" and "Rapa Ñui").

A major challenge in this setting is efficiency, where a pairwise matching would require $O(n^2)$ comparisons for $n$ the number of nodes. To address this issue, *blocking* can be used to group similar entities into (possibly overlapping, possibly disjoint) "blocks" based on similarity-preserving keys, with matching performed within each block [128, 251, 279]; for example, if matching places based on latitude/longitude, blocks may represent geographic regions. An alternative to discrete blocking is to use *windowing* over entities in a similarity-preserving ordering [128], or to consider searching for similar entities within *multi-dimensional spaces* (e.g., spacetime [435], spaces with Minkowski distances [357], orthodromic spaces [358], etc. [452]). The results can either be pairs of nodes with a computed confidence of them referring to the same entity, or crisp identity links extracted based on a fixed threshold, binary classification, etc. [279]. For confident identity links, the nodes' edges may then be *consolidated* [239]; for example, we may select (Easter Island) as the canonical node and merge the edges of (Rapa Ñui) onto it, enabling us to find, e.g., *World Heritage Sites in the Pacific Ocean* from Figure 32 based on the (consolidated) sub-graph (World Heritage Site)◄—named—(Easter Island)—ocean→(Pacific).
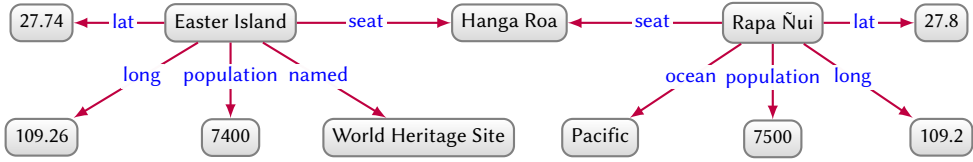
Fig. 32. Identity linking example, where Rapa Ñui and Easter Island refer to the same island

## 8.2 Correction

As opposed to completion – which finds new edges in a knowledge graph – correction identifies and removes existing incorrect edges in the knowledge graph. We here divide the principal approaches for knowledge graph correction into two main lines: *fact validation*, which assigns a plausibility score to a given edge, typically in reference to external sources; and *inconsistency repairs*, which aim to resolve inconsistencies found in the knowledge graph through ontological axioms.

*8.2.1 Fact validation.* The task of *fact validation* (aka *fact checking*) [59, 141, 175, 453, 456, 463, 486, 487, 536] involves assigning plausibility or *veracity* scores to facts/edges, typically between 0 and 1. An ideal fact-checking function assumes a hypothetical reference universe (an ideal knowledge graph) and would return 1 for the fact Santa Lucía –city→ Santiago (being true) while returning 0 for Sotomayor –city→ Santiago (being false). There is a clear relation between fact validation and link prediction – with both relying on assessing the plausibility of edges/facts/links – and indeed the same numeric- and symbol-based techniques can be applied for both cases. However, fact validation often considers online assessment of edges given as input, whereas link prediction is often an offline task that generates novel candidate edges to be assessed from the knowledge graph. Furthermore, works on fact validation are characterised by their consideration of external reference sources, which may be *unstructured sources* [175, 433, 486, 536] or *structured sources* [59, 453, 456, 463, 487].

Approaches based on unstructured sources assume that they are given a *verbalisation function* – using, for example, rule-based approaches [138, 359], encoder–decoder architectures [170], etc. – that is able to translate edges into natural language. Thereafter, approaches for computing the plausibility of facts in natural language – called *fact finders* [373, 374] – can be directly employed. Many fact finding algorithms construct an *n*-partite (often bipartite) graph whose nodes are facts and sources, where a source is connected to a fact if the source "evidences" the fact, i.e., if it contains a text snippet that matches – with sufficient confidence – the verbalisation of the input edge. Two mutually-dependent scores, namely the trustworthiness of sources and the plausibility of facts, are then calculated based on this graph, where fact finders differ on how they compute these scores [374]. Here we mention three scores proposed by Pasternack and Roth [373]:

- *Sums* [373] adapts the HITS algorithm [274] by defining sources as hubs (with 0 authority score) and facts as authorities (with 0 hub score).
- *Average Log* [373] extends HITS with a normalisation factor that prevents a single source from receiving a high trustworthiness score by evidencing many facts (that may be false).
- *Investment* [373] lets the scores of facts grow with a non-linear function based on "investments" coming from the connected sources. The score a source receives from a fact is based on the individual facts in this particular source compared to the other connected sources.

Pasternack and Roth [374] then show that these three algorithms can be generalised into a single multi-layered graph-based framework within which (1) a source can support a fact with a weight expressing uncertainty, (2) similar facts can support each other, and (3) sources can be grouped together leading to an implicit support between sources of the same group. Other approaches for

fact checking of knowledge graphs later extended this framework [165, 433]. Alternative approaches based on classifiers have also emerged, where commonly-used features include trust scores for information sources, co-occurrences of facts in sources, and so forth [175, 486].

Approaches for fact validation based on structured data typically assume external knowledge graphs as reference sources and are based on finding paths that evidence the input edge being validated. Unsupervised approaches search for undirected [87, 456] or directed [487] paths up to a given threshold length that evidence the input edge. The relatedness between input edges and paths is computed using a mutual information function, such as normalized pointwise mutual information [60]. Supervised approaches rather extract features for input edges from external knowledge graphs [289, 484, 541] and use these features to train a classification model to label the edges as true or false. An important set of features are *metapaths*, which encode sequences of predicates that correlate positively with the edge label of the input edge. Amongst such works, PredPath [453] automatically extracts metapaths based on type information. Several approaches rather encode the reference nodes and edges using graph embeddings (see Section 5.2), which are then used to estimate the plausibility of the input edge being validated.

*8.2.2 Inconsistency repairs.* Ontologies can contain axioms – such as disjointness – that lead to inconsistencies. While such axioms can be provided by experts, they can can also be derived through symbolic learning, as discussed in Section 5.4. Such axioms can then be used to detect inconsistencies. With respect to correcting a knowledge graph, however, detecting inconsistencies is not enough: techniques are also required to *repair* such inconsistencies, which itself is not a trivial task. In the simplest case, we may have an instance of two disjoint classes, such as that (Santiago) is of type (City) and (Airport), which are stated or found to be disjoint. To repair the inconsistency, it would be preferable to remove only the "incorrect" class, but which should we remove? This is not a trivial question, particularly if we consider that one edge can be involved in many inconsistencies, and one inconsistency can involve many edges. The issue of computing repairs becomes more complex when entailment is considered, where we not only need to remove the stated type, but also all of the ways in which it might be entailed; for example, removing the edge (Santiago)–type→(Airport) is insufficient if we further have an edge (Arica)–flight→(Santiago) combined with an axiom (flight)–range→(Airport). Töpper et al. [497] suggest potential repairs for such violations – remove a domain/range constraint, remove a disjointness constraint, remove a type edge, remove an edge with a domain/range constraint – where one is chosen manually. In contrast, Bonatti et al. [54] propose an automated method to repair inconsistencies based on *minimal hitting sets* [411], where each set is a minimal explanation for an inconsistency. The edges to remove are chosen based on scores of the trustworthiness of their sources and how many minimal hitting sets they are either elements of or help to entail an element of, where the knowledge graph is revised to avoid re-entailment of the removed edges. Rather than repairing the data, another option is to evaluate queries under inconsistency-aware semantics, such as returning *consistent answers* valid under every possible repair [309].

## 8.3 Other refinement tasks

In comparison to the quality clusters discussed in Section 7, the refinement methods discussed here address particular aspects of the accuracy, coverage, and coherency aspects. Beyond these, one could conceive of further refinement methods to address further quality issues of knowledge graphs, such as succinctness. In general, however, the refinement tasks of *knowledge graph completion* and *knowledge graph correction* have received the majority of attention until now. For further details on knowledge graph refinement, we refer to the survey by Paulheim [375].

# 9  PUBLICATION

While it may not be desirable to publish, for example, enterprise knowledge graphs that offer a competitive advantage to a company [365], it may be desirable – or even required – to publish other knowledge graphs, such as those produced by volunteers [291, 313, 515], by publicly-funded research [79, 193, 492], by governmental organisations [222, 450], etc. Publishing refers to making the knowledge graph (or part thereof) accessible to the public, often over the Web. Knowledge graphs published as open data are then called open knowledge graphs (discussed in Section 10.1).

In the following, we first discuss two sets of principles that have been proposed to guide the publication of data on the Web. We next discuss access protocols that constitute the interfaces by which the public can interact with the content of a knowledge graph. Finally, we consider techniques to restrict the access or usage of (parts of) a knowledge graph, as appropriate.

## 9.1  Best Practices

We now discuss two key sets of principles for publishing data, namely the FAIR Principles proposed by Wilkinson et al. [524], and the Linked Data Principles proposed by Berners-Lee [37].

*9.1.1  FAIR Principles.* The FAIR Principles were originally proposed in the context of publishing scientific data [524] – particularly motivated by maximising the impact of publicly-funded research – but the principles generally apply to other situations where data are to be published in a manner that facilitates their re-use by external agents, with particular emphasis on machine-readability.

FAIR itself is an acronym for four foundational principles, each with particular goals [524], that may apply to *data*, *metadata*, or both – the latter being denoted *(meta)data*.[30] We now describe the FAIR principles (slightly rephrasing the original wording in some cases for brevity [524]).

- *Findability* refers to the ease with which external agents who might benefit from the dataset can initially locate the dataset. Four sub-goals should be met:
  - F1: (meta)data are assigned a globally unique and persistent identifier.
  - F2: data are described with rich metadata (see R1).
  - F3: metadata clearly and explicitly include the identifier of the data they describe.
  - F4: (meta)data are registered or indexed in a searchable resource.
- *Accessibility* refers to the ease with which external agents (once they have located the dataset) can access the dataset. Two goals are defined, the first with two sub-goals:
  - A1: (meta)data are retrievable by their identifier using a standard protocol.
    - A1.1: the protocol is open, free, and universally implementable.
    - A1.2: the protocol allows for authentication and authorisation, where necessary.
  - A2. metadata are accessible, even when the data are no longer available.
- *Interoperability* refers to the ease with which the dataset can be exploited (in conjunction with other datasets) using standard tools. Three goals are defined:
  - I1: (meta)data use an accessible, shared, and general knowledge representation formalism.
  - I2: (meta)data use vocabularies that follow FAIR principles.
  - I3: (meta)data include qualified references to other (meta)data.
- *Reusability* refers to the ease with which the dataset can be re-used in conjunction with other datasets. One goal is defined (with three sub-goals):
  - R1: meta(data) are richly described with a plurality of accurate and relevant attributes.
    - R1.1. (meta)data are released with a clear and accessible data usage license.
    - R1.2. (meta)data are associated with detailed provenance.

---

[30]Metadata are data about data. The distinction is often important in observational sciences, where in astronomy, for example, data may include raw image data, while metadata may include the celestial coordinates and time of the image.
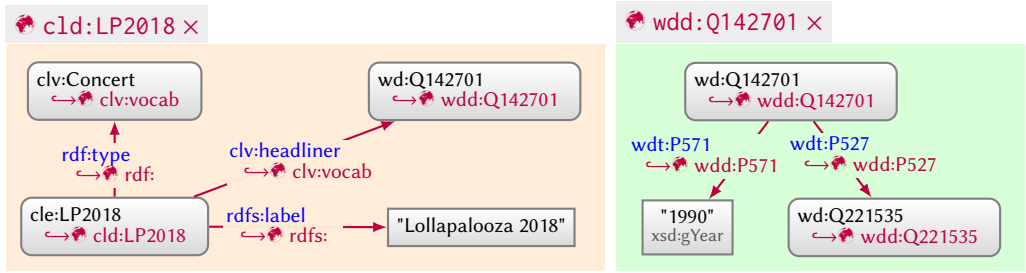
Fig. 33. Two example Linked Data documents from two websites, each containing an RDF graph, where wd:Q142701 refers to Pearl Jam in Wikidata while wdd:Q142701 to the RDF graph about Pearl Jam, and where wd:Q221535 refers to Eddie Vedder while wdd:Q221535 refers to the RDF graph about Eddie Vedder; the edge-label wdt:571 refers to "inception" in Wikidata, while wdt:527 refers to "has part"

     ∘ R1.3. (meta)data meet domain-relevant community standards.

In the context of knowledge graphs, a variety of vocabularies, tools, and services have been proposed that both directly and indirectly help to satisfy the FAIR principles. In terms of *Findability*, as discussed in Section 2, IRIs are built into the RDF model, providing a general schema for global identifiers. In addition, resources such as the Vocabulary of Interlinked Datasets (VoID) [9] allow for representing meta-data about graphs, while services such as DataHub [41] provide a central repository of such dataset descriptions. Access protocols that enable *Accessibility* will be discussed in Section 9.2, while mechanisms for authorisation will be discussed in Section 9.3. With respect to *Interoperability*, as discussed in Section 4, ontologies serve as a general knowledge representation formalism, and can in turn be used to describe vocabularies that follow FAIR principles. Finally, regarding *Reusability*, licensing will be discussed in Section 9.3, while the *PROV Data Model* [181] discussed in Section 3 allows for capturing detailed provenance.

A number of knowledge graphs have been published using FAIR principles, where Wilkinson et al. [524] explicitly mention Open PHACTS [193], a data integration platform for drug discovery, and UniProt [492], a large collection of protein sequence and annotation data, as conforming to FAIR principles. Both datasets offer graph views of their content through the RDF data model.

*9.1.2 Linked Data Principles.* Wilkinson et al. [524] state that FAIR Principles "precede implementation choices", meaning that the principles do not cover *how* they can or should be achieved. Preceding the FAIR Principles by almost a decade are the Linked Data Principles, proposed by Berners-Lee [37], which provide a technical basis for one possible way in which these FAIR Principles can be achieved. Specifically the Linked Data Principles are as follows:

(1) Use IRIs as names for things.
(2) Use HTTP IRIs so those names can be looked up.
(3) When a HTTP IRI is looked up, provide useful content about the entity that the IRI names using standard data formats.
(4) Include links to the IRIs of related entities in the content returned.

These principles were proposed in a Semantic Web setting, where for principle (3), the standards based on RDF (including RDFS, OWL, etc.) are currently recommended for use, particularly because they allow for naming entities using HTTP IRIs, which further paves the way for satisfying all four principles. As such, these principles outline a way in which (RDF) graph-structured data can be published on the Web such that these graphs are interlinked to form what Berners-Lee [37] calls a "Web of Data", whose goal is to increase automation on the Web by making content available not

only in (HTML) documents intended for human consumption, but also as (RDF) structured data that machines can locate, retrieve, combine, validate, reason over, query over, etc., towards solving tasks automatically. Conceptually, the Web of Data is then composed of graphs of data published on individual web-pages, where one can click on a node or edge-label – or more precisely perform a HTTP lookup on an IRI of the graph – to be transported to another graph elsewhere on the Web with relevant content on that node or edge-label, and so on recursively.

In Figure 33, we show a simple example with two Linked Data documents published on the Web, with each containing an RDF graph. As discussed in Section 3.2, terms such as clv:Concert, wd:Q142701, rdfs:label, etc., are abbreviations for IRIs, where, for example, wd:Q142701 expands to http://www.wikidata.org/entity/Q142701. Prefixes beginning with cl are fictitious prefixes we assume to have been created by the Chilean tourist board. The IRIs prefixed with ↪🏠 indicate the document returned if the node is looked up. The leftmost document is published by the tourist board and describes Lollapalooza 2018 (identified by the node (cle:LP2018)), which links to the headlining act Pearl Jam ((wd:Q142701)) described by an external knowledge graph, namely Wikidata. By looking up the node (wd:Q142701) in the leftmost graph, the IRI *dereferences* (i.e., returns via HTTP) the document with the RDF graph on the right describing that entity in more detail. From the rightmost document, the node (wd:Q221535) can be looked up, in turn, to find a graph about Eddie Vedder (not shown in the example). The IRIs for entities and documents are distinguished to ensure that we do not confuse data about the entity and the document; for example, while wd:Q221535 refers to Eddie Vedder, the IRI wd**d**:Q221535 refers to the document about Eddie Vedder; if we were to assign a last-modified date to the document, we should use (wd**d**:Q221535) not (wd:Q221535). In Figure 33, we can further observe that edge labels (which are also IRIs) and nodes representing classes (e.g., (clv:Concert)) can also be dereferenced, typically returning semantic definitions of the respective terms.

A key challenge is posed by the fourth principle – include links to related entities – as illustrated in Figure 33, where (wd:Q221535) in the leftmost graph constitutes a link to related content about Pearl Jam in an external knowledge graph. Specifically, the *link discovery* task considers adding such links from one knowledge graph to another, which may involve inclusion of IRIs that dereference to external graphs (per Figure 33), or links with special semantics such as identity links. In comparison with the link prediction task discussed in Section 8.1, which is used to complete links within a knowledge graph, link discovery aims to discover links across knowledge graphs, which involves unique aspects: first, link discovery typically considers disjoint sets of source (local) nodes and target (remote) nodes; second, the knowledge graphs may often use different vocabularies; third, while in link prediction there already exist local examples of the links to predict, in link discovery, there are often no existing links between knowledge graphs to learn from. A common technique is to define manually-crafted linkage rules (aka link specifications) that apply heuristics for defining links that potentially incorporate similarity measures [356, 513]. Link discovery is greatly expedited by the provision of standard identifier schemes within knowledge graphs, such as ISBNs for books, alpha-2 and alpha-3 codes for countries (e.g., CL, CLP), or even links to common knowledge graphs such as DBpedia [291] or Wikidata [515] (that themselves include standard identifiers). We refer to the survey on link discovery by Nentwig et al. [352] for more details.

Further guidelines have been proposed that provide finer-grained recommendations for publishing Linked Data, relating to how best to implement dereferencing, what kinds of links to include, how to publish and interlink vocabularies, amongst other considerations [216, 254]. We refer to the book by Heath and Bizer [216] for more discussion on how to publish Linked Data on the Web.
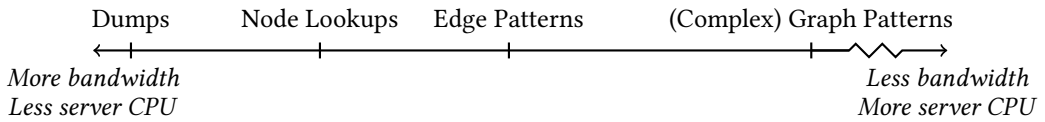
Fig. 34. Access protocols for knowledge graphs, from simple protocols (left) to more complex protocols (right)

## 9.2 Access Protocols

Publishing involves allowing the public to interact with the knowledge graph, which implies the provision of *access protocols* that define the requests that agents can make and the response that they can expect as a result. Per the *Accessibility* principle of FAIR (specifically A1.1), this protocol should be open, free, and universally implementable. In the context of knowledge graphs, as shown in Figure 34, there are a number of access protocols to choose from, varying from simple protocols that allow users to simply download all content, towards protocols that accept and evaluate increasingly complex requests. While simpler protocols require less computation on the server that publishes the data, more complex protocols allow agents to request more specific data, thus reducing bandwidth. A knowledge graph may also offer a variety of access protocols catering to different agents with different requirements [508]. We now discuss such access protocols.

*9.2.1 Dumps.* A dump is a file or collection of files containing the content of the knowledge graph available for download. The request in this case is for the file(s) and the response is the content of the file(s). In order to publish dumps, first of all, concrete – and ideally standard – syntaxes are required to serialise the graph. While for RDF graphs there are various standard syntaxes available based on XML [166], JSON [469], custom syntaxes [397], and more besides, currently there are only non-standard syntaxes available for property graphs [496]. Second, to reduce bandwidth, compression methods can be applied. While standard compression such as GZIP or BZip2 can be straightforwardly applied on any file, custom compression methods have been proposed for graphs that not only offer better compression ratios than these standard methods, but also offer additional functionalities, such as compact indexes for performing efficient lookups once the file is downloaded [150]. Finally, to further reduce bandwidth, when the knowledge graph is updated, "diffs" can be computed and published to obviate the need for agents to download all data from scratch (see [5, 372, 502]). Still, however, dumps are only suited to certain use-cases, in particular for agents that wish to maintain a full local copy of a knowledge graph. If an agent were rather only interested in, for example, all food festivals in Santiago, downloading the entire dump would require transferring and processing a lot of irrelevant data.

*9.2.2 Node lookups.* Protocols for performing node lookups accept a node (id) request (e.g., cle:LP2018 in Figure 33) and return a (sub-)graph describing that node (e.g., the document cld:LP2018). Such a protocol is the basis for the Linked Data principles outlined previously, where node lookups are implemented through HTTP dereferencing, which further allows nodes in remote graphs to be referenced from across the Web. Although there are varying definitions on what content should be returned for a node [474], a common convention is to return a sub-graph containing either all outgoing edges for that node or all incident edges (both outgoing and incoming) for that node [238]. Though simple, mechanisms for answering graph patterns can be implemented on top of a node lookup interface by traversing from node to node according to the particular graph pattern [209]; for example, to find all food festivals in Santiago – represented by the graph pattern Food Festival ←type– ?ff –location→ Santiago – we may perform a node lookup for Santiago, subsequently performing a node lookup for each node connected by a location edge to Santiago, returning those

nodes declared to be of type (Food Festival). However, such an approach may not be feasible if no starting node is declared (e.g., if all nodes are variables), if the node lookup service does not return incoming edges, etc. Furthermore, the client agent may need to request more data than necessary, where the document returned for (Santiago) may return a lot of irrelevant data, and where nodes with a location in (Santiago) that do not represent instances of (Food Festival) still need to be looked up to check their type. On the plus side, node lookups are relatively inexpensive for servers to support.

*9.2.3 Edge patterns.* Edge patterns – also known as *triple patterns* in the case of directed, edge-labelled graphs – are singleton graph patterns, i.e., graph patterns with a single edge. Examples of edge patterns are (?ff)–type→(Food Festival) or (?ff)–location→(Santiago), etc., where any term can be a variable or a constant. A protocol for edge patterns accepts such a pattern and returns all solutions for the pattern. Edge patterns provide more flexibility than node lookups, where graph patterns are more readily decomposed into edge patterns than node lookups. With respect to the agent interested in food festivals in Santiago, they can first, for example, request solutions for the edge pattern (?ff)–location→(Santiago) and locally join/intersect these solutions with those of (?ff)–type→(Food Festival). Given that some edge patterns (e.g., (?x)–?y→(?z)) can return many solutions, protocols for edge patterns may offer additional practical features such as iteration or pagination over results [509]. Much like node lookups, the server cost of responding to a request is relatively low and easy to predict. However, the server may often need to transfer irrelevant intermediate results to the client, which in the previous example may involve returning nodes located in Santiago that are not food festivals. This issue is further aggravated if the client does not have access to statistics about the knowledge graph in order to plan how to best perform the join; for example, if there are relatively few food festivals but many things located in Santiago, rather than intersecting the solutions of the two aforementioned edge patterns, it should be more efficient to send a request for each food festival to see if it is in Santiago, but deciding this requires statistics about the knowledge graph. Extensions to the edge-pattern protocol have thus been proposed to allow for more efficient joins [211], such as allowing batches of solutions to be sent alongside the edge pattern, returning only solutions compatible with the solutions in the request [210] (e.g., sending a batch of solutions for (?ff)–type→(Food Festival) to join with the solutions for the request (?ff)–location→(Santiago)).

*9.2.4 (Complex) graph patterns.* Another alternative is to let client agents make requests based on (complex) graph patterns (see Section 2.2), with the server returning (only) the final solutions. In our running example, this involves the client issuing a request for (Food Festival)←type–(?ff)–location→(Santiago) and directly receiving the relevant results. Compared with the previous protocols, this protocol is much more efficient in terms of bandwidth: it allows clients to make more specific requests and the server to return more specific responses. However, this reduction in bandwidth use comes at the cost of the server having to evaluate much more complex requests, where, furthermore, the costs of a single request are much more difficult to anticipate. While a variety of optimised engines exist for evaluating (complex) graph patterns (e.g., [139, 332, 493] amongst many others), the problem of evaluating such queries is known to be intractable [14]. Perhaps for this reason, public services offering such a protocol (most often supporting SPARQL queries [207]) have been found to often exhibit downtimes, timeouts, partial results, slow performance, etc. [72]. Even considering such issues, however, popular services continue to receive – and successfully evaluate – millions of requests/queries per day [315, 432], with difficult (worst-case) instances being rare in practice [58].

*9.2.5 Other protocols.* While Figure 34 makes explicit reference to some of the most commonly-encountered access protocols found for knowledge graphs in practice, one may of course imagine other protocols lying almost anywhere on the spectrum from more simple to more complex interfaces. To the right of (Complex) Graph Patterns, one could consider supporting even more complex
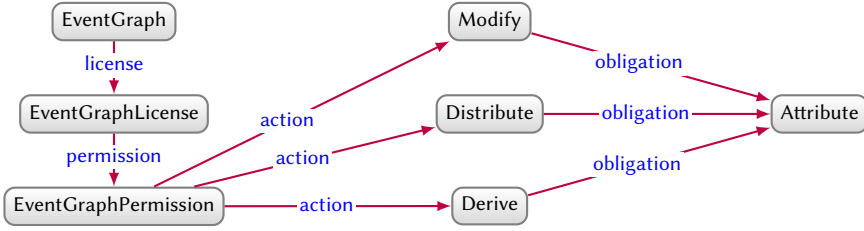
Fig. 35. Associating licenses with event data, along with permissions, actions, and obligations

requests, such as queries with entailments [184], queries that allow recursion [414], federated queries that can join results from remote services [71], or even (hypothetically) supporting Turing-complete requests that allow running arbitrary procedural code on a knowledge graph. As mentioned at the outset, a server may also choose to support multiple, complementary protocols [508].

## 9.3 Usage Control

Considering our hypothetical tourism knowledge graph, at first glance, one might assume that the knowledge required to deliver the envisaged services is public and thus can be used both by the tourism board and the tourists. On closer inspection, however, we may see the need for usage control in various forms: (i) both the tourist board and its partners should associate an appropriate license with knowledge that they contribute to the knowledge graph, such that the terms of use are clear to all parties; (ii) a tourist might opt to install an app on their mobile phone that could be used to recommend tourist attractions based on their location, bringing with it potential privacy concerns; (iii) the tourist board may be required to report criminal activities to the police services and thus may need to encrypt personal information; and (iv) the tourist board could potentially share information relating to tourism demographics in an anonymous format to allow for improving transport infrastructure on strategic routes. Thus in this section, we examine the state of the art in terms of knowledge graph licensing, usage policies, encryption, and anonymisation.

*9.3.1 Licensing.* When it comes to associating machine readable licenses with knowledge graphs, the W3C Open Digital Rights Language (ODRL) [247] provides an information model and related vocabularies that can be used to specify permissions, duties, and prohibitions with respect to actions relating to assets. ODRL supports fine-grained descriptions of digital rights that are represented as – and thus can be embedded within – graphs. Figure 35 illustrates a license granting the assignee the permission to (Modify), (Distribute), and (Derive) work from the (EventGraph) (e.g., Figure 1); however the assignee is obliged to (Attribute) the copyright holder. From a modelling perspective, ODRL can be used to model several well known license families, for instance Apache, Creative Commons (CC), and Berkeley Software Distribution (BSD), to name but a few [77, 370]. Additionally, Cabrio et al. [77] propose methods to automatically extract machine-readable licenses from unstructured text. From a reasoning perspective, license compatibility validation and composition techniques [191, 338, 511] can be used to combine knowledge graphs that are governed by different licenses. Such techniques are employed by the the Data Licenses Clearance Center (DALICC), which includes a library of standard machine readable licenses, and tools that enable users both to compose arbitrary custom licenses and also to verify the compatibility of different licenses [380].

*9.3.2 Usage policies.* Access control policies based on edge patterns can be used to restrict access to parts of a knowledge graph [156, 272, 410]. WebAccessControl (WAC)[31] is an access control

---
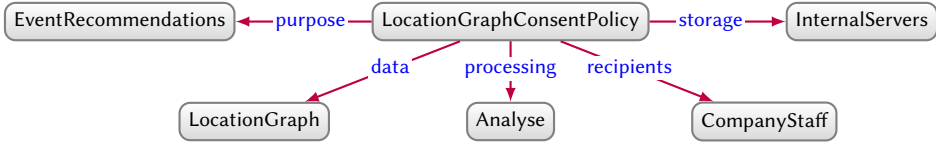
Fig. 36. A policy for the usage of a sub-graph of location data in the knowledge graph
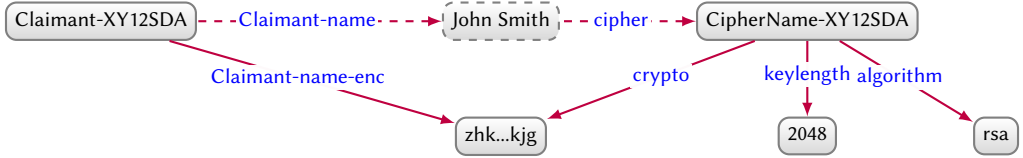


Fig. 37. Directed edge-labelled graph with the name of the claimant encrypted; plaintext elements are dashed and may be omitted from published data (possibly along with encryption details)

framework for graphs that uses WebID for authentication and provides a vocabulary for specifying access control policies. Extensions of this WAC vocabulary have been proposed to capture privacy preferences [428] and to cater for contextual constraints [101, 510]. Although ODRL is primarily used to specify licenses, profiles to specify access policies [473] and regulatory obligations [3, 117] have also been proposed in recent years, as discussed in the survey by Kirrane et al. [273].

As a generalisation of access policies, usage policies specify how data can be used: what kinds of processing can be applied, by whom, for what purpose, etc. The example usage policy presented in Figure 36 states that the process Analyse of LocationGraph can be performed on InternalServers by members of CompanyStaff in order to provide EventRecommendations. Vocabularies for usage policies have been proposed by the SPECIAL H2020 project [52] and the W3C Data Privacy Vocabularies and Controls Community Group (DPVCG) [55, 371]. Once specified, usage policies can then be used to verify that data processing conforms to legal norms and the consent provided by subjects [55, 119].

*9.3.3  Encryption.* Rather than internally controlling usage, the tourist board could use encryption mechanisms on parts of the published knowledge graph, for example relating to reports of crimes, and provide keys to partners who should have access to the plaintext. While a straightforward approach is to encrypt the entire graph (or sub-graphs) with one key, more fine-grained encryption can be performed for individual nodes or edge-labels in a graph, potentially providing different clients access to different information through different keys [180]. The CryptOntology [176] can further be used to embed details about the encryption mechanism used within the knowledge graph. Figure 37 illustrates how this could be used to encrypt the names of claimants from Figure 31, storing the ciphertext (zhk...kjg), as well as the key-length and encryption algorithm used. In order to grant access to the plaintext, one approach is to encrypt individual edges with symmetric keys so as to allow specific types of edge patterns to only be executed by clients with the appropriate key [265]. This approach can be used, for example, to allow clients who know a claimant ID (e.g., Claimant-XY12SDA) and have the appropriate key to find (only) the name of the claimant through an edge pattern Claimant-XY12SDA–Claimant-name→?name. A key limitation of this approach, however, is that it requires attempting to decrypt all edges to find all possible solutions. A more efficient alternative is to combine functional encryption and specialised indexing to retrieve solutions from the encrypted graph without attempting to decrypt all edges [149].
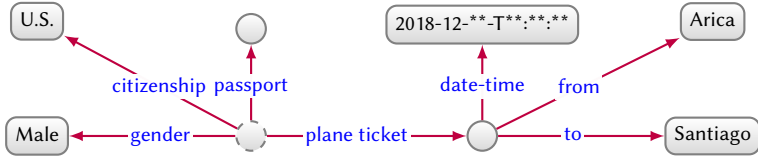
Fig. 38. Anonymised sample of a directed edge-labelled graph describing a passenger (dashed) of a flight

*9.3.4 Anonymisation.* Consider that the tourist board acquires information on transport taken by individuals within the country, which can be used to understand trajectories taken by tourists. However, from a data-protection perspective, it would be advisable to remove any personal data from the knowledge graph to avoid leaks of information about each individual's travel.

A first approach to anonymisation is to suppress and generalise knowledge in a graph such that individuals cannot be identified, based on $k$-anonymity [434][32], $l$-diversity [296][33], etc. Approaches to apply $k$-anonymity on graphs identify and suppress "quasi-identifiers" that would allow a given individual to be distinguished from fewer than $k - 1$ other individuals [219, 404]. Figure 38 illustrates a possible result of $k$-anonymisation for a sub-graph describing a flight passenger, where quasi-identifiers (passport, plane ticket) have been converted into blank nodes, ensuring that the passenger (the dashed blank node) cannot be distinguished from $k - 1$ other individuals. In the context of a graph, however, *neighbourhood attacks* [544] – using information about neighbours – can also break $k$-anonymity, where we also suppress the day and time of the flight, which, though not sensitive information per se, could otherwise break $k$-anonymity for passengers (if, for example, a particular flight had fewer than $k$ males from the U.S. onboard).

More complex neighbourhood attacks may rely on more abstract graph patterns, observing that individuals can be deanonymised purely from knowledge of the graph structure, even if all nodes and edge labels are left blank; for example, if we know that a team of $k - 1$ players take flights together for a particular number of away games, we could use this information for a neighbourhood attack that reveals the set of players in the graph. Hence a number of guarantees specific to graphs have been proposed, including $k$-degree anonymity [301], which ensures that individuals cannot be deanonymised by attackers with knowledge of the degree of particular individuals. The approach is based on minimally modifying the graph to ensure that each node has at least $k - 1$ other nodes with the same degree. A stronger guarantee, called $k$-isomorphic neighbour anonymity [543], avoids neighbourhood attacks where an attacker knows how an individual is connected to nodes in their neighbourhood; this is done by modifying the graph to ensure that for each node, there exist at least $k - 1$ nodes with isomorphic (i.e., identically structured) neighbourhoods elsewhere in the graph. Both approaches only protect against attackers with knowledge of bounded neighbourhoods. An even stronger notion is that of $k$-automorphism [547], which ensures that for every node, it is structurally indistinguishable from $k - 1$ other nodes, thus avoiding any attack based on structural information (as a trivial example, a $k$-clique or a $k$-cycle satisfy $k$-automorphism). Many of these techniques for anonymisation of graph data were originally motivated by social networks [349], though they can also be applied to knowledge graphs, per the work of Lin and Tripunitara [299], who adapt $k$-automorphism for directed edge-labelled graphs (specifically RDF graphs).

While the aforementioned approaches anonymise data, a second approach is to apply anonymisation when answering queries, such as adding noise to the solutions in a way that preserves

---

[32]$k$-anonymity guarantees that the data of an individual is indistinguishable from at least $k - 1$ other individuals.

[33]$l$-diversity guarantees that sensitive data fields have at least $l$ diverse values within each group of individuals; this avoids leaks such as that all tourists from Austria (a group of individuals) in the data have been pick-pocketed (a sensitive attribute).

privacy. One approach is to apply $\varepsilon$-differential privacy [132][34] for querying graphs [458]. Such mechanisms are typically used for aggregate (e.g., count) queries, where noise is added to avoid leaks about individuals. To illustrate, differential privacy may allow for counting the number of passengers of specified nationalities taking specified flights, adding (just enough) random noise to the count to ensure that we cannot tell, within a certain probability (controlled by $\varepsilon$), whether or not a particular individual took a flight, where we would require (proportionally) less noise for common nationalities, but more noise to "hide" individuals from more uncommon nationalities.

These approaches require information loss for stronger guarantees of privacy; which to choose is thus heavily application dependent. If the anonymised data are to be published in their entirety "dump", then an approach based on $k$-anonymity can be used to protect individuals, while $l$-diversity can be used to protect groups. On the other hand, if the data are to be made available, in part, through a query interface, then $\varepsilon$-differential privacy is a more suitable framework.

## 10 KNOWLEDGE GRAPHS IN PRACTICE

In this section, we discuss some of the most prominent knowledge graphs that have emerged in the past years. We begin by discussing open knowledge graphs, which have been published on the Web per the guidelines and protocols described in Section 9. We later discuss enterprise knowledge graphs that have been created by companies for a diverse range of applications.

### 10.1 Open Knowledge Graphs

By *open knowledge graphs*, we specifically refer to knowledge graphs published under the Open Data philosophy, namely that "*open means anyone can freely access, use, modify, and share for any purpose (subject, at most, to requirements that preserve provenance and openness)*".[35] Many open knowledge graphs have been published in the form of *Linked Open Datasets* [216], which are (RDF) graphs published under the Linked Data principles (see Section 9.1.2) following the Open Data philosophy. Many of the most prominent open knowledge graphs – including DBpedia [291], YAGO [481], Freebase [50], and Wikidata [515] – cover multiple domains, representing a broad diversity of entities and relationships; we first discuss these in turn. Later we discuss some of the other (specific) domains for which open knowledge graphs are currently available. Most of the open knowledge graphs we discuss in this section are modelled in RDF, published following Linked Data principles, and offer access to their data through dumps (RDF), node lookups (Linked Data), graph patterns (SPARQL) and, in some cases, edge patterns (Triple Pattern Fragments).

*10.1.1 DBpedia.* The DBpedia project was developed to extract a graph-structured representation of the semi-structured data embedded in Wikipedia articles [20], enabling the integration, processing, and querying of these data in a unified manner. The resulting knowledge graph is further enriched by linking to external open resources, including images, webpages, and external datasets such as DailyMed, DrugBank, GeoNames, MusicBrainz, New York Times, and WordNet [291]. The DBpedia extraction framework consists of several components, corresponding to abstractions of Wikipedia article sources, graph storage and serialisation destinations, wiki-markup extractors, parsers, and extraction managers [44]. Specific extractors are designed to process labels, abstracts, interlanguage links, images, redirects, disambiguation pages, external links, internal pagelinks, homepages, categories, and geocoordinates. The content in the DBpedia knowledge graph is not only multidomain, but also multilingual: as of 2012, DBpedia contained labels and abstracts in up to 97 different languages [327]. Entities within DBpedia are classified using four different schemata in

---

[34]$\varepsilon$-differential privacy ensures that the probability of achieving a given result from some process (e.g., query) applied to data, to which random noise is added, differs no more than $e^{\varepsilon}$ when the data includes or excludes any individual.
[35]See http://opendefinition.org/

order to address varying application requirements [44]. These schemata include a Simple Knowledge Organization System (SKOS) representation of Wikipedia categories, a Yet Another Great Ontology (YAGO) classification schema (discussed in the following), an Upper Mapping and Binding Exchange Layer (UMBEL) ontology categorisation schema, and a custom schema called the DBpedia ontology with classes such as `Person`, `Place`, `Organisation`, and `Work` [291]. DBpedia also supports live synchronisation in order to remain consistent with dynamic Wikipedia article [291].

*10.1.2  Yet Another Great Ontology.* YAGO likewise extracts graph-structured data from Wikipedia, which are then unified with the hierarchical structure of WordNet to create a "*light-weight and extensible ontology with high quality and coverage*" [481]. This knowledge graph aims to be applied for various information technology tasks, such as machine translation, word sense disambiguation, query expansion, document classification, data cleaning, information integration, etc. While earlier approaches automatically extracted structured knowledge from text using pattern matching, natural language processing (NLP), and statistical learning, the resulting content tended to lack in quality when compared with what was possible through manual construction [481]. However, manual construction is costly, making it challenging to achieve broad coverage and keep the data up-to-date. In order to extract data with high coverage and quality, YAGO (like DBpedia) mostly extracts data from Wikipedia infoboxes and category pages, which contain basic entity information and lists of articles for a specific category, respectively; these, in turn are unified with hierarchical concepts from WordNet [482]. A schema – called the YAGO model – provides a vocabulary defined in RDFS; this model allows for representing words as entities, capturing synonymy and ambiguity [481]. The model further supports reification, *n*-ary relations, and data types [482]. Refinement mechanisms employed within YAGO include canonicalisation, where each edge and node is mapped to a unique identifier and duplicate elements are removed, and type checking, where nodes that cannot be assigned to a class by deductive or inductive methods are eliminated [482]. YAGO would be extended in later years to support spatio-temporal context [232] and multilingual Wikipedias [313].

*10.1.3  Freebase.* Freebase was a general collection of human knowledge that aimed to address some of the large scale information integration problems associated with the decentralised nature of the Semantic Web, such as uneven adoption, implementation challenges, and distributed query performance limitations [51]. Unlike DBpedia and YAGO – which are mostly extracted from Wikipedia/WordNet – Freebase solicited contributions directly from human editors. Included in the Freebase platform were a scalable data store with versioning mechanisms; a large data object store (LOB) for the storage of text, image, and media files; an API that could be queried using the Metaweb Query Language (MQL); a Web user interface; and a lightweight typing system [51]. The latter typing system was designed to support collaborative processes. Rather than forcing ontological correctness or logical consistency, the system was implemented as a loose collection of structuring mechanisms – based on datatypes, semantic classes, properties, schema definitions, etc. – that allowed for incompatible types and properties to coexist simultaneously [51]. Content could be added to Freebase interactively through the Web user interface or in an automated way by leveraging the API's write functionality. Freebase had been acquired by Google in 2010, where the content of Freebase formed an important part of the Google Knowledge Graph announced in 2012 [459]. When Freebase became read-only as of March 2015, the knowledge graph contained over three billion edges. Much of this content was subsequently migrated to Wikidata [382].

*10.1.4  Wikidata.* As exploited by DBpedia and YAGO, Wikipedia contains a wealth of semi-structured data embedded in info-boxes, lists, tables, etc. However, these data have traditionally been curated and updated manually across different articles and languages; for example, a goal scored by a Chilean football player may require manual updates in the player's article, the tournament article,

the team article, lists of top scorers, and so forth, across hundreds of language versions. Manual curation has led to a variety of data quality issues, including contradictory data in different articles, languages, etc. The Wikimedia Foundation thus proposed Wikidata as a centralised, collaboratively-edited knowledge graph to supply Wikipedia – and arbitrary other clients – with data. Under this vision, a fact could be added to Wikidata once, triggering the automatic update of potentially multitudinous articles in Wikipedia across different languages [515]. Like Wikipedia, Wikidata is also considered a secondary source containing *claims* that should reference primary sources, though claims can also be initially added without reference [390]. Wikidata further allows for different viewpoints in terms of potentially contradictory (referenced) claims [515]. Wikidata is multilingual, where nodes and edges are assigned language-agnostic Qxx and Pxx codes (see Figure 33) and are subsequently associated with labels, aliases, and descriptions in various languages [261], allowing claims to be surfaced in these languages. Collaborative editing is not only permitted on the data level, but also on the schema level, allowing users to add or modify lightweight semantic axioms [391] – including sub-classes, sub-properties, inverse properties, etc. – as well as shapes [56]. Wikidata offers various access protocols [315] and has received broad adoption, being used by Wikipedia to generate infoboxes in certain domains [431], being supported by Google [382], and having been used as a data source for prominent applications such as Apple's Siri, amongst others [315].

*10.1.5 Other open cross-domain knowledge graphs.* A number of other cross-domain knowledge graphs have been developed down through the years. BabelNet [351] – in a similar fashion to YAGO – is based on unifying WordNet and Wikipedia, but with the integration of additional knowledge graphs such as Wikidata, and a focus on creating a knowledge graph of multilingual lexical forms (organized into multilingual synsets) by transforming lexicographic resources such as Wiktionary and OmegaWiki into knowledge graphs. Compared to other knowledge graphs, lexicalized knowledge graphs such as BabelNet bring together the encyclopedic information found in Wikipedia with the lexicographic information usually found in monolingual and bilingual dictionaries. The Cyc project [294] aims to encode common-sense knowledge in a machine-readable way, where over 900 person-years of effort [319] have, since 1986, gone into the creation of 2.2 million facts and rules. Though Cyc is proprietary, an open subset called OpenCyc has been published, where we refer to the comparison by Färber et al. [148] of DBpedia, Freebase, OpenCyc, and YAGO for further details. The Never Ending Language Learning (NELL) project [335] has, since 2010, extracted a graph of 120 million edges from the text of web pages using OIE methods (see Section 6). Each such open knowledge graph applies different combinations of the languages and techniques discussed in this paper over different sources with differing results.

*10.1.6 Domain-specific open knowledge graphs.* Open knowledge graphs have been published in a variety of specific domains. Schmachtenberg et al. [438] identify the most prominent domains in the context of Linked Data as follows: *media*, relating to news, television, radio, etc. (e.g., the BBC World Service Archive [406]); *government*, relating to the publication of data for transparency and development (e.g., by the U.S. [222] and U.K. [450] governments); *publications*, relating to academic literature in various disciplines (e.g., OpenCitations [385], SciGraph [246], Microsoft Academic Knowledge Graph [147]); *geographic*, relating to places and regions of interest (e.g., LinkedGeo-Data [472]); *life sciences*, relating to proteins, genes, drugs, diseases, etc. (e.g., Bio2RDF [79]); and *user-generated content*, relating to reviews, open source projects, etc. (e.g., Revyu [217]). Open knowledge graphs have also been published in other domains, including *cultural heritage* [245], *music* [407], *law* [337], *theology* [451], and even *tourism* [11, 263, 308, 540]. The envisaged applications for such knowledge graphs are as varied as the domains from which they emanate, but often relate to integration [79, 407], recommendation [308, 407], transparency [222, 450], archiving [245, 406], decentralisation [217], multilingual support [451], regulatory compliance [337], etc.

## 10.2 Enterprise Knowledge Graphs

A variety of companies have announced the creation of proprietary "enterprise knowledge graphs" with a variety of goals in mind, which include: improving search capabilities [83, 205, 280, 457, 459], providing user recommendations [83, 205], implementing conversational/personal agents [392], enhancing targeted advertising [214], empowering business analytics [214], connecting users [214, 365], extending multilingual support [214], facilitating research and discovery [34], assessing and mitigating risk [107, 495], tracking news events [326], and increasing transport automation [223], amongst (many) others. Though highly diverse, these enterprise knowledge graphs do follow some high-level trends, as reflected in the discussion by Noy et al. [365]: (1) data are typically integrated into the knowledge graph from a variety of both external and internal sources (often involving text); (2) the enterprise knowledge graph is often very large, with millions or even billions of nodes and edges, posing challenges in terms of scalability; (3) refinement of the initial knowledge graph – adding new links, consolidating duplicate entities, etc. – is important to improve quality; (4) techniques to keep the knowledge graph up-to-date with the domain are often crucial; (5) a mix of ontological and machine learning representations are often combined or used in different situations in order to draw conclusions from the enterprise knowledge graph; (6) the ontologies used tend to be lightweight, often simple taxonomies representing a hierarchy of classes or concepts.

We now discuss the main industries in which enterprise knowledge graphs have been deployed.

*10.2.1 Web search.* Web search engines have traditionally focused on matching a query string with sub-strings in web documents. The Google Knowledge Graph [365, 459] rather promoted a paradigm of "*things not strings*" – analogous to semantic search [199] – where the search engine would now try to identify the entities that a particular search may be expressing interest in. The knowledge graph itself describes these entities and how they interrelate. One of the main user-facing applications of the Google Knowledge Graph is the "Knowledge Panel", which presents a pane on the right-hand side of (some) search results describing the principal entity that the search appears to be seeking, including some images, attribute–value pairs, and a list of related entities that users also search for. The Google Knowledge Graph was key to popularising the modern usage of the phrase "knowledge graph" (see Appendix A). Other major search engines, such as Microsoft Bing[36] [457], would later announce knowledge graphs along similar lines.

*10.2.2 Commerce.* Enterprise knowledge graphs have also been announced by companies that are principally concerned with selling or renting goods and services. A prominent example of such a knowledge graph is that used by Amazon [127, 280], which describes the products on sale in their online marketplace. One of the main stated goals of this knowledge graph is to enable more advanced (semantic) search features for products, as well as to improve product recommendations to users of its online marketplace. Another knowledge graph for commerce was announced by eBay [392], which encodes product descriptions and shopping behaviour patterns, and is used to power conversational agents that aid users to find relevant products through a natural language interface. Airbnb [83] have also described a knowledge graph that encodes accommodation for rent, places, events, experiences, neighbourhoods, users, tags, etc., on top of which a taxonomic schema is defined. This knowledge graph is used to offer potential clients recommendations of attractions, events, and activities available in the neighbourhood of a particular home for rent. Uber [205] have similarly announced a knowledge graph focused on food and restaurants for their "Uber Eats" delivery service. The goals are again to offer semantic search features and recommendations to users who are uncertain precisely what kind of food they are looking for.

---

[36]Microsoft's Knowledge Graph was previously called "Satori" (meaning *understanding* in Japanese).

*10.2.3 Social networks.* Enterprise knowledge graphs have also emerged in the context of social networking services. Facebook [365] have gathered together a knowledge graph describing not only social data about users, but also the entities they are interested in, including celebrities, places, movies, music, etc., in order to connect people, understand their interests, and provide recommendations. LinkedIn [214] announced a knowledge graph containing users, jobs, skills, companies, places, schools, etc., on top of which a taxonomic schema is defined. The knowledge graph is used to provide multilingual translations of important concepts, to improve targeted advertising, to provide advanced features for job search and people search, and likewise to provide recommendations matching jobs to people (and vice versa). Another knowledge graph has been created by Pinterest [188], describing users and their interests, the latter being organised into a taxonomy. The main use-cases for the knowledge graph are to aid users to more easily find content of interest to them, as well as to enhance revenue through targetted advertisements.

*10.2.4 Finance.* The financial sector has also seen deployment of enterprise knowledge graphs. Amongst these, Bloomberg [326] has proposed a knowledge graph that powers financial data analytics, including sentiment analysis for companies based on current news reports and tweets, a question answering service, as well as detecting emerging events that may affect stock values. Thompson Reuters (Refinitiv) [495] have likewise announced a knowledge graph encoding "the financial ecosystem" of people, organisations, equity instruments, industry classifications, joint ventures and alliances, supply chains, etc., using a taxonomic schema to organise these entities. Some of the applications they mention for the knowledge graph include supply chain monitoring, risk assessment, and investment research. Knowledge graphs have also been explored in academic settings with Banca d'Italia [32], using rule-based reasoning to determine, for example, the percentage of ownership of a company by various stakeholders. Other companies exploring financial knowledge graphs include Accenture [368], Capital One [65], Wells Fargo [355], amongst others.

*10.2.5 Other industries.* Enterprises have also been actively developing knowledge graphs to enable novel applications in a variety of other industries, including: *health-care*, where IBM are exploring use-cases for drug discovery [365] and information extraction from package inserts [173], while AstraZeneca [34] are using a knowledge graph to advance genomics research and disease understanding; *transport*, where Bosch are exploring a knowledge graph of scenes and locations for driving automation [223]; *oil & gas*, where Maana [107] are using knowledge graphs to perform data integration for risk mitigation regarding oil wells and drilling; and more besides.

## 11 SUMMARY AND CONCLUSION

We have provided a comprehensive introduction to knowledge graphs, which have been receiving more and more attention in recent years. Under the definition of a knowledge graph as *a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities*, we have discussed models by which data can be structured as graphs; representations of schema, identity and context; techniques for leveraging deductive and inductive knowledge; methods for the creation, enrichment, quality assessment and refinement of knowledge graphs; principles and standards for publishing knowledge graphs; and finally, the adoption of knowledge graphs in the real world.

Knowledge graphs ultimately serve as a common substrate of knowledge within an organisation or community, enabling the representation, accumulation, curation, and dissemination of knowledge over time [365]. In this role, knowledge graphs have been applied in a wide variety of use-cases, ranging from commercial applications – involving semantic search, user recommendations, conversational agents, targeted advertising, transport automation, etc. – to open knowledge graphs made available for the public good. Some general trends in adoption include: (1) the use of knowledge

graphs to integrate and leverage data from diverse sources at large scale; (2) the combination of deductive formalisms (logical rules, ontologies, etc.) and inductive techniques (machine learning, analytics, etc.) to represent and accumulate knowledge; and (3) the bottom-up development of knowledge graphs, with extraction, enrichment, and refinement processes being intermittently or continuously applied to update, extend, and/or improve the knowledge graph over time.

Research on knowledge graphs should thus (ideally) become a confluence of techniques arising from different areas with the common objective of maximising the knowledge – and thus value – that can be distilled from diverse sources at large scale using a graph-based data abstraction. Pursuing this objective will benefit from expertise on graph databases, knowledge representation, logic, machine learning, graph algorithms and theory, ontology engineering, data quality, natural language processing, information extraction, privacy and security, and more besides. Given the availability of open knowledge graphs whose quality and adoption continue to improve, as well as the burgeoning adoption of enterprise knowledge graphs in various industries, future research on knowledge graphs has the potential to foster key advancements in broad aspects of society.

# REFERENCES

[1] Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux (Eds.). 2007. *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*. Lecture Notes in Computer Science, Vol. 4825. Springer.

[2] Serge Abiteboul. 1997. Querying Semi-Structured Data. In *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings (Lecture Notes in Computer Science)*, Foto N. Afrati and Phokion G. Kolaitis (Eds.), Vol. 1186. Springer, 1–18. https://doi.org/10.1007/3-540-62222-5_33

[3] Sushant Agarwal, Simon Steyskal, Franjo Antunovic, and Sabrina Kirrane. 2018. Legislative Compliance Assessment: Framework, Model and GDPR Instantiation. In *Privacy Technologies and Policy - 6th Annual Privacy Forum, APF 2018, Barcelona, Spain, June 13-14, 2018, Revised Selected Papers (Lecture Notes in Computer Science)*, Manel Medina, Andreas Mitrakas, Kai Rannenberg, Erich Schweighofer, and Nikolaos Tsouroulas (Eds.), Vol. 11079. Springer, 131–149.

[4] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*, Peter Buneman and Sushil Jajodia (Eds.). ACM Press, 207–216.

[5] Jinhyun Ahn, Dong-Hyuk Im, Jae-Hong Eom, Nansu Zong, and Hong-Gee Kim. 2015. G-Diff: A Grouping Algorithm for RDF Change Detection on MapReduce. In *Semantic Technology - 4th Joint International Conference, JIST 2014, Chiang Mai, Thailand, November 9-11, 2014. Revised Selected Papers (Lecture Notes in Computer Science)*, Thepchai Supnithi, Takahira Yamaguchi, Jeff Z. Pan, Vilas Wuwongse, and Marut Buranarach (Eds.), Vol. 8943. Springer, 230–235.

[6] Adnan Akhter, Axel-Cyrille Ngonga Ngomo, and Muhammad Saleem. 2018. An Empirical Evaluation of RDF Graph Partitioning Techniques. In *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW*

*2018, Nancy, France, November 12-16, 2018, Proceedings (Lecture Notes in Computer Science)*, Catherine Faron-Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint (Eds.), Vol. 11313. Springer, 3–18.

[7] Harith Alani, Lalana Kagal, Achille Fokoue, Paul T. Groth, , Josian Xavier Parreira, Lora Aroyo, Natasha Fridman Noy, Christopher A. Welty, and Krzysztof Janowicz (Eds.). 2013. *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II.* Lecture Notes in Computer Science, Vol. 8219. Springer.

[8] Harith Alani, Lalana Kagal, Achille Fokoue, Paul T. Groth, , Josian Xavier Parreira, Lora Aroyo, Natasha Fridman Noy, Christopher A. Welty, and Krzysztof Janowicz (Eds.). 2013. *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I.* Lecture Notes in Computer Science, Vol. 8218. Springer.

[9] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. 2009. Describing Linked Datasets. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009 (CEUR Workshop Proceedings)*, Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas (Eds.), Vol. 538. Sun SITE Central Europe (CEUR), 10. http://ceur-ws.org/Vol-538/ldow2009_paper20.pdf

[10] Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen (Eds.). 2008. *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico.* IEEE Computer Society.

[11] Ricardo Alonso Maturana, Elena Alvarado-Cortes, Susana López-Sola, María Ortega Martínez-Losa, and Pablo Hermoso-González. 2018. La Rioja Turismo: The Construction and Exploitation of a Queryable Tourism Knowledge Graph. In *Current Trends in Web Engineering - ICWE 2018 International Workshops, MATWEP, EnWot, KD-WEB, WEOD, TourismKG, Cáceres, Spain, June 5, 2018, Revised Selected Papers (Lecture Notes in Computer Science)*, Cesare Pautasso, Fernando Sánchez-Figueroa, Kari Systä, and Juan Manuel Murillo Rodriguez (Eds.), Vol. 11153. Springer, 213–220.

[12] Renzo Angles. 2018. The Property Graph Database Model. In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21–25, 2018 (CEUR Workshop Proceedings)*, Dan Olteanu and Barbara Poblete (Eds.), Vol. 2100. Sun SITE Central Europe (CEUR), 10. http://ceur-ws.org/Vol-2100/paper26.pdf

[13] Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter A. Boncz, George H. L. Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan F. Sequeda, Oskar van Rest, and Hannes Voigt. 2018. G-CORE: A Core for Future Graph Query Languages, See [112], 1421–1432.

[14] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of Modern Query Languages for Graph Databases. *ACM Computing Surveys* 50, 5 (2017), 68:1–68:40. https://doi.org/10.1145/3104031

[15] Renzo Angles and Claudio Gutiérrez. 2008. Survey of graph database models. *ACM Computing Surveys* 40, 1 (2008), 1:1–1:39. https://doi.org/10.1145/1322432.1322433

[16] Renzo Angles, Harsh Thakkar, and Dominik Tomaszuk. 2019. RDF and Property Graphs Interoperability: Status and Issues, See [237], 11. http://ceur-ws.org/Vol-2369/paper01.pdf

[17] Mikel Egaña Aranguren, Erick Antezana, Martin Kuiper, and Robert Stevens. 2008. Ontology Design Patterns for bio-ontologies: a case study on the Cell Cycle Ontology. *BMC Bioinformatics* 9, 5 (2008), S1. https://doi.org/10.1186/1471-2105-9-S5-S1

[18] Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and Juan Sequeda. 2012. *A Direct Mapping of Relational Data to RDF, W3C Recommendation 27 September 2012.* W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/

[19] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. 2009. The DL-Lite Family and Relations. *Journal of Artificial Intelligence Research* 36 (2009), 1–69.

[20] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data, See [1], 722–735.

[21] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. 2017. *An Introduction to Description Logic.* Cambridge University Press, Cambridge, United Kingdom.

[22] Nguyen Bach and Sameer Badaskar. 2007. *A Review of Relation Extraction.* Technical Report. Carnegie Mellon University.

[23] Ricardo Baeza-Yates. 2018. Bias on the Web. *Communications of the ACM* 61, 6 (2018), 54–61. https://doi.org/10.1145/3209581

[24] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference*, Christian Boitet and Pete Whitelock (Eds.). Morgan Kaufmann, 86–90.

[25] René Ronald Bakker. 1987. *Knowledge Graphs: Representation and Structuring of Scientific Knowledge.* Ph.D. Dissertation. University of Twente.

[26] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. Hypernetwork Knowledge Graph Embeddings. In *Artificial Neural Networks and Machine Learning - ICANN 2019 - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings - Workshop and Special Sessions (Lecture Notes in Computer Science)*, Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis (Eds.), Vol. 11731. Springer, 553–565.

[27] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). The Association for Computational Linguistics, 5184–5193. https://aclweb.org/anthology/volumes/D19-1/

[28] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, Manuela M. Veloso (Ed.). AAAI Press, 2670–2676.

[29] Pablo Barceló, Egor V. Kostylev, Mikael Monet, Jorge Peréz, Juan Reutter, and Juan Pablo Silva. 2020. The Logical Expressiveness of Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net, 20. https://openreview.net/forum?id=r1lZ7AEKvB

[30] Carlo Batini, Anisa Rula, Monica Scannapieco, and Gianluigi Viscusi. 2015. From Data Quality to Big Data Quality. *Journal of Database Management* 26, 1 (2015), 60–82. https://doi.org/10.4018/JDM.2015010103

[31] Carlo Batini and Monica Scannapieco. 2016. *Data and Information Quality - Dimensions, Principles and Techniques*. Springer.

[32] Luigi Bellomarini, Daniele Fakhoury, Georg Gottlob, and Emanuel Sallinger. 2019. Knowledge Graphs and Enterprise AI: The Promise of an Enabling Technology, See [248], 26–37.

[33] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. 2018. The Vadalog System: Datalog-based Reasoning for Knowledge Graphs. *Proceedings of the VLDB Endowment* 11, 9 (2018), 975–987.

[34] Claus Bendtsen and Slavé Petrovski. 2019. How data and AI are helping unlock the secrets of disease. AstraZeneca Blog. https://www.astrazeneca.com/what-science-can-do/labtalk-blog/uncategorized/how-data-and-ai-are-helping-unlock-the-secrets-of-disease.html.

[35] Yoshua Bengio and Yann LeCun (Eds.). 2013. *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. OpenReview.net. https://openreview.net/group?id=ICLR.cc/2013

[36] Michael K. Bergman. 2019. A Common Sense View of Knowledge Graphs. Adaptive Information, Adaptive Innovation, Adaptive Infrastructure Blog. http://www.mkbergman.com/2244/a-common-sense-view-of-knowledge-graphs/.

[37] Tim Berners-Lee. 2006. Linked Data. W3C Design Issues. https://www.w3.org/DesignIssues/LinkedData.html.

[38] Tim Berners-Lee and Dan Connolly. 2011. *Notation3 (N3): A readable RDF syntax, W3C Team Submission 28 March 2011*. W3C Team Submission. World Wide Web Consortium. https://www.w3.org/TeamSubmission/2011/SUBM-n3-20110328/

[39] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The Semantic Web. *Scientific American* 284, 5 (May 2001), 34–43.

[40] Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan (Eds.). 2009. *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*. Lecture Notes in Computer Science, Vol. 5823. Springer.

[41] Anant P. Bhardwaj, Souvik Bhattacherjee, Amit Chavan, Amol Deshpande, Aaron J. Elmore, Samuel Madden, and Aditya G. Parameswaran. 2015. DataHub: Collaborative Data Science & Dataset Version Management at Scale, See [88], 7. http://cidrdb.org/cidr2015/Papers/CIDR15_Paper18.pdf

[42] Daniel M. Bikel, Richard M. Schwartz, and Ralph M. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning* 34, 1-3 (1999), 211–231.

[43] Stefan Bischof, Stefan Decker, Thomas Krennwallner, Nuno Lopes, and Axel Polleres. 2012. Mapping between RDF and XML with XSPARQL. *Journal of Web Semantics* 1, 3 (2012), 147–185.

[44] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia-A crystallization point for the Web of Data. *Journal of Web Semantics* 7, 3 (2009), 154–165.

[45] Eva Blomqvist, Paolo Ciancarini, Francesco Poggi, and Fabio Vitali (Eds.). 2016. *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings*. Lecture Notes in Computer Science, Vol. 10024. Springer.

[46] Eva Blomqvist, Karl Hammar, and Valentina Presutti. 2016. Engineering Ontologies with Patterns – The eXtreme Design Methodology. In *Ontology Engineering with Ontology Design Patterns*, Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi, and Valentina Presutti (Eds.). Studies on the Semantic Web, Vol. 25. IOS Press.

[47] Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig (Eds.). 2017. *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I.*

Lecture Notes in Computer Science, Vol. 10249. Springer.

[48] Eva Blomqvist and Kurt Sandkuhl. 2005. Patterns in Ontology Engineering: Classification of Ontology Patterns. In *ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005*, Chin-Sheng Chen, Joaquim Filipe, Isabel Seruca, and José Cordeiro (Eds.), Vol. 3. 413–416.

[49] Eva Blomqvist, Azam Seil Sepour, and Valentina Presutti. 2012. Ontology Testing - Methodology and Tool. In *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings (Lecture Notes in Computer Science)*, Annette ten Teije, Johanna Völker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d'Aquin, Andriy Nikolov, Nathalie Aussenac-Gilles, and Nathalie Hernandez (Eds.), Vol. 7603. Springer, 216–226.

[50] Kurt Bollacker, Robert Cook, and Patrick Tufts. 2007. Freebase: A Shared Database of Structured General Human Knowledge. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*. AAAI Press, 1962–1963.

[51] Kurt Bollacker, Patrick Tufts, Tomi Pierce, and Robert Cook. 2007. A platform for scalable, collaborative, structured information integration. In *Intl. Workshop on Information Integration on the Web (IIWeb'07)*, Ullas Nambiar and Zaiqing Nie (Eds.). 6.

[52] Piero Bonatti, Sabrina Kirrane, Iliana Mineva Petrova, Luigi Sauro, and Eva Schlehahn. 2019. *The SPECIAL Usage Policy Language, V1.0*. Draft. Vienna University of Economics and Business. https://ai.wu.ac.at/policies/policylanguage/

[53] Piero Andrea Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti. 2018. Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371). *Dagstuhl Reports* 8, 9 (2018), 29–111.

[54] Piero A. Bonatti, Aidan Hogan, Axel Polleres, and Luigi Sauro. 2011. Robust and scalable Linked Data reasoning incorporating provenance and trust annotations. *Journal of Web Semantics* 9, 2 (2011), 165–201.

[55] Piero A. Bonatti and Sabrina Kirrane. 2019. Big Data and Analytics in the Age of the GDPR. In *2019 IEEE International Congress on Big Data, BigData Congress 2019, Milan, Italy, July 8-13, 2019*, Elisa Bertino, Carl K. Chang, Peter Chen, Ernesto Damiani, Michael Goul, and Katsunori Oyama (Eds.). IEEE Computer Society, 7–16.

[56] Iovka Boneva, Jérémie Dusart, Daniel Fernández-Álvarez, and José Emilio Labra Gayo. 2019. Shape Designer for ShEx and SHACL constraints, See [480], 269–272. http://ceur-ws.org/Vol-2456

[57] Iovka Boneva, Jose Emilio Labra Gayo, and Eric G. Prud'hommeaux. 2017. Semantics and Validation of Shapes Schemas for RDF, See [108], 104–120.

[58] Angela Bonifati, Wim Martens, and Thomas Timm. 2017. An Analytical Study of Large SPARQL Query Logs. *Proceedings of the VLDB Endowment* 11, 2 (2017), 149–161.

[59] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data, See [76], 2787–2795. http://papers.nips.cc/book/advances-in-neural-information-processing-systems-26-2013

[60] Gerlof Bouma. 2009. Normalized (Pointwise) Mutual Information in Collocation Extraction. In *Von der Form zur Bedeutung: Texte automatisch verarbeiten - From Form to Meaning: Processing Texts Automatically, Proceedings of the Biennial GSCL Conference 2009*, Christian Chiarcos, Richard Eckart de Castilho, and Manfred Stede (Eds.). Gunter Narr Verlag, 31–40.

[61] Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao (Eds.). 2016. *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*. ACM Press.

[62] Ronald J. Brachman. 1977. *A structural paradigm for representing knowledge*. Ph.D. Dissertation. Harvard University.

[63] Ronald J. Brachman and Hector J. Levesque. 1986. The Knowledge Level of a KBMS. In *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies, Book resulting from the Islamorada Workshop 1985 (Islamorada, FL, USA) (Topics in Information Systems)*, Michael L. Brodie and John Mylopoulos (Eds.). Springer, 9–12.

[64] Ronald J. Brachman and James G. Schmolze. 1985. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9, 2 (1985), 171–216.

[65] Patricia Branum and Bethany Sehon. 2019. Knowledge Graph Pilot Improves Data Quality While Providing a Customer 360 View. In *Knowledge Graph Conference*. (Invited talk).

[66] Dan Brickley and R. V. Guha. 2014. *RDF Schema 1.1, W3C Recommendation 25 February 2014*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/

[67] Carla E. Brodley and Peter Stone (Eds.). 2014. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. AAAI Press.

[68] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). OpenReview.net. https://openreview.net/group?id=ICLR.cc/2014

[69] Bruce G. Buchanan and Edward A. Feigenbaum. 1978. Dendral and Meta-Dendral: Their Applications Dimension. *Artificial Intelligence* 11, 1-2 (1978), 5–24.

[70] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. 2016. DL-Learner – A framework for inductive learning on the Semantic Web. *Journal of Web Semantics* 39 (2016), 15–24. https://doi.org/10.1016/j.websem.2016.06.001

[71] Carlos Buil Aranda, Marcelo Arenas, Óscar Corcho, and Axel Polleres. 2013. Federating queries in SPARQL 1.1: Syntax, semantics and evaluation. *Journal of Web Semantics* 18, 1 (2013), 1–17.

[72] Carlos Buil-Aranda, Aidan Hogan, Jürgen Umbrich, and Pierre-Yves Vandenbussche. 2013. SPARQL Web-Querying Infrastructure: Ready for Action?, See [7], 277–293.

[73] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini (Eds.). 2005. *Ontology learning from text: methods, evaluation and applications*. Frontiers in Artificial Intelligence and Applications, Vol. 123. IOS Press.

[74] Razvan C. Bunescu and Raymond J. Mooney. 2005. Subsequence Kernels for Relation Extraction. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 171–178. http://papers.nips.cc/book/advances-in-neural-information-processing-systems-18-2005

[75] Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to Extract Relations from the Web using Minimal Supervision. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*, John A. Carroll, Antal van den Bosch, and Annie Zaenen (Eds.). The Association for Computational Linguistics, 576–583.

[76] Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 2013. *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. http://papers.nips.cc/book/advances-in-neural-information-processing-systems-26-2013

[77] Elena Cabrio, Alessio Palmero Aprosio, and Serena Villata. 2014. These Are Your Rights - A Natural Language Processing Approach to Automated RDF Licenses Generation, See [396], 255–269.

[78] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1, 1 (2008), 538–549.

[79] Alison Callahan, Jose Cruz-Toledo, Peter Ansell, and Michel Dumontier. 2013. Bio2RDF Release 2: Improved Coverage, Interoperability and Provenance of Life Science Linked Data, See [90], 200–212.

[80] Nicholas J. Car, Paul J. Box, and Ashley Sommer. 2019. The Location Index: A Semantic Web Spatial Data Infrastructure, See [226], 543–557.

[81] Šejla Čebirić, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou, and Mussab Zneika. 2019. Summarizing semantic graphs: a survey. *The Very Large Data Base Journal* 28, 3 (2019), 295–327.

[82] Stefano Ceri, Georg Gottlob, and Letizia Tanca. 1989. What you Always Wanted to Know About Datalog (And Never Dared to Ask). *IEEE Transactions on Knowledge and Data Engineering* 1, 1 (1989), 146–166.

[83] Spencer Chang. 2018. Scaling Knowledge Access and Retrieval at Airbnb. AirBnB Medium Blog. https://medium.com/airbnb-engineering/scaling-knowledge-access-and-retrieval-at-airbnb-665b6ba21e95.

[84] Laura Chiticariu, Marina Danilevsky, Yunyao Li, Frederick Reiss, and Huaiyu Zhu. 2018. SystemT: Declarative Text Understanding for Enterprise. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers)*, Srinivas Bangalore, Jennifer Chu-Carroll, and Yunyao Li (Eds.). The Association for Computational Linguistics, 76–83.

[85] Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. The Association for Computational Linguistics, 827–832. https://www.aclweb.org/anthology/volumes/D13-1/

[86] Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel (Eds.). 2014. *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*. ACM Press.

[87] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PLOS One* 10, 6 (2015), e0128193.

[88] CIDR 2015. *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org.

[89] Philipp Cimiano. 2006. Ontology Learning from Text. In *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer, Chapter 3, 19–34.

[90] Philipp Cimiano, Óscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph (Eds.). 2013. *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*.

Lecture Notes in Computer Science, Vol. 7882. Springer.

[91] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. 2017. Biased Graph Walks for RDF Graph Embeddings. In *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017*, Rajendra Akerkar, Alfredo Cuzzocrea, Jannong Cao, and Mohand-Said Hacid (Eds.). ACM Press, Article 21, 12 pages.

[92] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. 2017. Global RDF Vector Space Embeddings, See [108], 190–207.

[93] Diego Collarana, Mikhail Galkin, Christoph Lange, Irlán Grangel-González, Maria-Esther Vidal, and Sören Auer. 2016. FuhSen: A Federated Hybrid Search Engine for Building a Knowledge Graph On-Demand (Short Paper), See [118], 752–761.

[94] Michael Collins and Yoram Singer. 1999. Unsupervised Models for Named Entity Classification. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP 1999, College Park, MD, USA, June 21-22, 1999*. The Association for Computational Linguistics, 11. https://www.aclweb.org/anthology/W99-0613/

[95] Dan Connolly. 2007. *Gleaning Resource Descriptions from Dialects of Languages (GRDDL), W3C Recommendation 11 September 2007*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2007/REC-grddl-20070911/

[96] Mariano P. Consens and Alberto O. Mendelzon. 1990. GraphLog: a Visual Formalism for Real Life Recursion. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2-4, 1990, Nashville, Tennessee, USA*, Daniel J. Rosenkrantz and Yehoshua Sagiv (Eds.). ACM Press, 404–416.

[97] Olivier Corby and Catherine Faron-Zucker. 2010. The KGRAM Abstract Machine for Knowledge Graph Querying. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2010, Toronto, Canada, August 31 - September 3, 2010, Main Conference Proceedings*, Jimmy Xiangji Huang, Irwin King, Vijay V. Raghavan, and Stefan Rueger (Eds.). IEEE Computer Society, 338–341.

[98] Francesco Corcoglioniti, Marco Rospocher, and Alessio Palmero Aprosio. 2016. Frame-Based Ontology Population with PIKES. *IEEE Transactions on Knowledge and Data Engineering* 28, 12 (2016), 3261–3275.

[99] Julien Corman, Fernando Florenzano, Juan L. Reutter, and Ognjen Savkovic. 2019. Validating SHACL Constraints over a SPARQL Endpoint, See [178], 145–163. https://doi.org/10.1007/978-3-030-30793-6_9

[100] Julien Corman, Juan L. Reutter, and Ognjen Savković. 2018. Semantics and Validation of Recursive SHACL, See [514], 318–336.

[101] Luca Costabello, Serena Villata, Nicolas Delaforge, and Fabien Gandon. 2012. Linked Data Access Goes Mobile: Context-Aware Authorization for Graph Stores. In *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012 (CEUR Workshop Proceedings)*, Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas (Eds.), Vol. 937. Sun SITE Central Europe (CEUR), 8. http://ceur-ws.org/Vol-937/ldow2012-paper-05.pdf

[102] Kino Coursey and Rada Mihalcea. 2009. Topic Identification Using Wikipedia Graph Centrality. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA, Short Papers*. The Association for Computational Linguistics, 117–120.

[103] Simon Cox, Chris Little, Jerry R. Hobbs, and Feng Pan. 2017. *Time Ontology in OWL, W3C Recommendation 19 October 2017*. W3C Recommendation / OGC 16-071r2. World Wide Web Consortium and Open Geospatial Consortium. https://www.w3.org/TR/2017/REC-owl-time-20171019/

[104] Eric Crestan and Patrick Pantel. 2011. Web-scale table census and classification. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*, Irwin King, Wolfgang Nejdl, and Hang Li (Eds.). ACM Press, 545–554.

[105] Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist (Eds.). 2012. *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*. Lecture Notes in Computer Science, Vol. 7649. Springer.

[106] Richard Cyganiak, David Wood, and Markus Lanthaler. 2014. *RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[107] Jeff Dalgliesh. 2016. How the Enterprise Knowledge Graph Connects Oil and Gas Data Silos. Maana Blog. https://www.maana.io/blog/enterprise-knowledge-graph-connects-oil-gas-data-silos/.

[108] Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin (Eds.). 2017. *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*. Lecture Notes in Computer Science, Vol. 10587. Springer.

[109] Claudia d'Amato, Steffen Staab, Andrea G. B. Tettamanzi, Duc Minh Tran, and Fabien L. Gandon. 2016. Ontology enrichment by discovering multi-relational association rules from ontological knowledge bases. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, Sascha Ossowski (Ed.). ACM Press, 333–338.

[110] Claudia d'Amato, Andrea G. B. Tettamanzi, and Minh Duc Tran. 2016. Evolutionary Discovery of Multi-relational Association Rules from Ontological Knowledge Bases, See [45], 113–128.

[111] Fariz Darari, Werner Nutt, Giuseppe Pirrò, and Simon Razniewski. 2018. Completeness Management for RDF Data Sources. *ACM Transactions on the Web* 12, 3 (2018), 18:1–18:53.

[112] Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). 2018. *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. ACM Press.

[113] Souripriya Das, Seema Sundara, and Richard Cyganiak. 2012. *R2RML: RDB to RDF Mapping Language, W3C Recommendation 27 September 2012*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2012/REC-r2rml-20120927/

[114] Ankur Dave, Alekh Jindal, Li Erran Li, Reynold Xin, Joseph Gonzalez, and Matei Zaharia. 2016. GraphFrames: an integrated API for mixing graph and relational queries. In *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, Redwood Shores, CA, USA, June 24 - 24, 2016*, Peter A. Boncz and Josep-Lluís Larriba-Pey (Eds.). ACM Press, 2.

[115] Gerard de Melo. 2015. Lexvo.org: Language-Related Information for the Linguistic Linked Data Cloud. *Semantic Web Journal* 6, 4 (7 Aug. 2015), 393–400.

[116] Luc De Raedt, Bart Vandersmissen, Marc Denecker, and Maurice Bruynooghe. 1990. A hybrid approach to learning and its knowledge representation. In *Proceedings of the third COGNITIVA symposium on At the crossroads of artificial intelligence, cognitive science, and neuroscience*. Elsevier, 409–416.

[117] Marina De Vos, Sabrina Kirrane, Julian Padget, and Ken Satoh. 2019. ODRL policy modelling and compliance checking. In *Rules and Reasoning - Third International Joint Conference, RuleML+RR 2019, Bolzano, Italy, September 16-19, 2019, Proceedings (Lecture Notes in Computer Science)*, Paul Fodor, Marco Montali, Diego Calvanese, and Dumitru Roman (Eds.), Vol. 11784. Springer, 36–51.

[118] Christophe Debruyne, Hervé Panetto, Robert Meersman, Tharam S. Dillon, eva Kühn, Declan O'Sullivan, and Claudio Agostino Ardagna (Eds.). 2016. *On the Move to Meaningful Internet Systems: OTM 2016 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings*. Lecture Notes in Computer Science, Vol. 10033. Springer.

[119] Remy Delanaux, Angela Bonifati, Marie-Christine Rousset, and Romuald Thion. 2018. Query-Based Linked Data Anonymization, See [514], 530–546.

[120] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted Rule Injection for Relation Embeddings, See [479], 1389–1399.

[121] Dong Deng, Yu Jiang, Guoliang Li, Jian Li, and Cong Yu. 2013. Scalable Column Concept Determination for Web Tables Using Large Knowledge Bases. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1606–1617.

[122] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings, See [325], 1811–1818.

[123] Deepika Devarajan. 2017. Happy Birthday Watson Discovery. IBM Cloud Blog. https://www.ibm.com/blogs/bluemix/2017/12/happy-birthday-watson-discovery/.

[124] Rose Dieng, Alain Giboin, Paul-André Tourtier, and Olivier Corby. 1992. Knowledge Acquisition for Explainable, Multi-Expert, Knowledge-Based Design Systems. In *Current Developments in Knowledge Acquisition - EKAW'92, 6th European Knowledge Acquisition Workshop, Heidelberg and Kaiserslautern, Germany, May 18-22, 1992 (Lecture Notes in Computer Science)*, Thomas Wetter, Klaus-Dieter Althoff, John H. Boose, Brian R. Gaines, and Marc Linster (Eds.), Vol. 599. Springer, 298–317.

[125] Renata Queiroz Dividino, Sergej Sizov, Steffen Staab, and Bernhard Schueler. 2009. Querying for provenance, trust, uncertainty and other meta knowledge in RDF. *Journal of Web Semantics* 7, 3 (2009), 204–219. https://doi.org/10.1016/j.websem.2009.07.004

[126] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM Press, 601–610.

[127] Xin Luna Dong. 2019. Building a Broad Knowledge Graph for Products, See [248], 25–25.

[128] Uwe Draisbach and Felix Naumann. 2011. A generalization of blocking and windowing algorithms for duplicate detection. In *2011 International Conference on Data and Knowledge Engineering, ICDKE 2011, Milano, Italy, September 6, 2011*, Ji Zhang and Giovanni Livraga (Eds.). IEEE Computer Society, 18–24.

[129] Martin Dürst and Michel Suignard. 2005. *Internationalized Resource Identifiers (IRIs)*. RFC 3987. Internet Engineering Task Force. http://www.ietf.org/rfc/rfc3987.txt

[130] Arnab Dutta, Christian Meilicke, and Heiner Stuckenschmidt. 2014. Semantifying Triples from Open Information Extraction Systems. In *STAIRS 2014 - Proceedings of the 7th European Starting AI Researcher Symposium, Prague, Czech Republic, August 18-22, 2014 (Frontiers in Artificial Intelligence and Applications)*, Ulle Endriss and João Leite (Eds.), Vol. 264. IOS Press, 111–120.

[131] Arnab Dutta, Christian Meilicke, and Heiner Stuckenschmidt. 2015. Enriching Structured Knowledge with Open Information, See [168], 267–277.

[132] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II (Lecture Notes in Computer Science)*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.), Vol. 4052. Springer, 1–12.

[133] Eugene Dynkin. 1965. *Markov processes*. Springer.

[134] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. 2015. Building the Dresden Web Table Corpus: A Classification Approach. In *2nd IEEE/ACM International Symposium on Big Data Computing, BDC 2015, Limassol, Cyprus, December 7-10, 2015*, Ioan Raicu, Omer F. Rana, and Rajkumar Buyya (Eds.). IEEE Computer Society, 41–50.

[135] Mikel Egaña, Alan Rector, Robert Stevens, and Erick Antezana. 2008. Applying Ontology Design Patterns in Bio-ontologies. In *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings (Lecture Notes in Computer Science)*, Aldo Gangemi and Jérôme Euzenat (Eds.), Vol. 5268. Springer, 7–16.

[136] Lisa Ehrlinger and Wolfram Wöß. 2016. Towards a Definition of Knowledge Graphs. In *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016 (CEUR Workshop Proceedings)*, Michael Martin, Martí Cuquet, and Erwin Folmer (Eds.), Vol. 1695. Sun SITE Central Europe (CEUR), 4. http://ceur-ws.org/Vol-1695/paper4.pdf

[137] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. 2009. Language-model-based ranking for queries on RDF-graphs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin (Eds.). ACM Press, 977–986.

[138] Basil Ell, Andreas Harth, and Elena Simperl. 2014. SPARQL Query Verbalization for Explaining Semantic Search Engine Queries, See [396], 426–441.

[139] Orri Erling. 2012. Virtuoso, a Hybrid RDBMS/Graph Column Store. *IEEE Data Engineering Bulletin* 35, 1 (2012), 3–8.

[140] Ivan Ermilov and Axel-Cyrille Ngonga Ngomo. 2016. TAIPAN: Automatic Property Mapping for Tabular Data, See [45], 163–179.

[141] Diego Esteves, Anisa Rula, Aniketh Janardhan Reddy, and Jens Lehmann. 2018. Toward Veracity Assessment in RDF Knowledge Bases: An Exploratory Analysis. *Journal of Data and Information Quality* 9, 3 (2018), 16:1–16:26. https://doi.org/10.1145/3177873

[142] Ernesto Estrada. 2011. *The Structure of Complex Networks: Theory and Applications*. Oxford University Press, Inc.

[143] Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills (Eds.). ACM Press, 100–110.

[144] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open Information Extraction: The Second Generation, See [518], 3–10.

[145] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. The Association for Computational Linguistics, 1535–1545. https://www.aclweb.org/anthology/volumes/D11-1/

[146] Wenfei Fan, Xin Wang, and Yinghui Wu. 2013. Diversified Top-$k$ Graph Pattern Matching. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1510–1521.

[147] Michael Färber. 2019. The Microsoft Academic Knowledge Graph: A Linked Data Source with 8 Billion Triples of Scholarly Data, See [179], 113–129.

[148] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. 2018. Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web Journal* 9, 1 (2018), 77–129.

[149] Javier D. Fernández, Sabrina Kirrane, Axel Polleres, and Simon Steyskal. 2017. Self-Enforcing Access Control for Encrypted RDF, See [47], 607–622.

[150] Javier D. Fernández, Miguel A. Martínez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. 2013. Binary RDF representation for publication and exchange (HDT). *Journal of Web Semantics* 19 (2013), 22–41.

[151] Mariano Fernández, Asuncón Gómez-Pérez, and Natalia Juristo. 1997. METHONTOLOGY: from Ontological Art towards Ontological Engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*.

[152] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. 2014. Web data extraction, applications and techniques: A survey. *Knowledge-based Systems* 70 (2014), 301–323.

[153] Charles J. Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences* 280, 1 (1976), 20–32.

[154] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling, See [276], 363–370.

[155] Sergio Flesca, Giuseppe Manco, Elio Masciari, Eugenio Rende, and Andrea Tagarelli. 2004. Web wrapper induction: a brief survey. *AI Communications* 17, 2 (2004), 57–61.

[156] Giorgos Flouris, Irini Fundulaki, Maria Michou, and Grigoris Antoniou. 2010. Controlling Access to RDF Graphs. In *Future Internet - FIS 2010 - Third Future Internet Symposium, Berlin, Germany, September 20-22, 2010. Proceedings (Lecture Notes in Computer Science)*, Arne-Jørgen Berre, Asunción Gómez-Pérez, Kurt Tutschku, and Dieter Fensel (Eds.), Vol. 6369. Springer, 107–117.

[157] Association for Ontology Design & Patterns. [n. d.]. Ontology Design Patterns . org (ODP). http://ontologydesignpatterns.org/wiki/Main_Page

[158] Charles Forgy. 1982. Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. *Artificial Intelligence* 19, 1 (1982), 17–37.

[159] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An Evolving Query Language for Property Graphs, See [112], 1433–1445.

[160] Gottlob Frege. 1879. *Begriffsschrift*. Halle.

[161] Christian Fürber and Martin Hepp. 2011. SWIQA - a semantic web information quality assessment framework. In *19th European Conference on Information Systems, ECIS 2011, Helsinki, Finland, June 9-11, 2011*, Virpi Kristiina Tuunainen, Matti Rossi, and Joe Nandhakumar (Eds.). 76. http://aisel.aisnet.org/ecis2011/

[162] Mohamed H. Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2016. Exception-Enriched Rule Learning from Knowledge Graphs, See [194], 234–251. https://doi.org/10.1007/978-3-319-46523-4_15

[163] Luis Galárraga, Chistina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The Very Large Data Base Journal* 24, 6 (2015), 707–730.

[164] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases, See [444], 413–422.

[165] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. 2010. Corroborating Information from Disagreeing Views. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu (Eds.). ACM Press, 131–140.

[166] Fabien Gandon and Guus Schreiber. 2014. *RDF 1.1 XML Syntax, W3C Recommendation 25 February 2014*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/

[167] Aldo Gangemi. 2005. Ontology design patterns for semantic web content, See [182], 262–276.

[168] Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). 2015. *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. ACM Press.

[169] Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio, and Misael Mongiovì. 2017. Semantic Web Machine Reading with FRED. *Semantic Web Journal* 8, 6 (2017), 873–893.

[170] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, José M. Alonso, Alberto Bugarín, and Ehud Reiter (Eds.). The Association for Computational Linguistics, 124–133.

[171] Michael Gelfond and Vladimir Lifschitz. 1988. The Stable Model Semantics for Logic Programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes)*, Robert A. Kowalski and Kenneth A. Bowen (Eds.). The MIT Press, 1070–1080.

[172] James Geller, Soon Ae Chun, and Yoo Jung An. 2008. Toward the Semantic Deep Web. *IEEE Computer* 41, 9 (2008), 95–97.

[173] Anna Lisa Gentile, Daniel Gruhl, Petar Ristoski, and Steve Welch. 2019. Personalized Knowledge Graphs for the Pharmaceutical Domain, See [179], 400–417.

[174] Anna Lisa Gentile, Ziqi Zhang, and Fabio Ciravegna. 2014. Self Training Wrapper Induction with Linked Data. In *Text, Speech and Dialogue - 17th International Conference, TSD 2014, Brno, Czech Republic, September 8-12, 2014. Proceedings*

(*Lecture Notes in Computer Science*), Petr Sojka, Ales Horák, Ivan Kopecek, and Karel Pala (Eds.), Vol. 8655. Springer, 285–292.

[175] Daniel Gerber, Diego Esteves, Jens Lehmann, Lorenz Bühmann, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, and René Speck. 2015. DeFacto–temporal and multilingual deep fact validation. *Journal of Web Semantics* 35 (2015), 85–101.

[176] Sabrina Gerbracht. 2008. Possibilities to Encrypt an RDF-Graph. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*. IEEE Computer Society.

[177] Lise Getoor and Ben Taskar (Eds.). 2007. *Introduction to Statistical Relational Learning*. The MIT Press.

[178] Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon (Eds.). 2019. *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*. Lecture Notes in Computer Science, Vol. 11778. Springer.

[179] Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon (Eds.). 2019. *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. Lecture Notes in Computer Science, Vol. 11779. Springer.

[180] Mark Giereth. 2005. On Partial Encryption of RDF-Graphs, See [182], 308–322.

[181] Yolanda Gil, Simon Miles, Khalid Belhajjame, Daniel Garijo, Graham Klyne, Paolo Missier, Stian Soiland-Reyes, and Stephan Zednik. 2013. *PROV Model Primer, W3C Working Group Note 30 April 2013*. W3C Working Group Note. World Wide Web Consortium. https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/

[182] Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen (Eds.). 2005. *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*. Lecture Notes in Computer Science, Vol. 3729. Springer.

[183] José M. Giménez-García, Antoine Zimmermann, and Pierre Maret. 2017. NdFluents: An Ontology for Annotated Statements with Inference Preservation, See [47], 638–654.

[184] Birte Glimm. 2011. Using SPARQL with RDFS and OWL Entailment. In *Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures (Lecture Notes in Computer Science)*, Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter F. Patel-Schneider (Eds.), Vol. 6848. Springer, 137–201.

[185] Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. 2013. A Semantic Matching Energy Function for Learning with Multi-relational Data, See [35], 4. http://arxiv.org/abs/1301.3485

[186] Patrick Golden and Ryan B. Shaw. 2016. Nanopublication beyond the sciences: the PeriodO period gazetteer. *PeerJ Computer Science* 2 (2016), e44. https://doi.org/10.7717/peerj-cs.44

[187] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. 2006. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer.

[188] Rafael S. Gonçalves, Matthew Horridge, Rui Li, Yu Liu, Mark A. Musen, Csongor I. Nyulas, Evelyn Obamos, Dhananjay Shrouty, and David Temple. 2019. Use of OWL and Semantic Web Technologies at Pinterest, See [179], 418–435.

[189] Larry González and Aidan Hogan. 2018. Modelling Dynamics in Semantic Web Knowledge Graphs with Formal Concept Analysis. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM Press, 1175–1184.

[190] Simon Gottschalk and Elena Demidova. 2018. EventKG: A Multilingual Event-Centric Temporal Knowledge Graph. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings (Lecture Notes in Computer Science)*, Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Troncy Raphaël, Laura Hollink, Anna Tordai, and Mehwish Alam (Eds.), Vol. 10843. Springer, 272–287.

[191] Guido Governatori, Ho-Pun Lam, Antonino Rotolo, Serena Villata, and Fabien Gandon. 2013. Heuristics for Licenses Composition. In *Legal Knowledge and Information Systems - JURIX 2013: The Twenty-Sixth Annual Conference, December 11-13, 2013, University of Bologna, Italy (Frontiers in Artificial Intelligence and Applications)*, Kevin D. Ashley (Ed.), Vol. 259. IOS Press, 77–86.

[192] Ralph Grishman. 2012. *Information Extraction: Capabilities and Challenges*. Technical Report. NYU Dept. CS. Notes prepared for the 2012 International Winter School in Language and Speech Technologies.

[193] Paul T. Groth, Antonis Loizou, Alasdair J. G. Gray, Carole A. Goble, Lee Harland, and Steve Pettifer. 2014. API-centric Linked Data integration: The Open PHACTS Discovery Platform case study. *Journal of Web Semantics* 29 (2014), 12–18.

[194] Paul T. Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil (Eds.). 2016. *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*. Lecture Notes in Computer Science, Vol. 9981. Springer.

[195] Michael Grüninger and Mark S. Fox. 1995. Methodology for the Design and Evaluation of Ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal*.

[196] Michael Grüninger and Mark S. Fox. 1995. The role of competency questions in enterprise engineering. In *Benchmarking—Theory and practice*, Asbjorn Rolstadas (Ed.). Springer, 22–31.

[197] Nicola Guarino, Daniel Oberle, and Steffen Staab. 2009. What Is an Ontology? See [471], 1–17. https://doi.org/10.1007/978-3-540-92673-3

[198] Ramanathan V. Guha, Rob McCool, and Richard Fikes. 2004. Contexts for the Semantic Web. In *The Semantic Web - ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings (Lecture Notes in Computer Science)*, Frank van Harmelen, Sheila McIlraith, and Dimitri Plexousakis (Eds.), Vol. 3298. Springer, 32–46.

[199] Ramanathan V. Guha, Rob McCool, and Eric Miller. 2003. Semantic search. In *Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003*, Gusztáv Hencsey, Bebo White, Yih-Farn Robin Chen, László Kovács, and Steve Lawrence (Eds.). ACM Press, 700–709.

[200] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly Embedding Knowledge Graphs and Logical Rules, See [479], 192–202.

[201] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge Graph Embedding With Iterative Guidance From Soft Rules, See [325], 4816–4823. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16369

[202] Sonal Gupta and Christopher D. Manning. 2014. Improved Pattern Learning for Bootstrapped Entity Extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, Roser Morante and Wen-tau Yih (Eds.). The Association for Computational Linguistics, 98–108.

[203] Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. 2007. Introducing Time into RDF. *IEEE Transactions on Knowledge and Data Engineering* 19, 2 (2007), 207–218.

[204] Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 2017. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach, CA, USA*. http://papers.nips.cc/book/advances-in-neural-information-processing-systems-30-2017

[205] Ferras Hamad, Isaac Liu, and Xian Xing Zhang. 2018. Food Discovery with Uber Eats: Building a Query Understanding Engine. Uber Engineering Blog. https://eng.uber.com/uber-eats-query-understanding/.

[206] Tony Hammond, Michele Pasin, and Evangelos Theodoridis. 2017. Data integration and disintegration: Managing Springer Nature SciGraph with SHACL and OWL. In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017 (CEUR Workshop Proceedings)*, Nadeschda Nikitina, Dezhao Song, Achille Fokoue, and Peter Haase (Eds.), Vol. 1963. Sun SITE Central Europe (CEUR), 2. http://ceur-ws.org/Vol-1963/paper493.pdf

[207] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. 2013. *SPARQL 1.1 Query Language, W3C Recommendation 21 March 2013*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/

[208] Olaf Hartig. 2017. Foundations of RDF* and SPARQL* – An Alternative Approach to Statement-Level Metadata in RDF. In *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017 (CEUR Workshop Proceedings)*, Juan L. Reutter and Divesh Srivastava (Eds.), Vol. 1912. Sun SITE Central Europe (CEUR), 11. http://ceur-ws.org/Vol-1912/paper12.pdf

[209] Olaf Hartig, Christian Bizer, and Johann Christoph Freytag. 2009. Executing SPARQL Queries over the Web of Linked Data, See [40], 293–309.

[210] Olaf Hartig and Carlos Buil Aranda. 2016. Bindings-Restricted Triple Pattern Fragments, See [118], 762–779.

[211] Olaf Hartig, Ian Letter, and Jorge Pérez. 2017. A Formal Framework for Comparing Linked Data Fragments, See [108], 364–382.

[212] Olaf Hartig and Bryan Thompson. 2014. Foundations of an Alternative Approach to Reification in RDF. *CoRR* abs/1406.3399 (2014), 14. arXiv:1406.3399 http://arxiv.org/abs/1406.3399

[213] Patrick J. Hayes and Peter F. Patel-Schneider. 2014. *RDF 1.1 Semantics, W3C Recommendation 25 February 2014*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/

[214] Qi He, Bee-Chung Chen, and Deepak Agarwal. 2016. Building The LinkedIn Knowledge Graph. LinkedIn Blog. https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph.

[215] Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*. 539–545. https://www.aclweb.org/anthology/volumes/C92-1/

[216] Tom Heath and Christian Bizer. 2011. *Linked Data: Evolving the Web into a Global Data Space (1st Edition)*. Synthesis Lectures on the Semantic Web: Theory and Technology, Vol. 1. Morgan & Claypool. 136 pages.

[217] Tom Heath and Enrico Motta. 2008. Revyu: Linking reviews and ratings into the Web of Data. *Journal of Web Semantics* 6, 4 (2008), 266–273.

[218] Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism Detection in Wikidata. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi (Eds.). ACM Press, 327–336.

[219] Benjamin Heitmann, Felix Hermsen, and Stefan Decker. 2017. k-RDF-Neighbourhood Anonymity: Combining Structural and Attribute-based Anonymisation for Linked Data. In *Proceedings of the 5th Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn2017) co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 22, 2017 (CEUR Workshop Proceedings)*, Christopher Brewster, Michelle Cheatham, Mathieu d'Aquin, Stefan Decker, and Sabrina Kirrane (Eds.), Vol. 1951. Sun SITE Central Europe (CEUR), 16. http://ceur-ws.org/Vol-1951/PrivOn2017_paper_3.pdf

[220] Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. 2013. Integrating NLP Using Linked Data, See [7], 98–113.

[221] Remko Helms and Kees Buijsrogge. 2005. Knowledge Network Analysis: A Technique to Analyze Knowledge Management Bottlenecks in Organizations. In *16th International Workshop on Database and Expert Systems Applications (DEXA 2005), 22-26 August 2005, Copenhagen, Denmark*. IEEE Computer Society, 410–414. https://ieeexplore.ieee.org/xpl/conhome/10080/proceeding

[222] James A. Hendler, Jeanne Holm, Chris Musialek, and George Thomas. 2012. US Government Linked Open Data: Semantic.data.gov. *IEEE Intelligent Systems* 27, 3 (2012), 25–31.

[223] Cory Henson, Stefan Schmid, Anh Tuan Tran, and Antonios Karatzoglou. 2019. Using a Knowledge Graph of Scenes to Enable Search of Autonomous Driving Data, See [480], 313–314. http://ceur-ws.org/Vol-2456

[224] Daniel Hernández, Aidan Hogan, and Markus Krötzsch. 2015. Reifying RDF: What Works Well With Wikidata?. In *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015 (CEUR Workshop Proceedings)*, Thorsten Liebig and Achille Fokoue (Eds.), Vol. 1457. Sun SITE Central Europe (CEUR), 32–47. http://ceur-ws.org/Vol-1457/SSWS2015_paper3.pdf

[225] Frank L. Hitchcock. 1927. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics* 6, 1–4 (1927), 164–189.

[226] Pascal Hitzler, Miriam Fernández, Krzysztof Janowicz, Amrapali Zaveri, Alasdair J. G. Gray, Vanessa López, Armin Haller, and Karl Hammar (Eds.). 2019. *The Semantic Web - 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019, Proceedings*. Lecture Notes in Computer Science, Vol. 11503. Springer.

[227] Pascal Hitzler and Adila Krisnadhi. 2018. A Tutorial on Modular Ontology Modeling with Ontology Design Patterns: The Cooking Recipes Ontology. *CoRR* abs/1808.08433 (2018), 22. arXiv:1808.08433 http://arxiv.org/abs/1808.08433

[228] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. 2012. *OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation 11 December 2012*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

[229] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. 2010. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press.

[230] Vinh Thinh Ho, Daria Stepanova, Mohamed H. Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. 2018. Rule Learning from Knowledge Graphs Guided by Embedding Models, See [514], 72–90. https://doi.org/10.1007/978-3-030-00671-6_5

[231] Cornelis Hoede. 1995. On the ontology of knowledge graphs. In *Conceptual Structures: Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS '95, Santa Cruz, California, USA, August 14-18, 1995, Proceedings (Lecture Notes in Computer Science)*, Gerard Ellis, Robert Levinson, William Rich, and John F. Sowa (Eds.), Vol. 954. Springer, 308–322.

[232] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, and Gerhard Weikum. 2011. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011 (Companion Volume)*, Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar (Eds.). ACM Press, 229–232.

[233] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (Eds.). The Association for Computational Linguistics,

541–550.

[234] Aidan Hogan. 2017. Canonical Forms for Isomorphic and Equivalent RDF Graphs: Algorithms for Leaning and Labelling Blank Nodes. *ACM Transactions on the Web* 11, 4 (2017), 22:1–22:62. https://doi.org/10.1145/3068333

[235] Aidan Hogan, Marcelo Arenas, Alejandro Mallea, and Axel Polleres. 2014. Everything you always wanted to know about blank nodes. *Journal of Web Semantics* 27-28 (2014), 42–69. https://doi.org/10.1016/j.websem.2014.06.004

[236] Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres. 2010. Weaving the Pedantic Web. In *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010 (CEUR Workshop Proceedings)*, Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas (Eds.), Vol. 628. Sun SITE Central Europe (CEUR), 10. http://ceur-ws.org/Vol-628/ldow2010_paper04.pdf

[237] Aidan Hogan and Tova Milo (Eds.). 2019. *Proceedings of the 13th Alberto Mendelzon International Workshop on Foundations of Data Management, Asunción, Paraguay, June 3-7, 2019*. CEUR Workshop Proceedings, Vol. 2369. Sun SITE Central Europe (CEUR).

[238] Aidan Hogan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, and Stefan Decker. 2012. An empirical survey of Linked Data conformance. *Journal of Web Semantics* 14 (2012), 14–44. https://doi.org/10.1016/j.websem.2012.02.001

[239] Aidan Hogan, Antoine Zimmermann, Jürgen Umbrich, Axel Polleres, and Stefan Decker. 2012. Scalable and distributed methods for entity matching, consolidation and disambiguation over Linked Data corpora. *Journal of Web Semantics* 10 (2012), 76–110.

[240] Martin Homola and Luciano Serafini. 2012. Contextualized Knowledge Repositories for the Semantic Web. *Journal of Web Semantics* 12 (2012), 64–87.

[241] Ian Horrocks and Peter F. Patel-Schneider. 2004. Reducing OWL entailment to description logic satisfiability. *Journal of Web Semantics* 1, 4 (2004), 345–357.

[242] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. 2004. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004*. W3C Member Submission. https://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[243] Andy Hunt and Dave Thomas. 2003. The Trip-Packing Dilemma. *IEEE Software* 20, 3 (2003), 106–107.

[244] Dylan Hutchison, Bill Howe, and Dan Suciu. 2017. LaraDB: A Minimalist Kernel for Linear and Relational Algebra Computation. In *Proceedings of the 4th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond, BeyondMR@SIGMOD 2017, Chicago, IL, USA, May 19, 2017*, Foto N. Afrati and Jacek Sroka (Eds.). ACM Press, 2:1–2:10.

[245] Eero Hyvönen, Eetu Mäkelä, Tomi Kauppinen, Olli Alm, Jussi Kurki, Tuukka Ruotsalo, Katri Seppälä, Joeli Takala, Kimmo Puputti, Heini Kuittinen, Kim Viljanen, Jouni Tuominen, Tuomas Palonen, Matias Frosterus, Reetta Sinkkilä, Panu Paakkarinen, Joonas Laitio, and Katariina Nyberg. 2009. CultureSampo: A National Publication System of Cultural Heritage on the Semantic Web 2.0. In *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*, Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl (Eds.), Vol. 5554. Springer, 851–856.

[246] Andreea Iana, Steffen Jung, Philipp Naeser, Aliaksandr Birukou, Sven Hertling, and Heiko Paulheim. 2019. Building a Conference Recommender System Based on SciGraph and WikiCFP. In *Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019, Karlsruhe, Germany, September 9-12, 2019, Proceedings (Lecture Notes in Computer Science)*, Maribel Acosta, Philippe Cudré-Mauroux, Maria Maleshkova, Tassilo Pellegrini, Harald Sack, and York Sure-Vetter (Eds.), Vol. 11702. Springer, 117–123.

[247] Renato Iannella and Serena Villata. 2018. *ODRL Information Model 2.2*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/odrl-model/

[248] ICDE 2019 2019. *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE Computer Society.

[249] International Health Terminology Standards Development Organisation 2019. *SNOMED CT Editorial Guide*. International Health Terminology Standards Development Organisation. https://confluence.ihtsdotools.org/display/DOCEG?preview=/71172150/94404969/SNOMED%20CT%20Editorial%20Guide-20190731.pdf

[250] Alexandru Iosup, Tim Hegeman, Wing Lung Ngai, Stijn Heldens, Arnau Prat-Pérez, Thomas Manhardt, Hassan Chafi, Mihai Capota, Narayanan Sundaram, Michael J. Anderson, Ilie Gabriel Tanase, Yinglong Xia, Lifeng Nai, and Peter A. Boncz. 2016. LDBC Graphalytics: A Benchmark for Large-Scale Graph on Parallel and Distributed Platforms. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1317–1328.

[251] Robert Isele, Anja Jentzsch, and Christian Bizer. 2011. Efficient multidimensional blocking for link discovery without losing recall. In *Proceedings of the 14th International Workshop on the Web and Databases 2011, WebDB 2011, Athens, Greece, June 12, 2011*, Amélie Marian and Vasilis Vassalos (Eds.). 6.

[252] P. James. 1992. Knowledge Graphs. *Linguistic Instruments in Knowledge Engineering* (1992).

[253] Daniel Janke, Steffen Staab, and Matthias Thimm. 2018. Impact analysis of data placement strategies on query efforts in distributed RDF stores. *Journal of Web Semantics* 50 (2018), 21–48. https://doi.org/10.1016/j.websem.2018.02.002

[254] Krzysztof Janowicz, Pascal Hitzler, Benjamin Adams, Dave Kolas, and Charles Vardeman. 2014. Five stars of Linked Data vocabulary use. *Semantic Web Journal* 5, 3 (2014), 173–176.

[255] Krzysztof Janowicz, Bo Yan, Blake Regalia, Rui Zhu, and Gengchen Mai. 2018. Debiasing Knowledge Graphs: Why Female Presidents are not like Female Popes, See [505], 5. http://ceur-ws.org/Vol-2180/ISWC_2018_Outrageous_Ideas_paper_17.pdf

[256] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on NaturalLanguage Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computational Linguistics, 687–696. https://www.aclweb.org/anthology/volumes/P15-1/

[257] Yun-fei Jiang and Ning Ma. 2002. A Plan Recognition Algorithm Based on Plan Knowledge Graph. *Journal of Software* 13 (2002).

[258] Dan Jurafsky and James H. Martin. 2019. *Speech and Language Processing*. https://web.stanford.edu/~jurafsky/slp3/ Draft chapters in progress.

[259] David Jurgens and Roberto Navigli. 2014. It's All Fun and Games until Someone Annotates: Video Games with a Purpose for Linguistic Annotation. *Transactions of the Association for Computational Linguistics* 2 (2014), 449–464.

[260] Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, and Aidan Hogan. 2013. Observing Linked Data Dynamics, See [90], 213–227.

[261] Lucie-Aimée Kaffee, Alessandro Piscopo, Pavlos Vougiouklis, Elena Simperl, Leslie Carr, and Lydia Pintscher. 2017. A Glimpse into Babel: An Analysis of Multilinguality in Wikidata. In *Proceedings of the 13th International Symposium on Open Collaboration, OpenSym 2017, Galway, Ireland, August 23-25, 2017*, Lorraine Morgan (Ed.). ACM Press, 14:1–14:5.

[262] Hans Kamp. 1981. A Theory of Truth and Semantic Representation. In *Formal Semantics – the Essential Readings*, Paul H. Portner and Barbara H. Partee (Eds.). Blackwell, 189–222.

[263] Elias Kärle, Umutcan Simsek, Oleksandra Panasiuk, and Dieter Fensel. 2018. Building an Ecosystem for the Tyrolean Tourism Knowledge Graph. *CoRR* abs/1805.05744 (2018), 8. http://arxiv.org/abs/1805.05744

[264] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum. 2008. NAGA: Searching and Ranking Knowledge, See [10], 953–962.

[265] Andreas Kasten, Ansgar Scherp, Frederik Armknecht, and Matthias Krause. 2013. Towards Search on Encrypted Graph Data. In *Proceedings of the Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn2013) co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 22, 2013 (CEUR Workshop Proceedings)*, Stefan Decker, Jim Hendler, and Sabrina Kirrane (Eds.), Vol. 1121. Sun SITE Central Europe (CEUR), 46–57. http://ceur-ws.org/Vol-1121/privon2013_paper5.pdf

[266] Seyed Mehran Kazemi and David Poole. 2018. SimplE Embedding for Link Prediction in Knowledge Graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 4289–4300. http://papers.nips.cc/book/advances-in-neural-information-processing-systems-31-2018

[267] C. Maria Keet. 2018. *An Introduction to Ontology Engineering*. College Publications.

[268] C. Maria Keet and Agnieszka Ławrynowicz. 2016. Test-driven development of ontologies. In *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings (Lecture Notes in Computer Science)*, Harald Sack, Eva Blomqvist, Mathieu d'Aquin, Chiara Ghidini, Simone Paolo Ponzetto, and Christoph Lange (Eds.), Vol. 9678. Springer, 642–657.

[269] Elisa F. Kendall and Deborah L. McGuinness. 2019. *Ontology Engineering*. Synthesis Lectures on the Semantic Web: Theory and Technology, Vol. 9. Morgan & Claypool.

[270] Michael Kifer and Harold Boley. 2013. *RIF Overview (Second Edition), W3C Working Group Note 5 February 2013*. W3C Working Group Note. World Wide Web Consortium. https://www.w3.org/TR/2013/NOTE-rif-overview-20130205/.

[271] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 14. https://openreview.net/forum?id=SJU4ayYgl

[272] Sabrina Kirrane, Ahmed Abdelrahman, Alessandra Mileo, and Stefan Decker. 2013. Secure Manipulation of Linked Data, See [8], 248–263.

[273] Sabrina Kirrane, Alessandra Mileo, and Stefan Decker. 2017. Access control and the Resource Description Framework: A survey. *Semantic Web Journal* 8, 2 (2017), 311–352. https://doi.org/10.3233/SW-160236

[274] Jon M Kleinberg. 1999. Hubs, authorities, and communities. *ACM Computing Surveys* 31, 4es (1999), 5.

[275] Peter Kluegl, Martin Atzmueller, and Frank Puppe. 2009. TextMarker: A Tool for Rule-Based Information Extraction. In *UIMAGSCL Workshop*. 233–240.

[276] Kevin Knight, Hwee Tou Ng, and Kemal Oflazer (Eds.). 2005. *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*. The Association for Computational Linguistics.

[277] Holger Knublauch, James A. Hendler, and Kingsley Idehen. 2011. *SPIN – Overview and Motivation, W3C Member Submission 22 February 2011*. W3C Member Submission. https://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/.

[278] Holger Knublauch and Dimitris Kontokostas. 2017. *Shapes Constraint Language (SHACL), W3C Recommendation 20 July 2017*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2017/REC-shacl-20170720/

[279] Hanna Köpcke and Erhard Rahm. 2010. Frameworks for entity matching: A comparison. *Data and Knowledge Engineering* 69, 2 (2010), 197–210. https://doi.org/10.1016/j.datak.2009.10.003

[280] Arun Krishnan. 2018. Making search easier: How Amazon's Product Graph is helping customers find products more easily. Amazon Blog. https://blog.aboutamazon.com/innovation/making-search-easier.

[281] Adila Krisnadhi and Pascal Hitzler. 2016. A Core Pattern for Events. In *Advances in Ontology Design and Patterns [revised and extended versions of the papers presented at the 7th edition of the Workshop on Ontology and Semantic Web Patterns, WOP@ISWC 2016, Kobe, Japan, 18th October 2016] (Studies on the Semantic Web)*, Karl Hammar, Pascal Hitzler, Adila Krisnadhi, Agnieszka Lawrynowicz, Andrea Giovanni Nuzzolese, and Monika Solanki (Eds.), Vol. 32. IOS Press, 29–37.

[282] Adila Krisnadhi and Pascal Hitzler. 2016. Modeling With Ontology Design Patterns: Chess Games As a Worked Example. In *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, Pascal Hitzler, Aldo Gangemi, Krysztof Janowicz, Adila Krisnadhi, and Valentina Presutti (Eds.). Studies on the Semantic Web, Vol. 25. IOS Press, 3–21.

[283] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* 60, 6 (2017), 84–90. https://doi.org/10.1145/3065386

[284] Peter Kümmel. 1973. An Algorithm of Limited Syntax Based on Language Universals. In *Computational And Mathematical Linguistics: Proceedings of the 5th International Conference on Computational Linguistics, COLING 1973, Pisa, Italy, August 27 - September 1, 1973*, Antonio Zampolli and Nicoletta Calzolari (Eds.). The Association for Computational Linguistics, 225–248. https://www.aclweb.org/anthology/volumes/C73-1/

[285] H. T. Kung. 1982. Why Systolic Architectures? *IEEE Computer* 15, 1 (1982), 37–46.

[286] Jose Emilio Labra Gayo, Herminio García-González, Daniel Fernández-Alvarez, and Eric Prud'hommeaux. 2019. Challenges in RDF Validation. In *Current Trends in Semantic Web Technologies: Theory and Practice*, Giner Alor-Hernández, José Luis Sánchez-Cervantes, Alejandro Rodríguez-González, and Rafael Valencia-García (Eds.). Springer, 121–151. https://doi.org/10.1007/978-3-030-06149-4_6

[287] Jose Emilio Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, and Dimitris Kontokostas. 2017. *Validating RDF Data*. Synthesis Lectures on the Semantic Web: Theory and Technology, Vol. 7. Morgan & Claypool. 1–328 pages. https://doi.org/10.2200/s00786ed1v01y201707wbe016

[288] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, Kevin Knight, Ani Nenkova, and Owen Rambow (Eds.). The Association for Computational Linguistics, 260–270.

[289] Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning* 81, 1 (2010), 53–67.

[290] Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Jon Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, David Liu, and Sören Auer. 2012. deqa: Deep Web Extraction for Question Answering. In *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II (Lecture Notes in Computer Science)*, Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist (Eds.), Vol. 7650. Springer, 131–147.

[291] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal* 6, 2 (2015), 167–195.

[292] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A Large Public Corpus of Web Tables containing Time and Context Metadata, See [61], 75–76.

[293] Yuangui Lei, Victoria Uren, and Enrico Motta. 2007. A framework for evaluating semantic metadata. In *Proceedings of the Fourth International Conference on Knowledge Capture*, Derek Sleeman and Ken Barker (Eds.). ACM Press, 135–142.

[294] Douglas B Lenat. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38, 11 (1995), 33–38.

[295] Mark Levene and Alexandra Poulovassilis. 1989. The Hypernode Model: A Graph-Theoretic Approach to Integrating Data and Computation. In *Workshop on Foundations of Models and Languages for Data and Objects, Aigen, Austria, 25.-29. September 1989 (Informatik-Berichte des IfI)*, Andreas Heuer (Ed.), Vol. 89-2. Technische Universität Clausthal, 55–77.

[296] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis (Eds.). IEEE Computer Society, 106–115.

[297] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proceedings of the VLDB Endowment* 3, 1 (2010), 1338–1347.

[298] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 2181–2187.

[299] Zhiyuan Lin and Mahesh Tripunitara. 2017. Graph Automorphism-Based, Semantics-Preserving Security for the Resource Description Framework (RDF). In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017, Scottsdale, AZ, USA, March 22-24, 2017*, Gail-Joon Ahn, Alexander Pretschner, and Gabriel Ghinita (Eds.). ACM Press, 337–348.

[300] Xiao Ling and Daniel S. Weld. 2012. Fine-Grained Entity Recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*, Jörg Hoffmann and Bart Selman (Eds.). AAAI Press, 94–100. http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5152

[301] Kun Liu and Evimaria Terzi. 2008. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, Jason Tsong-Li Wang (Ed.). ACM Press, 93–106.

[302] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Computing Surveys* 51, 3 (2018), 62:1–62:34.

[303] John W. Lloyd. 2012. *Foundations of logic programming.* Springer.

[304] Colin Lockard, Xin Luna Dong, Prashant Shiralkar, and Arash Einolghozati. 2018. CERES: Distantly Supervised Relation Extraction from the Semi-Structured Web. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1084–1096.

[305] Dave Longley and Manu Sporny. 2019. *RDF Dataset Normalization, A Standard RDF Dataset Normalization Algorithm, Draft Community Group Report 27 February 2019.* W3C Community Group Draft Report. http://json-ld.github.io/normalization/spec/

[306] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. 2012. Distributed GraphLab: A Framework for Machine Learning in the Cloud. *Proceedings of the VLDB Endowment* 5, 8 (2012), 716–727.

[307] Chunliang Lu, Lidong Bing, Wai Lam, Ki Chan, and Yuan Gu. 2013. Web Entity Detection for Semi-structured Text Data Records with Unlabeled Data. *International Journal of Computational Linguistics and Applications* 4, 2 (2013), 135–150.

[308] Chun Lu, Philippe Laublet, and Milan Stankovic. 2016. Travel Attractions Recommendation with Knowledge Graphs, See [45], 416–431. https://doi.org/10.1007/978-3-319-49004-5_27

[309] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari. 2013. Complexity of Inconsistency-Tolerant Query Answering in Datalog+/-. In *Informal Proceedings of the 26th International Workshop on Description Logics, Ulm, Germany, July 23 - 26, 2013 (CEUR Workshop Proceedings)*, Thomas Eiter, Birte Glimm, Yevgeny Kazakov, and Markus Krötzsch (Eds.), Vol. 1014. Sun SITE Central Europe (CEUR), 791–803. http://ceur-ws.org/Vol-1014/paper_6.pdf

[310] Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint Entity Recognition and Disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton (Eds.). The Association for Computational Linguistics, 879–888. https://www.aclweb.org/anthology/volumes/D15-1/

[311] Ricardo José Machado and Armando Freitas da Rocha. 1990. The Combinatorial Neural Network: A Connectionist Model for Knowledge Based Systems. In *Uncertainty in Knowledge Bases, 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU '90, Paris, France, July 2-6, 1990, Proceedings (Lecture Notes in Computer Science)*, Bernadette Bouchon-Meunier, Ronald R. Yager, and Lotfi A. Zadeh (Eds.), Vol. 521. Springer, 578–587.

[312] Jayant Madhavan, David Ko, Lucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Y. Halevy. 2008. Google's Deep Web crawl. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1241–1252.

[313] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A Knowledge Base from Multilingual Wikipedias, See [88], 11. http://cidrdb.org/cidr2015/Papers/CIDR15_Paper1.pdf

[314] Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, Ahmed K. Elmagarmid and Divyakant Agrawal (Eds.). ACM Press, 135–146.

[315] Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. 2018. Getting the most out of Wikidata: Semantic technology usage in Wikipedia's knowledge graph. In *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II (Lecture Notes in Computer Science)*, Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl (Eds.), Vol. 11137. Springer, 376–394.

[316] Ezio Marchi and Osvaldo Miguel. 1974. On the structure of the teaching-learning interactive process. *International Journal of Game Theory* 3 (1974), 83–99. Issue 2.

[317] Jose L. Martínez-Rodríguez, Aidan Hogan, and Ivan Lopez-Arevalo. 2020. Information Extraction meets the Semantic Web: A Survey. *Semantic Web Journal* 11, 2 (2020), 255–335.

[318] Jose L. Martínez-Rodríguez, Ivan López-Arévalo, and Ana B. Rios-Alvarado. 2018. OpenIE-based approach for Knowledge Graph construction from text. *Expert Systems With Applications* 113 (2018), 339–355.

[319] Cynthia Matuszek, John Cabral, Michael J. Witbrock, and John De Oliveira. 2006. An Introduction to the Syntax and Content of Cyc. In *Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-05, Stanford, California, USA, March 27-29, 2006*. AAAI Press, 44–49. http://www.aaai.org/Library/Symposia/Spring/ss06-05.php

[320] Mausam. 2016. Open Information Extraction Systems and Downstream Applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI, 4074–4077.

[321] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open Language Learning for Information Extraction, See [500], 523–534. https://www.aclweb.org/anthology/volumes/D12-1/

[322] Diana Maynard, Kalina Bontcheva, and Isabelle Augenstein. 2016. *Natural Language Processing for the Semantic Web*. Morgan & Claypool.

[323] John McCarthy. 1990. *Formalizing Commonsense*. Greenwood Publishing Group.

[324] John McCarthy. 1993. Notes on Formalizing Context. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, Ruzena Bajcsy (Ed.). Morgan Kaufmann, 555–562.

[325] Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). 2018. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

[326] Edgar Meij. 2019. Understanding News using the Bloomberg Knowledge Graph. Invited talk at the Big Data Innovators Gathering (TheWebConf). Slides at https://speakerdeck.com/emeij/understanding-news-using-the-bloomberg-knowledge-graph.

[327] Pablo N. Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A Multilingual Cross-domain Knowledge Base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis (Eds.). European Language Resources Association (ELRA), 1813–1817.

[328] Pablo N. Mendes, Hannes Mühleisen, and Christian Bizer. 2012. Sieve: linked data quality assessment and fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops, Berlin, Germany, March 30, 2012*, Divesh Srivastava and Ismail Ari (Eds.). Journal of the ACM, 116–123.

[329] Peter Mika, Tania Tudorache, Abraham Bernstein, Christopher A. Welty, Craig A. Knoblock, Denny Vrandecic, Paul T. Groth, Natasha Fridman Noy, Krzysztof Janowicz, and Carole A. Goble (Eds.). 2014. *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*. Lecture Notes in Computer Science, Vol. 8796. Springer.

[330] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space, See [35], 12. http://arxiv.org/abs/1301.3781 arXiv preprint arXiv:1301.3781.

[331] George A. Miller and Christiane Fellbaum. 2007. WordNet then and now. *Language Resources and Evaluation (LRE)* 41, 2 (2007), 209–214.

[332] Justin J. Miller. 2013. Graph Database Applications and Concepts with Neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA March 23rd-24th, 2013*. AIS eLibrary, Article 24, 7 pages. https://aisel.aisnet.org/sais2013/24

[333] Marvin Minsky. 1974. A Framework for representing knowledge. *MIT-AI Memo 306, Santa Monica* (1974), 76.

[334] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, Keh-Yih Su, Jian Su, and Janyce Wiebe (Eds.). The Association for Computational Linguistics, 1003–1011.

[335] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha P. Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2018. Never-ending learning. *Communications of the ACM* 61, 5 (2018), 103–115.

[336] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 5425–5434. https://doi.org/10.1109/CVPR.2017.576

[337] Elena Montiel-Ponsoda, Víctor Rodríguez-Doncel, and Jorge Gracia. 2017. Building the Legal Knowledge Graph for Smart Compliance Services in Multilingual Europe. In *Proceedings of the 1st Workshop on Technologies for Regulatory Compliance co-located with the 30th International Conference on Legal Knowledge and Information Systems (JURIX 2017), Luxembourg, December 13, 2017 (CEUR Workshop Proceedings)*, Víctor Rodríguez-Doncel, Pompeu Casanovas, and Jorge González-Conejero (Eds.), Vol. 2049. Sun SITE Central Europe (CEUR), 15–17. http://ceur-ws.org/Vol-2049/02paper.pdf

[338] Benjamin Moreau, Patricia Serrano-Alvarado, Matthieu Perrin, and Emmanuel Desmontils. 2019. Modelling the Compatibility of Licenses, See [226], 255–269.

[339] Andrea Moro and Roberto Navigli. 2013. Integrating Syntactic and Semantic Analysis into the Open Information Extraction Paradigm. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, Francesca Rossi (Ed.). IJCAI/AAAI, 2148–2154.

[340] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.

[341] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. 2012. *OWL 2 Web Ontology Language Profiles (Second Edition), W3C Recommendation 11 December 2012*. W3C Recommendation. World Wide Web Consortium. http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

[342] Boris Motik, Rob Shearer, and Ian Horrocks. 2009. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research* 36 (2009), 165–228.

[343] Varish Mulwad, Tim Finin, and Anupam Joshi. 2013. Semantic Message Passing for Generating Linked Data from Tables, See [8], 363–378.

[344] Chris Mungall, Alan Ruttenberg, Ian Horrocks, and David Osumi-Sutherland. 2012. OBO Flat File Format 1.4 Syntax and Semantics. Editor's Draft. http://owlcollab.github.io/oboformat/doc/obo-syntax.html

[345] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. 2014. Using Linked Data to mine RDF from Wikipedia's tables. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, Ben Carterette, Fernando Diaz, Carlos Castillo, and Donald Metzler (Eds.). ACM Press, 533–542.

[346] Sergio Muñoz, Jorge Pérez, and Claudio Gutiérrez. 2009. Simple and Efficient Minimal RDFS. *Journal of Web Semantics* 7, 3 (2009), 220–234.

[347] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26.

[348] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained Semantic Typing of Emerging Entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*. The Association for Computational Linguistics, 1488–1497. https://www.aclweb.org/anthology/volumes/P13-1/

[349] Arvind Narayanan and Vitaly Shmatikov. 2009. De-anonymizing Social Networks. In *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*. IEEE Computer Society, 173–187.

[350] Roberto Navigli. 2009. Word Sense Disambiguation: A Survey. *ACM Computing Surveys* 41, 2 (2009), 1–69.

[351] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193 (2012), 217–250.

[352] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. 2017. A survey of current link discovery frameworks. *Semantic Web Journal* 8, 3 (2017), 419–436.

[353] Sebastian Neumaier and Axel Polleres. 2019. Enabling Spatio-Temporal Search in Open Data. *Journal of Web Semantics* 55 (2019), 21 – 36. https://doi.org/10.1016/j.websem.2018.12.007

[354] Sebastian Neumaier, Jürgen Umbrich, Josiane Xavier Parreira, and Axel Polleres. 2016. Multi-level Semantic Labelling of Numerical Values, See [194], 428–445.

[355] David Newman. 2019. Knowledge Graphs and AI: The Future of Financial Data. In *Knowledge Graph Conference*. (Invited talk).

[356] Axel-Cyrille Ngonga Ngomo and Sören Auer. 2011. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data, See [518], 2312–2317.

[357] Axel-Cyrille Ngonga Ngomo. 2012. Link discovery with guaranteed reduction ratio in affine spaces with Minkowski measures, See [105], 378–393.

[358] Axel-Cyrille Ngonga Ngomo. 2013. ORCHID–reduction-ratio-optimal computation of geo-spatial distances for link discovery, See [8], 395–410.

[359] Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, I don't speak SPARQL: translating SPARQL queries into natural language, See [444], 977–988.

[360] Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2016. J-NERD: Joint Named Entity Recognition and Disambiguation with Rich Linguistic Features. *Transactions of the Association for Computational Linguistics* 4 (2016), 215–229.

[361] Vinh Nguyen, Olivier Bodenreider, and Amit Sheth. 2014. Don't Like RDF Reification?: Making Statements About Statements Using Singleton Property, See [86], 759–770.

[362] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE* 104, 1 (2016), 11–33.

[363] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic Embeddings of Knowledge Graphs, See [443], 1955–1961.

[364] Maximilian Nickel and Volker Tresp. 2013. Tensor factorization for multi-relational learning. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III (Lecture Notes in Computer Science)*, Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Zelezný (Eds.), Vol. 8190. Springer, 617–621.

[365] Natasha F. Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale Knowledge Graphs: Lessons and Challenges. *ACM Queue* 17, 2 (2019), 20.

[366] Natalya F. Noy and Deborah L. McGuinness. 2001. *Ontology Development 101: A Guide to Creating Your First Ontology*. Technical Report. Stanford Knowledge Systems Laboratory. https://protege.stanford.edu/publications/ontology_development/ontology101.pdf

[367] Sri Nurdiati and Cornelis Hoede. 2012. 25 Years of Development of Knowledge Graph Theory: the Results and the Challenge. Memorandum 1876, University of Twente. https://core.ac.uk/download/pdf/11468596.pdf

[368] Ekpe Okorafor and Atish Ray. 2019. The path from data to knowledge. Accenture Applied Intelligence Blog. https://www.accenture.com/us-en/insights/digital/data-to-knowledge.

[369] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing order to the Web*. Technical Report 1999-66. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/

[370] Oleksandra Panasiuk, Simon Steyskal, Giray Havur, Anna Fensel, and Sabrina Kirrane. 2018. Modeling and Reasoning over Data Licenses. In *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers (Lecture Notes in Computer Science)*, Aldo Gangemi, Anna Lisa Gentile, Andrea Giovanni Nuzzolese, Sebastian Rudolph, Maria Maleshkova, Heiko Paulheim, Jeff Z. Pan, and Mehwish Alam (Eds.), Vol. 11155. Springer, 218–222.

[371] Harshvardhan J. Pandit, Axel Polleres, Bert Bos, Rob Brennan, Bud Bruegger, Fajar J. Ekaputra, Javier D. Fernández, Ramisa Gachpaz Hamed, Elmar Kiesling, Mark Lizar, Eva Schlehahn, Simon Steyskal, and Rigo Wenning. 2019. *Data Privacy Vocabulary v0.1*. Draft Community Group Report. World Wide Web Consortium. https://www.w3.org/ns/dpv

[372] Vicky Papavasileiou, Giorgos Flouris, Irini Fundulaki, Dimitris Kotzinos, and Vassilis Christophides. 2013. High-level change detection in RDF(S) KBs. *ACM Transactions on Database Systems* 38, 1 (2013), 1:1–1:42.

[373] Jeff Pasternack and Dan Roth. 2010. Knowing What to Believe (when You Already Know Something). In *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, Chu-Ren Huang and Dan Jurafsky (Eds.). Tsinghua University Press, 877–885. https://www.aclweb.org/anthology/volumes/C10-1/

[374] Jeff Pasternack and Dan Roth. 2011. Making Better Informed Trust Decisions with Generalized Fact-Finding, See [518], 2324–2329.

[375] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web Journal* 8, 3 (2017), 489–508. https://doi.org/10.3233/SW-160218

[376] Heiko Paulheim. 2018. How much is a Triple? Estimating the Cost of Knowledge Graph Creation, See [505], 4. http://ceur-ws.org/Vol-2180/ISWC_2018_Outrageous_Ideas_paper_10.pdf

[377] Heiko Paulheim and Christian Bizer. 2013. Type inference on noisy RDF data, See [7], 510–525.

[378] Chaveevan Pechsiri and Rapepun Piriyakul. 2010. Explanation Knowledge Graph Construction Through Causality Extraction from Texts. *Journal of Computer Science and Technology* 25, 5 (2010), 1055–1070.

[379] Charles S. Peirce. 1878. How to Make Our Ideas Clear. *Popular Science Monthly* 12 (1878), 286–302.

[380] Tassilo Pellegrini, Giray Havur, Simon Steyskal, Oleksandra Panasiuk, Anna Fensel, Victor Mireles, Thomas Thurner, Axel Polleres, Sabrina Kirrane, and Andrea Schönhofer. 2019. DALICC: A License Management Framework for Digital Assets. In *Proceedings of the Internationales Rechtsinformatik Symposion (IRIS)*. 10.

[381] Thomas Pellissier Tanon, Daria Stepanova, Simon Razniewski, Paramita Mirza, and Gerhard Weikum. 2017. Completeness-Aware Rule Learning from Knowledge Graphs, See [108], 507–525. https://doi.org/10.1007/978-3-319-68288-4_30

[382] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From Freebase to Wikidata: The Great Migration. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016*, Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao (Eds.). ACM Press, 1419–1428.

[383] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). The Association for Computational Linguistics, 1532–1543. https://www.aclweb.org/anthology/volumes/D14-1/

[384] Silvio Peroni. 2016. A simplified agile methodology for ontology development. In *OWL: - Experiences and Directions - Reasoner Evaluation - 13th International Workshop, OWLED 2016, and 5th International Workshop, ORE 2016, Bologna, Italy, November 20, 2016, Revised Selected Papers (Lecture Notes in Computer Science)*, Mauro Dragoni, María Poveda-Villalón, and Ernesto Jiménez-Ruiz (Eds.), Vol. 10161. Springer, 55–69.

[385] Silvio Peroni, David M. Shotton, and Fabio Vitali. 2017. One Year of the OpenCitations Corpus – Releasing RDF-Based Scholarly Citation Data into the Public Domain. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II (Lecture Notes in Computer Science)*, Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin (Eds.), Vol. 10588. Springer, 184–192.

[386] David Peterson, Shudi Gao, Ashok Malhotra, C. M. Sperberg-McQueen, Henry S. Thompson, and Paul V. Biron. 2012. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, W3C Recommendation 5 April 2012*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/

[387] Giulio Petrucci, Chiara Ghidini, and Marco Rospocher. 2016. Ontology learning in the deep, See [45], 480–495.

[388] Minh-Duc Pham, Linnea Passing, Orri Erling, and Peter A. Boncz. 2015. Deriving an Emergent Relational Schema from RDF Data, See [168], 864–874.

[389] H. Sofia Pinto, C. Tempich, and Steffen Staab. 2009. Ontology Engineering and Evolution in a Distributed World Using DILIGENT. See [471], 153–176. https://doi.org/10.1007/978-3-540-92673-3

[390] Alessandro Piscopo, Lucie-Aimée Kaffee, Chris Phethean, and Elena Simperl. 2017. Provenance Information in a Collaborative Knowledge Graph: An Evaluation of Wikidata External References, See [108], 542–558.

[391] Alessandro Piscopo and Elena Simperl. 2018. Who Models the World?: Collaborative Ontology Creation and User Roles in Wikidata. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 141:1–141:18. https://doi.org/10.1145/3274410

[392] R. J. Pittman, Amit Srivastava, Sanjika Hewavitharana, Ajinkya Kale, and Saab Mansour. 2017. Cracking the Code on Conversational Commerce. eBay Blog. https://www.ebayinc.com/stories/news/cracking-the-code-on-conversational-commerce/.

[393] Aleksander Pivk, Philipp Cimiano, York Sure, Matjaz Gams, Vladislav Rajkovic, and Rudi Studer. 2007. Transforming arbitrary tables into logical form with TARTAR. *Data and Knowledge Engineering* 60, 3 (2007), 567–595.

[394] Roel Popping. 2003. Knowledge Graphs and Network Text Analysis. *Social Science Information* 42, 91 (2003), 91–106.

[395] Valentina Presutti, Enrico Daga, Aldo Gangemi, and Eva Blomqvist. 2009. eXtreme Design with Content Ontology Design Patterns. In *Proceedings of the Workshop on Ontology Patterns (WOP 2009) , collocated with the 8th International Semantic Web Conference ( ISWC-2009 ), Washington D.C., USA, 25 October, 2009 (CEUR Workshop Proceedings)*, Eva Blomqvist, Kurt Sandkuhl, François Scharffe, and Vojtech Svátek (Eds.), Vol. 516. Sun SITE Central Europe (CEUR), 15. http://ceur-ws.org/Vol-516/pap21.pdf

[396] Valentina Presutti, Claudia d'Amato, Fabien Gandon, Mathieu d'Aquin, Stephen Staab, and Anna Tordia (Eds.). 2014. *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*. Lecture Notes in Computer Science, Vol. 8465. Springer.

[397] Eric Prud'hommeaux and Gavin Carothers. 2014. *RDF 1.1 Turtle – Terse RDF Triple Language, W3C Recommendation 25 February 2014*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2014/REC-turtle-20140225/

[398] Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. 2014. Shape Expressions: An RDF Validation and Transformation Language, See [429], 32–40. https://doi.org/10.1145/2660517.2660523

[399] Jay Pujara, Hui Miao, Lise Getoor, and William W. Cohen. 2013. Knowledge Graph Identification, See [8], 542–557. https://doi.org/10.1007/978-3-642-41335-3_34

[400] Guilin Qi, Huajun Chen, Kang Liu, Haofen Wang, Qiu Ji, and Tianxing Wu. 2020. *Knowledge Graph*. Springer. (to appear).

[401] Ross Quillian. 1963. *A notation for representing conceptual information: An application to semantics and mechanical English paraphrasing. SP-1395, System Development Corporation*. Technical Report SP-1395. Systems Development Corp., Santa Monica, California.

[402] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. 2017. Introduction to Tensor Decompositions and their Applications in Machine Learning. *CoRR* abs/1711.10781 (2017), 13. arXiv:1711.10781 http://arxiv.org/abs/1711.10781

[403] Roy Rada. 1986. Gradualness eases refinement of medical knowledge. *Medical Informatics* 11, 1 (1986), 59–73.

[404] Filip Radulovic, Raúl García-Castro, and Asunción Gómez-Pérez. 2015. Towards the Anonymisation of RDF Data. In *The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 6-8, 2015*, Haiping Xu (Ed.). KSI Research Inc. and Knowledge Systems Institute Graduate School, 646–651.

[405] Luc De Raedt (Ed.). 2008. *Logical and Relational Learning: From ILP to MRDM (Cognitive Technologies)*. Springer-Verlag.

[406] Yves Raimond, Tristan Ferne, Michael Smethurst, and Gareth Adams. 2014. The BBC World Service Archive prototype. *Journal of Web Semantics* 27-28 (2014), 2–9.

[407] Yves Raimond, Christopher Sutton, and Mark B. Sandler. 2009. Interlinking Music-Related Data on the Web. *IEEE MultiMedia* 16, 2 (2009), 52–63.

[408] Alain T. Rappaport and Albert M. Gouyet. 1988. Dynamic, interactive display system for a knowledge base. US Patent US4752889A.

[409] Lev-Arie Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009, Boulder, Colorado, USA, June 4-5, 2009*, Suzanne Stevenson and Xavier Carreras (Eds.). The Association for Computational Linguistics, 147–155.

[410] Pavan Reddivari, Tim Finin, and Anupam Joshi. 2005. Policy-based access control for an RDF store. In *Policy Management for the Web, A workshop held at the 14th International World Wide Web Conference Tuesday 10 May 2005, Chiba Japan*, Lalana Kagal, Tim Finin, and James Hendler (Eds.). 78–81. https://ebiquity.umbc.edu/_file_directory_/papers/159.pdf

[411] Raymond Reiter. 1987. A Theory of Diagnosis from First Principles. *Artificial Intelligence* 32, 1 (1987), 57–95.

[412] Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R. Voss, and Jiawei Han. 2015. ClusType: Effective Entity Recognition and Typing by Relation Phrase-Based Clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM Press, 995–1004.

[413] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, Tarek F. Abdelzaher, and Jiawei Han. 2017. CoType: Joint Extraction of Typed Entities and Relations with Knowledge Bases. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM Press, 1015–1024.

[414] Juan L. Reutter, Adrián Soto, and Domagoj Vrgoc. 2015. Recursion in SPARQL. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I (Lecture Notes in Computer Science)*, Marcelo Arenas, Óscor Corcho, Elena Paslaru Bontas Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul T. Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Stephen Staab (Eds.), Vol. 9366. Springer, 19–35.

[415] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions without Labeled Text. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III (Lecture Notes in Computer Science)*, José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag (Eds.), Vol. 6323. Springer, 148–163.

[416] Petar Ristoski and Heiko Paulheim. 2016. RDF2Vec: RDF Graph Embeddings for Data Mining, See [194], 498–514.

[417] Richard H. Ritchens. 1956. General program for mechanical translation between any two languages via an algebraic interlingua. *Mechanical Translation* 3, 2 (Nov. 1956), 37.

[418] Giuseppe Rizzo, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. 2017. Terminological Cluster Trees for Disjointness Axiom Discovery, See [47], 184–201.

[419] Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end Differentiable Proving, See [204], 3788–3800. http://papers.nips.cc/paper/6969-end-to-end-differentiable-proving

[420] Marko A. Rodriguez. 2015. The Gremlin graph traversal machine and language. In *Proceedings of the 15th Symposium on Database Programming Languages, Pittsburgh, PA, USA, October 25-30, 2015*, James Cheney and Thomas Neumann

(Eds.). ACM Press, 1–10.

[421] Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst Patterns Revisited: Automatic Hypernym Detection from Large Text Corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, Iryna Gurevych and Yusuke Miyao (Eds.). The Association for Computational Linguistics, 358–363. https://www.aclweb.org/anthology/volumes/P18-2/

[422] Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. 2016. Building event-centric knowledge graphs from news. *Journal of Web Semantics* 37-38 (2016), 132–151. https://doi.org/10.1016/j.websem.2015.12.004

[423] Jacobo Rouces, Gerard de Melo, and Katja Hose. 2015. Framebase: Representing *n*-ary relations using semantic frames. In *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings (Lecture Notes in Computer Science)*, Fabien Gandon, Marta Sabou, Harald Sack, Claudia d'Amato, Philippe Cudré-Mauroux, and Antoine Zimmermann (Eds.), Vol. 9088. Springer, 505–521.

[424] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. 2008. Description Logic Reasoning with Decision Diagrams: Compiling SHIQ to Disjunctive Datalog. In *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings (Lecture Notes in Computer Science)*, Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan (Eds.), Vol. 5318. Springer, 435–450.

[425] Anisa Rula, Matteo Palmonari, Andreas Harth, Steffen Stadtmüller, and Andrea Maurino. 2012. On the Diversity and Availability of Temporal Information in Linked Open Data, See [105], 492–507.

[426] Anisa Rula, Matteo Palmonari, Simone Rubinacci, Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Andrea Maurino, and Diego Esteves. 2019. TISCO: Temporal scoping of facts. *Journal of Web Semantics* 54 (2019), 72–86. https://doi.org/10.1016/j.websem.2018.09.002

[427] Anisa Rula, Luca Panziera, Matteo Palmonari, and Andrea Maurino. 2014. Capturing the Currency of DBpedia Descriptions and Get Insight into their Validity. In *Proceedings of the 5th International Workshop on Consuming Linked Data, COLD 2014 co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 20, 2014 (CEUR Workshop Proceedings)*, Olaf Hartig, Aidan Hogan, and Juan F. Sequeda (Eds.), Vol. 1264. Sun SITE Central Europe (CEUR). http://ceur-ws.org/Vol-1264/cold2014_RulaPPM.pdf

[428] Owen Sacco and Alexandre Passant. 2011. A Privacy Preference Ontology (PPO) for Linked Data. In *WWW2011 Workshop on Linked Data on the Web, Hyderabad, India, March 29, 2011 (CEUR Workshop Proceedings)*, Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas (Eds.), Vol. 813. Sun SITE Central Europe (CEUR), 5. http://ceur-ws.org/Vol-813/ldow2011-paper01.pdf

[429] Harald Sack, Agata Filipowska, Jens Lehmann, and Sebastian Hellmann (Eds.). 2014. *Proceedings of the 10th International Conference on Semantic Systems, SEMANTICS 2014, Leipzig, Germany, September 4-5, 2014*. ACM Press.

[430] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Patrick Wang. 2019. DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8–14 December 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 15321–15331. http://papers.nips.cc/paper/9669-drum-end-to-end-differentiable-rule-mining-on-knowledge-graphs

[431] Tomás Sáez and Aidan Hogan. 2018. Automatically Generating Wikipedia Info-boxes from Wikidata. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM Press, 1823–1830.

[432] Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2015. LSQ: The Linked SPARQL Queries Dataset. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II (Lecture Notes in Computer Science)*, Marcelo Arenas, Óscar Corcho, Elena Paslaru Bontas Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul T. Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Stephen Staab (Eds.), Vol. 9367. Springer, 261–269.

[433] Mehdi Samadi, Partha Talukdar, Manuela Veloso, and Manuel Blum. 2016. ClaimEval: Integrated and Flexible Framework for Claim Evaluation Using Credibility of Sources, See [443], 222–228. http://dl.acm.org/citation.cfm?id=3015812.3015845

[434] Pierangela Samarati and Latanya Sweeney. 1998. *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression*. Technical Report SRI-CSL-98-04. Computer Science Laboratory, SRI International. http://www.csl.sri.com/papers/sritr-98-04/

[435] Georgios M. Santipantakis, Apostolos Glenis, Christos Doulkeridis, Akrivi Vlachou, and George A. Vouros. 2019. stLD: towards a spatio-temporal link discovery framework. In *Proceedings of the International Workshop on Semantic Big Data, SBD@SIGMOD 2019, Amsterdam, The Netherlands, July 5, 2019*, Sven Groppe and Le Gruenwald (Eds.). ACM

Press, 4:1–4:6.

[436] Eugene Santos Jr. and Eugene S. Santos. 1999. A framework for building knowledge-bases under uncertainty. *Journal of Experimental & Theoretical Artificial Intelligence* 11, 2 (1999), 265–286.

[437] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80.

[438] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. 2014. Adoption of the Linked Data Best Practices in Different Topical Domains, See [329], 245–260.

[439] Manfred Schmidt-Schauß and Gert Smolka. 1991. Attributive Concept Descriptions with Complements. *Artificial Intelligence* 48, 1 (1991), 1–26. https://doi.org/10.1016/0004-3702(91)90078-X

[440] Edward W. Schneider. 1973. Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis. In *Association for the Development of Instructional Systems (ADIS), Chicago, Illinois, April 1972.*

[441] Michael Schneider and Geoff Sutcliffe. 2011. Reasoning in the OWL 2 Full Ontology Language Using First-Order Automated Theorem Proving. In *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings (Lecture Notes in Computer Science)*, Nikolaj Bjørner and Viorica Sofronie-Stokkermans (Eds.), Vol. 6803. Springer, 461–475.

[442] Christoph Schuetz, Loris Bozzato, Bernd Neumayr, Michael Schrefl, and Luciano Serafini. 2020. Knowledge Graph OLAP: A Multidimensional Model and Query Operations for Contextualized Knowledge Graphs. *Semantic Web Journal* (2020). (Under open review).

[443] Dale Schuurmans and Michael P. Wellman (Eds.). 2016. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.* AAAI Press.

[444] Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo Baeza-Yates, and Sue B. Moon (Eds.). 2013. *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013.* ACM Press.

[445] Philipp Seifer, Johannes Härtel, Martin Leinberger, Ralf Lämmel, and Steffen Staab. 2019. Empirical study on the usage of graph query languages in open source Java projects. In *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2019, Athens, Greece, October 20-22, 2019*, Oscar Nierstrasz, Jeff Gray, and Bruno C. d. S. Oliveira (Eds.). ACM Press, 152–166.

[446] Juan F. Sequeda, Marcelo Arenas, and Daniel P. Miranker. 2012. On directly mapping relational databases to RDF and OWL. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab (Eds.). ACM Press, 649–658.

[447] Juan F. Sequeda, Marcelo Arenas, and Daniel P. Miranker. 2014. OBDA: Query Rewriting or Materialization? In Practice, Both!, See [329], 535–551. https://doi.org/10.1007/978-3-319-11964-9_34

[448] Juan F. Sequeda, Willard J. Briggs, Daniel P. Miranker, and Wayne P. Heideman. 2019. A Pay-as-you-go Methodology to Design and Build Enterprise Knowledge Graphs from Relational Databases, See [179], 526–545.

[449] Stephan Seufert, Patrick Ernst, Srikanta J. Bedathur, Sarath Kumar Kondreddi, Klaus Berberich, and Gerhard Weikum. 2016. Instant Espresso: Interactive Analysis of Relationships in Knowledge Graphs, See [61], 251–254.

[450] Nigel Shadbolt and Kieron O'Hara. 2013. Linked Data in Government. *IEEE Inteternet Computing* 17, 4 (2013), 72–77.

[451] Mohamed Ahmed Sherif and Axel-Cyrille Ngonga Ngomo. 2015. Semantic Quran. *Semantic Web Journal* 6, 4 (2015), 339–345.

[452] Mohamed Ahmed Sherif and Axel-Cyrille Ngonga Ngomo. 2018. A systematic survey of point set distance measures for link discovery. *Semantic Web Journal* 9, 5 (2018), 589–604. https://doi.org/10.3233/SW-170285

[453] Baoxu Shi and Tim Weninger. 2016. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-based Systems* 104 (2016), 123–133.

[454] Cogan Shimizu, Quinn Hirt, and Pascal Hitzler. 2019. MODL: A Modular Ontology Design Library. *CoRR* abs/1904.05405 (2019), 12. arXiv:1904.05405 http://arxiv.org/abs/1904.05405

[455] Solomon Eyal Shimony, Carmel Domshlak, and Eugene Santos Jr. 1997. Cost-Sharing in Bayesian Knowledge Bases. In *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, Dan Geiger and Prakash P. Shenoy (Eds.). Morgan Kaufmann, 421–428.

[456] Prashant Shiralkar, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. 2017. Finding streams in knowledge graphs to support fact checking. In *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, Vijay Raghavan, Srinivas Aluru, George Karypis, Lucio Miele, and Xindong Wu (Eds.). IEEE Computer Society, 859–864.

[457] Saurabh Shrivastava. 2017. Bring rich knowledge of people, places, things and local businesses to your apps. Bing Blogs. https://blogs.bing.com/search-quality-insights/2017-07/bring-rich-knowledge-of-people-places-things-and-local-businesses-to-your-apps.

[458] Rôney Reis C. Silva, Bruno C. Leal, Felipe T. Brito, Vânia M. P. Vidal, and Javam C. Machado. 2017. A Differentially Private Approach for Querying RDF Data of Social Networks. In *Proceedings of the 21st International Database*

*Engineering & Applications Symposium, IDEAS 2017, Bristol, United Kingdom, July 12-14, 2017*, Bipin C. Desai, Jun Hong, and Richard McClatchey (Eds.). ACM Press, 74–81. https://doi.org/10.1145/3105831.3105838

[459] Amit Singhal. 2012. Introducing the Knowledge Graph: things, not strings. Google Blog. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/.

[460] Martin G. Skjæveland, Daniel P. Lupp, Leif Harald Karlsen, and Henrik Forssell. 2018. Practical Ontology Pattern Instantiation, Discovery, and Maintenance with Reasonable Ontology Templates, See [514], 477–494.

[461] Jennifer Sleeman and Tim Finin. 2013. Type Prediction for Efficient Coreference Resolution in Heterogeneous Semantic Graphs. In *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*. IEEE Computer Society, 78–85.

[462] Alisa Smirnova and Philippe Cudré-Mauroux. 2019. Relation Extraction Using Distant Supervision: A Survey. *ACM Computing Surveys* 51, 5 (2019), 106:1–106:35.

[463] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion, See [76], 926–934. http://papers.nips.cc/book/advances-in-neural-information-processing-systems-26-2013

[464] Arnaud Soulet, Arnaud Giacometti, Béatrice Markhoff, and Fabian M. Suchanek. 2018. Representativeness of Knowledge Bases with the Generalized Benford's Law, See [514], 374–390.

[465] John Sowa. 1979. Semantics of Conceptual Graphs. In *17th Annual Meeting of the Association for Computational Linguistics, 29 June - 1 July 1979, University of California at San Diego, La Jolla, CA, USA*, Norman K. Sondheimer (Ed.). The Association for Computational Linguistics, 39–44. https://www.aclweb.org/anthology/P79-1010/

[466] John Sowa. 1987. Semantic Networks. In *Encyclopedia of Cognitive Science*, Stuart C. Shapiro (Ed.). John Wiley & Sons. Revised version available at http://www.jfsowa.com/pubs/semnet.htm.

[467] Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, and Andrea Maurino. 2016. ABSTAT: Ontology-Driven Linked Data Summaries with Pattern Minimalization. In *The Semantic Web - ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers (Lecture Notes in Computer Science)*, Harald Sack, Giuseppe Rizzo, Nadine Steinmetz, Dunja Mladenic, Sören Auer, and Christoph Lange (Eds.), Vol. 9989. Springer, 381–395.

[468] Alessandro Sperduti and Antonina Starita. 1997. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks* 8, 3 (1997), 714–735. https://doi.org/10.1109/72.572108

[469] Manu Sporny, Gregg Kellogg, Markus Lanthaler, Dave Longley, and Niklas Lindström. 2014. *JSON-LD 1.0, A JSON-based Serialization for Linked Data, W3C Recommendation 16 January 2014*. W3C Recommendation. World Wide Web Consortium.

[470] Rajan Srikanth and Matthias Jarke. 1989. The Design of Knowledge-Based Systems for Managing Ill-Structured Software Projects. *Decision Support Systems* 5, 4 (1989), 425–447.

[471] Steffen Staab and Rudi Studer (Eds.). 2009. *Handbook on Ontologies*. Springer. https://doi.org/10.1007/978-3-540-92673-3

[472] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. 2012. LinkedGeoData: A core for a web of spatial open data. *Semantic Web Journal* 3, 4 (2012), 333–354.

[473] Simon Steyskal and Axel Polleres. 2014. Defining expressive access policies for linked data using the ODRL ontology 2.0, See [429], 20–23.

[474] Patrick Stickler. 2005. *CBD − Concise Bounded Description, W3C Member Submission 3 June 2005*. W3C Member Submission. https://www.w3.org/Submission/2005/SUBM-CBD-20050603/

[475] Radu Stoica, George H. L. Fletcher, and Juan F. Sequeda. 2019. On Directly Mapping Relational Databases to Property Graphs, See [237], 4. http://ceur-ws.org/Vol-2369/short06.pdf

[476] Frans N. Stokman and Pieter H. de Vries. 1988. Structuring knowledge in a graph. In *Human-Computer Interaction*, Gerrit C. van der Veer and Gijsbertus Mulder (Eds.). Springer, Chapter 11, 186–206.

[477] Umberto Straccia. 2009. A Minimal Deductive System for General Fuzzy RDF. In *Web Reasoning and Rule Systems, Third International Conference, RR 2009, Chantilly, VA, USA, October 25-26, 2009, Proceedings (Lecture Notes in Computer Science)*, Axel Polleres and Terrance Swift (Eds.), Vol. 5837. Springer, 166–181.

[478] Philip Stutz, Daniel Strebel, and Abraham Bernstein. 2016. Signal/Collect12. *Semantic Web Journal* 7, 2 (2016), 139–166.

[479] Jian Su, Xavier Carreras, and Kevin Duh (Eds.). 2016. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics.

[480] Mari Carmen Suárez-Figueroa, Gong Cheng, Anna Lisa Gentile, Christophe Guéret, C. Maria Keet, and Abraham Bernstein (Eds.). 2019. *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019*. CEUR Workshop Proceedings, Vol. 2456. Sun SITE Central Europe (CEUR). http://ceur-ws.org/Vol-2456

[481] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy (Eds.). ACM Press, 697–706.

[482] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A Large Ontology from Wikipedia and WordNet. *Journal of Web Semantics* 6, 3 (2008), 203–217.

[483] Fabian M. Suchanek, Jonathan Lajus, Armand Boschin, and Gerhard Weikum. 2019. Knowledge Representation and Rule Mining in Entity-Centric Knowledge Bases. In *Reasoning Web. Explainable Artificial Intelligence - 15th International Summer School 2019, Bolzano, Italy, September 20-24, 2019, Tutorial Lectures (Lecture Notes in Computer Science)*, Markus Krötzsch and Daria Stepanova (Eds.), Vol. 11810. Springer, 110–152. https://doi.org/10.1007/978-3-030-31423-1_4

[484] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[485] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction, See [500], 455–465. https://www.aclweb.org/anthology/volumes/D12-1/

[486] Zafar Habeeb Syed, Michael Röder, and Axel-Cyrille Ngonga Ngomo. 2018. FactCheck: Validating RDF Triples Using Textual Evidence. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang (Eds.). ACM Press, 1599–1602.

[487] Zafar Habeeb Syed, Michael Röder, and Axel-Cyrille Ngonga Ngomo. 2019. Unsupervised Discovery of Corroborative Paths for Fact Validation, See [178], 630–646.

[488] James Joseph Sylvester. 1878. Chemistry and Algebra. *Nature* 17 (1878), 284.

[489] Jeremy Tandy, Ivan Herman, and Gregg Kellogg. 2015. *Generating RDF from Tabular Data on the Web, W3C Recommendation 17 December 2015*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2015/REC-csv2rdf-20151217/.

[490] Jeni Tennison and Gregg Kellogg. 2015. *Metadata Vocabulary for Tabular Data, W3C Recommendation 17 December 2015*. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2015/REC-tabular-metadata-20151217/

[491] The R Foundation. 1992. The R Project for Statistical Computing. https://www.r-project.org.

[492] The UniProt Consortium. 2014. UniProt: a hub for protein information. *Nucleic Acids Research* 43, D1 (2014), D204–D212.

[493] Bryan B. Thompson, Mike Personick, and Martyn Cutcher. 2014. The Bigdata® RDF Graph Database. In *Linked Data Management*, Andreas Harth, Katja Hose, and Ralf Schenkel (Eds.). CRC Press, 193–237.

[494] Katherine Thornton, Harold Solbrig, Gregory S. Stupp, José Emilio Labra Gayo, Daniel Mietchen, Eric Prud'hommeaux, and Andra Waagmeester. 2019. Using Shape Expressions (ShEx) to Share RDF Data Models and to Guide Curation with Rigorous Validation, See [226], 606–620. https://doi.org/10.1007/978-3-030-21348-0_39

[495] Felice Tobin. 2017. Thomson Reuters Launches first of its kind Knowledge Graph Feed allowing Financial Services customers to accelerate their AI and Digital Strategies. Thomspon Reuters Press Release. https://www.thomsonreuters.com/en/press-releases/2017/october/thomson-reuters-launches-first-of-its-kind-knowledge-graph-feed.html.

[496] Dominik Tomaszuk, Renzo Angles, Lukasz Szeremeta, Karol Litman, and Diego Cisterna. 2019. Serialization for Property Graphs. In *Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis - 15th International Conference, BDAS 2019, Ustroń, Poland, May 28-31, 2019, Proceedings (Communications in Computer and Information Science)*, Stanislaw Kozielski, Dariusz Mrozek, Pawel Kasprowski, Bozena Malysiak-Mrozek, and Daniel Kostrzewa (Eds.), Vol. 1018. Springer, 57–69.

[497] Gerald Töpper, Magnus Knuth, and Harald Sack. 2012. DBpedia ontology enrichment for inconsistency detection. In *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*, Valentina Presutti and Helena Sofia Pinto (Eds.). ACM Press, 33–40.

[498] Jeffrey Travers and Stanley Milgram. 1969. An Experimental Study of the Small World Problem. *Sociometry* 32, 4 (1969), 425–443.

[499] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. JMLR.org, 2071–2080. http://proceedings.mlr.press/v48/

[500] Jun'ichi Tsujii, James Henderson, and Marius Pasca (Eds.). 2012. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*. The Association for Computational Linguistics. https://www.aclweb.org/anthology/volumes/D12-1/

[501] Ledyard R. Tucker. 1964. The extension of factor analysis to three-dimensional matrices. In *Contributions to Mathematical Psychology*. Holt, Rinehart and Winston, 110–127.

[502] Giovanni Tummarello, Christian Morbidoni, Reto Bachmann-Gmür, and Orri Erling. 2007. RDFSync: Efficient Remote Synchronization of RDF Models, See [1], 537–551.

[503] Octavian Udrea, Diego Reforgiato Recupero, and V. S. Subrahmanian. 2010. Annotated RDF. *ACM Transactions on Computational Logics* 11, 2 (2010), 10:1–10:41. https://doi.org/10.1145/1656242.1656245

[504] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri E. Bal. 2012. WebPIE: A Web-scale Parallel Inference Engine using MapReduce. *Journal of Web Semantics* 10 (2012), 59–75.

[505] Marieke van Erp, Medha Atre, Vanessa López, Kavitha Srinivas, and Carolina Fortuna (Eds.). 2018. *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*. CEUR Workshop Proceedings, Vol. 2180. Sun SITE Central Europe (CEUR). http://ceur-ws.org/Vol-2180

[506] Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction. *Computational Linguistics* 39, 3 (2013), 665–707.

[507] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 12. https://openreview.net/forum?id=rJXMpikCZ

[508] Ruben Verborgh, Miel Vander Sande, Pieter Colpaert, Sam Coppens, Erik Mannens, and Rik Van de Walle. 2014. Web-Scale Querying through Linked Data Fragments. In *Proceedings of the Workshop on Linked Data on the Web, co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014 (CEUR Workshop Proceedings)*, Christian Bizer, Tom Heath, Sören Auer, and Tim Berners-Lee (Eds.), Vol. 1184. Sun SITE Central Europe (CEUR), 10. http://ceur-ws.org/Vol-1184/ldow2014_paper_04.pdf

[509] Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. 2016. Triple Pattern Fragments: A low-cost knowledge graph interface for the Web. *Journal of Web Semantics* 37-38 (2016), 184–206.

[510] Serena Villata, Nicolas Delaforge, Fabien Gandon, and Amelie Gyrard. 2011. An Access Control Model for Linked Data. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops - Confederated International Workshops and Posters: EI2N+NSF ICE, ICSP+INBAST, ISDE, ORM, OTMA, SWWS+MONET+SeDeS, and VADER 2011, Hersonissos, Crete, Greece, October 17-21, 2011. Proceedings (Lecture Notes in Computer Science)*, Robert Meersman, Tharam S. Dillon, and Pilar Herrero (Eds.), Vol. 7046. Springer, 454–463.

[511] Serena Villata and Fabien Gandon. 2012. Licenses Compatibility and Composition in the Web of Data. In *Proceedings of the Third International Workshop on Consuming Linked Data, COLD 2012, Boston, MA, USA, November 12, 2012 (CEUR Workshop Proceedings)*, Juan F. Sequeda, Andreas Harth, and Olaf Hartig (Eds.), Vol. 905. Sun SITE Central Europe (CEUR), 12. http://ceur-ws.org/Vol-905/VillataAndGandon_COLD2012.pdf

[512] Johanna Völker, Daniel Fleischhacker, and Heiner Stuckenschmidt. 2015. Automatic Acquisition of Class Disjointness. *Journal of Web Semantics* 35, P2 (2015), 124–139.

[513] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. 2009. Discovering and Maintaining Links on the Web of Data, See [40], 650–665.

[514] Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl (Eds.). 2018. *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*. Lecture Notes in Computer Science, Vol. 11136. Springer.

[515] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM* 57, 10 (2014), 78–85.

[516] Andreas Wagner, Duc Thanh Tran, Günter Ladwig, Andreas Harth, and Rudi Studer. 2012. Top-$k$ Linked Data Query Processing. In *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings (Lecture Notes in Computer Science)*, Elena Simperl, Philipp Cimiano, Axel Polleres, Óscar Corcho, and Valentina Presutti (Eds.), Vol. 7295. Springer, 56–71.

[517] Claudia Wagner, Eduardo Graells-Garrido, David García, and Filippo Menczer. 2016. Women through the glass ceiling: gender asymmetries in Wikipedia. *EPJ Data Science* 5, 1 (2016), 5.

[518] Toby Walsh (Ed.). 2011. *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. IJCAI/AAAI.

[519] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (Dec. 2017), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

[520] Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge Base Completion Using Embeddings and Rules. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July*

*25-31, 2015*, Qiang Yang and Michael J. Wooldridge (Eds.). IJCAI/AAAI, 1859–1866.

[521] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes, See [67], 1112–1119.

[522] Gerhard Weikum and Martin Theobald. 2010. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, Jan Paredaens and Dirk Van Gucht (Eds.). ACM Press, 65–76.

[523] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge Base Completion via Search-Based Question Answering, See [86], 515–526.

[524] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3 (2016), 9.

[525] William A. Woods. 1975. What's in a Link: Foundations for Semantic Networks. In *Representation and Understanding*, Daniel G. Bobrow and Allan Collins (Eds.). Elsevier, 35–82.

[526] Gong-Qing Wu, Ying He, and Xuegang Hu. 2018. Entity Linking: An Issue to Extract Corresponding Entity With Knowledge Base. *IEEE Access* 6 (2018), 6220–6231.

[527] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *CoRR* abs/1901.00596 (2019), 22. http://arxiv.org/abs/1901.00596

[528] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. 2018. Ontology-Based Data Access: A Survey. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, Jérôme Lang (Ed.). IJCAI/AAAI, 5511–5519.

[529] Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. 2013. GraphX: a resilient distributed graph system on Spark. In *First International Workshop on Graph Data Management Experiences and Systems, GRADES 2013, co-located with SIGMOD/PODS 2013, New York, NY, USA, June 24, 2013*, Peter A. Boncz and Thomas Neumann (Eds.). CWI/ACM, 2.

[530] Reynold S. Xin, Josh Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2013. Shark: SQL and rich analytics at scale. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias (Eds.). ACM Press, 13–24.

[531] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 17. https://openreview.net/forum?id=ryGs6iA5Km

[532] Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling Knowledge Base Gaps for Distant Supervision of Relation Extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*. The Association for Computational Linguistics, 665–670. https://www.aclweb.org/anthology/volumes/P13-2/

[533] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). 12. http://arxiv.org/abs/1412.6575

[534] Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning, See [204], 2319–2328. http://papers.nips.cc/paper/6826-differentiable-learning-of-logical-rules-for-knowledge-base-reasoning

[535] Taha Yasseri, Robert Sumi, András Rung, András Kornai, and János Kertész. 2012. Dynamics of Conflicts in Wikipedia. *PLOS One* 7, 6 (June 2012), 12.

[536] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. 2008. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering* 20, 6 (2008), 796–808.

[537] Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding Methods for Fine Grained Entity Type Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on NaturalLanguage Processing of the Asian Federation of Natural Language Processing,*

*ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Short Papers.* The Association for Computational Linguistics, 291–296. https://www.aclweb.org/anthology/volumes/P15-2/

[538] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. 2016. Quality assessment for Linked Data: A Survey. *Semantic Web Journal* 7, 1 (2016), 63–93.

[539] Lei Zhang. 2002. *Knowledge Graph Theory and Structural Parsing.* Ph.D. Dissertation. University of Twente.

[540] Weizhen Zhang, Han Cao, Fei Hao, Lu Yang, Muhib Ahmad, and Yifei Li. 2019. The Chinese Knowledge Graph on Domain-Tourism. In *Advanced Multimedia and Ubiquitous Engineering, MUE/FutureTech 2019 (Lecture Notes in Electrical Engineering)*, Vol. 590. Springer, 20–27.

[541] Mingbo Zhao, Tommy WS Chow, Zhao Zhang, and Bing Li. 2015. Automatic image annotation via compact graph based semi-supervised learning. *Knowledge-based Systems* 76 (2015), 148–165.

[542] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Ka (Eds.). The Association for Computational Linguistics, 1227–1236. https://www.aclweb.org/anthology/volumes/P1&-1/

[543] Bin Zhou and Jian Pei. 2008. Preserving Privacy in Social Networks Against Neighborhood Attacks, See [10], 506–515.

[544] Bin Zhou and Jian Pei. 2011. The *k*-anonymity and *l*-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems* 28, 1 (2011), 47–77. https://doi.org/10.1007/s10115-010-0311-2

[545] Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring Various Knowledge in Relation Extraction, See [276], 427–434.

[546] Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. 2012. A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data. *Journal of Web Semantics* 12 (mar 2012), 72–95.

[547] Lei Zou, Lei Chen, and M. Tamer Özsu. 2009. K-Automorphism: A General Framework For Privacy Preserving Network Publication. *Proceedings of the VLDB Endowment* 2, 1 (2009), 946–957.

# A BACKGROUND

We now discuss the broader historical context that has paved the way for the modern advent of knowledge graphs, as well as the definitions of the notion of "knowledge graph" that have been proposed both before and after the announcement of the Google Knowledge Graph [459]. We remark that the discussion presented here builds upon (but does not subsume) previous discussion by Ehrlinger and Wöß [136] and by Bergman [36], which we refer to for further details. Though our goal is to be comprehensive, the list of historical references should not be considered exhaustive.

## A.1 Historical Perspective

The lineage of knowledge graphs can be traced back to the origins of diagrammatic forms of knowledge representation: a tradition going back at least to Aristotle (∼350 BC), followed by notions such as Euler circles and Venn diagrams that helped humans to reason through visual insights. Later researchers – particularly Sylvester (1878) [488], Peirce (1878) [379] and Frege (1879) [160] – independently devised formal diagrammatic systems that not only help reasoning, but also codify reasoning; in other words, their goal was to use diagrams as formal systems.

With the advent of digital computers, programs began to be used to perform formal reasoning and to code representations of knowledge. These developments can be traced back to works such as those of Ritchens (1956) [417], Quillian (1963) [401], and Travers and Milgram (1969) [498], which focused on formal representations for natural language, information, and knowledge. These early works faced limitations (at least by modern standards) in terms of the poor computational resources available. From the formal (logical) point of view, a number of influential developments took place in the 70's, including the introduction of *frames* by Minsky (1974) [333], the formalisation of *semantic networks* by Brachman (1977) [62] and Woods (1975) [525], and the proposal of *conceptual graphs* by Sowa (1979) [465]. These works tried to integrate formal logic with diagrammatic representations of knowledge by giving a (more-or-less) formal semantics to graph representations. But as Sowa later wrote in the entry "*Semantic networks*" in the Encyclopedia of Cognitive Science (1987) [466]: "*Woods (1975) and McDermott (1976) observed, the semantic networks themselves have no well-defined semantics. Standard predicate calculus does have a precisely defined, model theoretic semantics; it is adequate for describing mathematical theories with a closed set of axioms. But the real world is messy, incompletely explored, and full of unexpected surprises.*"

From this era of exploration and attempts to define programs to simulate the visual and formal reasoning of humans, the following key notions were established that are still of relevance today:

- knowledge representation through diagrams (specifically graphs) and visual means;
- computational procedures and algorithms to perform formal reasoning;
- combinations of formal (logical) and statistical forms of reasoning;
- relevance of different types of data (e.g., images, sound) as sources of knowledge.

These works on conceptual graphs, semantic networks, and frames were direct predecessors of Description Logics, which aimed to give a well-defined semantics to these earlier notions towards building practical reasoning systems for decidable logics. Description Logics stem from the KL-ONE system proposed by Brachman and Schmolze (1985) [64], and the "*attributive concept descriptions with complements*" language (aka $\mathcal{ALC}$) proposed by Schmidt-Schauß and Smolka (1991) [439]. Description Logics would be further explored in later years (see Section B.5.3), and formed the underpinnings of the Web Ontology Language (OWL) standard [228]. Together with the Resource Description Framework (RDF) [106], OWL would become one of the building blocks of the Semantic Web [39], within which many of the formative ideas and standards underlying knowledge graphs would later be developed, including not only RDF and OWL, but also RDFS [66], SPARQL [207], Linked Data principles [37], Shape Expressions [66, 494], and indeed, many of the other concepts,

standards and techniques discussed in this paper. Most of the open knowledge graphs discussed in Section 10.1 – including BabelNet [351], DBpedia [291], Freebase [51], Wikidata [515], YAGO [481], etc. – have either emerged from the Semantic Web community, or would later adopt its standards.

## A.2 "Knowledge Graphs": Pre 2012

Long before the 2012 announcement of the Google Knowledge Graph, various authors had used the phrase "knowledge graph" in publications stretching back to the 40's, but with unrelated meaning. To the best of our knowledge, the first reference to a "knowledge graph" of relevance to the modern meaning was in a paper by Schneider (1973) [440] in the area of computerised instructional systems for education, where a knowledge graph – in his case a directed graph whose nodes are units of knowledge (concepts) that a student should acquire, and whose edges denote dependencies between such units of knowledge – is used to represent and store an instructional course on a computer. An analogous notion of a "knowledge graph" was used by Marchi and Miguel (1974) [316] to study paths through the knowledge units of an instructional course that yield the highest payoffs for teachers and students in a game-theoretic sense. Around the same time, in a paper on linguistics, Kümmel (1973) [284] describes a numerical representation of knowledge, with "radicals" – referring to some symbol with meaning – forming the nodes of a knowledge graph.

Further authors were to define instantiations of knowledge graphs in the 80's. Rada (1986) [403] defines a knowledge graph in the context of medical expert systems, where domain knowledge is defined as a weighted graph, over which a "gradual" learning process is applied to refine knowledge by making small change to weights. Bakker (1987) [25] defines a knowledge graph with the purpose of accumulatively representing content gleaned from medical and sociological texts, with a focus on causal relationships. Work on knowledge graphs from the same group would continue over the years, with contributions by Stokman and de Vries (1988) [476] further introducing mereological (*part of*) and instantiation (*is a*) relations to the knowledge graph, and thereafter by James [252], Hoede [231], Popping [394], Zhang [539], amongst others, in the decades that followed [367]. The notion of knowledge graph used in such works considered a fixed number of relations. Other authors pursued their own parallel notions of knowledge graphs towards the end of the 80's. Rappaport and Gouyet (1988) [408] describe a user interface for visualising a knowledge-base – composed of facts and rules – using a knowledge graph that connects related elements of the knowledge-base. Srikanth and Jarke (1989) [470] use the notion of a knowledge graph to represent the entities and relations involved in projects, particularly software projects, where partitioning techniques are applied to the knowledge graph to modularise the knowledge required in the project.

Continuing to the 90's, the notion of a "knowledge graph" would again arise in different, seemingly independent settings. De Raedt et al. (1990) [116] propose a knowledge graph as a directed graph composed of a taxonomy of instances being related with weighted edges to a taxonomy of classes; they use symbolic learning to extract such knowledge graphs from examples. Machado and Freitas da Rocha (1990) [311] define a knowledge graph as an acyclic, weighted AND–OR graph,[37] defining fuzzy dependencies that connect observations to hypotheses through intermediary nodes. These knowledge graphs are elicited from domain experts and can be used to generate neural networks for selecting hypotheses from input observations. Knowledge graphs were again later used by Dieng et al. (1992) [124] to represent the results of knowledge acquisition from experts. Shimony et al. (1997) [455] rather define a knowledge graph based on a *Bayesian knowledge base* – i.e., a Bayesian network that permits directed cycles – over which Bayesian inference can be applied. This definition was further built upon in a later work by Santos Jr. and Santos (1999) [436].

---

[37]An AND–OR graph denotes dependency relations, where AND denotes a conjunction of sub-goals on which a goal depends, while OR denotes a disjunction of sub-goals.

Moving to the 00's, Jiang and Ma (2002) [257] introduce the notion of "plan knowledge graphs" where nodes represent goals and edges dependencies between goals, further encoding supporting degrees that can change upon further evidence. Search algorithms are then defined on the graph to determine a plan for a particular goal. Helms and Buijsrogge (2005) [221] propose a knowledge graph to represent the flow of knowledge in an organisation, with nodes representing knowledge actors (creators, sharers, users), edges representing knowledge flow from one actor to another, and edge weights indicating the "velocity" (delay of flow) and "viscosity" (the depth of knowledge transferred). Graph algorithms are then proposed to find bottlenecks in knowledge flow. Kasneci et al. (2008) [264] propose a search engine for knowledge graphs, defined to be weighted directed edge-labelled graphs, where weights denote confidence scores based on the centrality of source documents from which the edge/relation was extracted. From the same group, Elbassuoni et al. (2009) [137] adopt a similar notion of a knowledge graph, adding edge attributes to include keywords from the source, a count of supporting sources, etc., showing how the graph can be queried. Coursey and Mihalcea (2009) [102] construct a knowledge graph from Wikipedia, where nodes represent Wikipedia articles and categories, while edges represent the proximity of nodes. Subsequently, given an input text, entity linking and centrality measures are applied over the knowledge graph to determine relevant Wikipedia categories for the text.

Concluding with the 10's (prior to 2012), Pechsiri and Piriyakul (2010) [378] use knowledge graphs to capture "explanation knowledge" – the knowledge of why something is the way it is – by representing events as nodes and causal relationships as edges, claiming that this graphical notation offers more intuitive explanations to users; their work focuses on extracting such knowledge graphs from text. Corby and Faron-Zucker (2010) [97] use the phrase "knowledge graph" in a general way to denote any graph encoding knowledge, proposing an abstract machine for querying such graphs.

Other phrases were used to represent similar notions by other authors, including "information graphs" [284], "information networks" [484], "knowledge networks" [87], as well as "semantic networks" [62, 351, 525] and "conceptual graphs" [465], as mentioned previously. Here we exclusively considered works that (happen to) use the phrase "knowledge graph" prior to Google's announcement of their knowledge graph in 2012, where we see that many works had independently coined this phrase for different purposes. Similar to the current practice, all of the works of this period consider a knowledge graph to be formed of a set of nodes denoting entities of interest and a set of edges denoting relations between those entities, with different entities and relations being considered in different works. Some works add extra elements to these knowledge graphs, such as edge weights, edge labels, or other meta-data [137]. Other trends include knowledge acquisition from experts [124, 311, 403] and knowledge extraction from text [25, 231, 252, 476], combinations of symbolic and inductive methods [116, 311, 436, 455], as well as the use of rules [408], ontologies [231], graph analytics [221, 264, 470], learning [116, 403, 436, 455], and so forth. Later papers (2008–2010) by Kasneci et al. [264], Elbassuoni et al. [137], Coursey and Mihalcea [102] and Corby and Faron-Zucker [97] introduce notions of knowledge graph similar to current practice.

However, some trends are not reflected in current practice. Of particular note, quite a lot of the knowledge graphs defined in this period consider edges as denoting a form of dependence or causality, where $x \longrightarrow y$ may denote that $x$ is a prerequisite for $y$ [257, 316, 440] or that $x$ leads to $y$ [25, 257, 311, 403, 408, 455]. In some cases AND–OR graphs are used to denote conjunctions or disjunctions of such relations [311], while in other cases edges are weighted to assign a belief to a relation [257, 311, 403]. In addition, papers from 1970–2000 tend to have worked with small graphs, which contrasts with modern practice where knowledge graphs can reach scales of millions or billions of nodes [365]: during this period, computational resources were more limited [440], and fewer sources of structured data were readily available meaning that the knowledge graphs were often sourced solely from human experts [124, 311, 403] or from text [25, 231, 252, 476].

### A.3 "Knowledge Graphs": 2012 Onwards

Google Knowledge Graph was announced in 2012 [459]. This initial announcement was targeted at a broad audience, mainly motivating the knowledge graph and describing applications that it would enable, where the knowledge graph itself is described as "*[a graph] that understands real-world entities and their relationships to one another*" [459]. Mentions of "knowledge graphs" quickly gained momentum in the research literature from that point. As noted by Bergman [36], this announcement by Google was a watershed moment in terms of adopting the phrase "knowledge graph". However, given the informal nature of the announcement, a technical definition was lacking [53, 136].

Given that knowledge graphs were gaining more and more attention in the academic literature, formal definitions were becoming a necessity in order to precisely characterise what they were, how they were structured, how they could be used, etc., and more generally to facilitate their study in a precise manner. We can determine four general categories of definitions.

*Category I*: The first category simply defines the knowledge graph as a graph where nodes represent entities, and edges represent relationships between those entities. Often a directed edge-labelled graph is assumed (or analogously, a set of binary relations, or a set of triples). This simple and direct definition was popularised by some of the seminal papers on knowledge graph embeddings [298, 521] (2014–2015), being sufficient to represent the data structure upon which these embeddings would operate. As reflected in the survey by Wang et al. [519], the multitude of works that would follow on knowledge graph embeddings have continued to use this definition. Though simple, the *Category I* definition raises some doubts: How is a knowledge graph different from a graph (database)? Where does knowledge come into play?

*Category II*: A second common definition goes as follows: "*a knowledge graph is a graph-structured knowledge base*", where, to the best of our knowledge, the earliest usages of this definition in the academic literature were by Nickel et al. [362] (2016) and Seufert et al. (2016) [449] (interestingly in the formal notation of these initial papers, a knowledge graph is defined analogously to a directed edge-labelled graph). Such a definition raises the question: what, then is a "knowledge base"? The phrase "knowledge base" was popularised in the 70's (possibly earlier) in the context of rule-based expert systems [69], and later were used in the context of ontologies and other logical formalisms [64]. The follow-up question then is whether or not one can have a knowledge base (graph-structured or not) without a logical formalism while staying true to the original definitions. Looking in further detail, similar ambiguities have also existed regarding the definition of a "knowledge base" (KB). Of note: Brachman and Levesque (1986) [63] – reporting after a workshop on this issue – state that "*if we ask what the KB tells us about the world, we are asking about its Knowledge Level*".

*Category III*: The third category of definitions define additional, technical characteristics that a "knowledge graph" should comply with, where we list some prominent definitions.

- In an influential survey on knowledge graph refinement, Paulheim [375] lists four criteria that characterise the knowledge graphs considered for the paper. Specifically, that a knowledge graph "*mainly describes real world entities and their interrelations, organized in a graph; defines possible classes and relations of entities in a schema; allows for potentially interrelating arbitrary entities with each other; covers various topical domains*"; he thus rules out ontologies without instances (e.g., DOLCE) and graphs of word senses (e.g., WordNet) as not meeting the first two criteria, while relational databases do not meet the third criterion (due to schema restrictions), and domain-specific graphs (e.g., Geonames) are considered to not meet the fourth criterion; this leaves graphs such as DBpedia, YAGO, Freebase, etc.

- Ehrlinger and Wöß [136] also review definitions of "knowledge graph", where they criticise the *Category II* definitions based on the argument that knowledge bases are often synonymous with ontologies[38], while knowledge graphs are not; they further criticise Google for calling its knowledge graph a "knowledge base". After reviewing prior definitions of terms such as "knowledge base", "ontology", and "knowledge graph", they propose their definition: "*A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge*". In the subsequent discussion, they remark that a knowledge graph is distinguished from an ontology (considered synonymous with a knowledge base) by the provision of reasoning capabilities.

- One of the most detailed technical definitions for a "knowledge graph" is provided by Bellomarini et al. [32], who state: "*A knowledge graph is a semi-structured data model characterized by three components: (i) a ground extensional component, that is, a set of relational constructs for schema and data (which can be effectively modeled as graphs or generalizations thereof); (ii) an intensional component, that is, a set of inference rules over the constructs of the ground extensional component; (iii) a derived extensional component that can be produced as the result of the application of the inference rules over the ground extensional component (with the so-called "reasoning" process).*" They remark that ontologies and rules represent analogous structures, and that a knowledge graph is then a knowledge base extended with reasoning along similar lines to the definition provided by Ehrlinger and Wöß [136].

We refer to Bergman [36] for a list of further definitions that fit this category. While having a specific, technical definition for knowledge graphs provides a more solid grounding for their study, as Bergman [36] remarks, many of these definitions do not seem to fit the current practice of knowledge graphs. For example, it is not clear which of these definitions the Google Knowledge Graph itself – responsible for popularising the idea – would meet (if any). Furthermore, many of the criteria proposed by such definitions are orthogonal to the multitude of works in the area of knowledge graph embeddings [519].

*Category IV*: While the previous three categories involve (sometimes conflicting) intensional definitions, the fourth category adopts an extensional definition of knowledge graphs, defining them by example. Knowledge graphs are then characterised by examples such as DBpedia, Google's Knowledge Graph, Freebase, YAGO, amongst others [53]. Arguably this category sidesteps the issue of defining a knowledge graph, rather than providing such a definition.

These categories refer to definitions that have appeared in the academic literature. In terms of enterprise knowledge graphs, an important reference is the paper of Noy et al. [365], which has been co-authored by leaders of knowledge graph projects from eBay, Facebook, Google, IBM, and Microsoft, and thus can be seen as representing a form of consensus amongst these companies on what is a knowledge graph — a concept these companies have played a key role in popularising. Specifically this paper states that "*a knowledge graph describes objects of interest and connections between them*", and goes on to state that "*many practical implementations impose constraints on the links in knowledge graphs by defining a schema or ontology*". They later add "*Knowledge graphs and similar structures usually provide a shared substrate of knowledge within an organization, allowing different products and applications to use similar vocabulary and to reuse definitions and descriptions that others create. Furthermore, they usually provide a compact formal representation that developers can use to infer new facts and build up the knowledge*". We interpret this definition as corresponding to *Category I*, but further acknowledging that while not a necessary condition for a knowledge graph, ontologies and formal representations *usually* play a key role. The definition we provide at the outset of the paper is largely compatible with that of Noy et al. [365].

---

[38]Prior definitions of an ontology – such as by Guarino et al. [197] – would seem to contradict this conclusion.

# B FORMAL DEFINITIONS

In order to keep the discussion as accessible as possible, the body of the paper uses example-driven explanations of the main concepts and techniques associated with knowledge graphs. In this section, we complement the discussion of the paper with formal definitions.

## B.1 Data Graph Models

We define the graph data models in line with previous conventions (e.g., [14]). While different types of constants may be used in different models (e.g., RDF allows IRIs and literals), these definitions use a single (countably) infinite set of constants denoted **Con**. (We thus also abstract away from issues that are not exigent for the current introductory discussion, such as the existential semantics of blank nodes in RDF [235], $D$-entailment over literals [213], positional restrictions [106], etc.)

*B.1.1 Directed edge-labelled graph.* We first provide definitions for a directed edge-labelled graph.

*Definition B.1 (Directed edge-labelled graph).* A *directed edge-labelled graph* is a tuple $G := (V, E, L)$, where $V \subseteq$ **Con** is a set of nodes, $L \subseteq$ **Con** is a set of edge labels, and $E \subseteq V \times L \times V$ is a set of edges.

*Example B.2.* In reference to Figure 1, the set of nodes $V$ has 15 elements, including `Arica`, `EID16`, etc. The set of edges $E$ has 23 triples, including (`Arica,flight,Santiago`). Bidirectional edges are represented with two edges. The set of edge labels $L$ has 8 elements, including `start`, `flight`, etc.

Definition B.1 does not state that $V$ and $L$ are disjoint: though not present in the example, a node can also serve as an edge-label. The definition also permits that nodes and edge labels can be present without any associated edge. Either restriction could be explicitly stated – if necessary – in a particular application while still conforming to a directed edge-labelled graph.

In some of the definitions that follow, for ease of presentation, we may treat a set of (directed labelled) edges $E \subseteq V \times L \times V$ as a directed edge-labelled graph $(V, E, L)$, in which case we refer to the graph induced by $E$ assuming that $V$ and $L$ contain all and only those nodes and edge labels, respectively, used in $E$. We may similarly apply set operators on directed edge-labelled graphs, which should be interpreted as applying to their sets of edges; for example, given $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$, by $G_1 \cup G_2$ we refer to the directed edge-labelled graph induced by $E_1 \cup E_2$.

*B.1.2 Graph dataset.* Next we define a graph dataset.

*Definition B.3 (Graph dataset).* A *named graph* is a pair $(n, G)$ where $G$ is a directed edge-labelled graph, and $n \in$ **Con** is a graph name. A *graph dataset* is a pair $D := (G_D, N)$ where $G_D$ is a directed edge-labelled graph called the *default graph* and $N$ is either the empty set or a set of named graphs $\{(n_1, G_1), \ldots (n_k, G_k)\}$ ($k > 0$) such that $n_i = n_j$ if and only if $i = j$ ($1 \leq i \leq k$, $1 \leq j \leq k$).

*Example B.4.* In reference to Figure 2, the graph dataset $D$ consists of two named graphs and a default graph. The default graph does not have a name associated with it. The two graph names are `Events` and `Routes`; these are also used as nodes in the default graph.

Though Definition B.3 is based on directed edge-labelled graphs, the notion of a graph dataset trivially generalises for managing other forms of graphs (such as property graphs). An RDF dataset is an example of a graph dataset model standardised by the W3C [106] where each graph is an RDF graph, and graph names can be blank nodes or IRIs. The definition of a graph dataset permits that graph names can be used as nodes or edge labels in graphs; this is also the case for RDF datasets.

*B.1.3 Property graph.* Finally, we define a property graph.

*Definition B.5 (Property graph).* A *property graph* is a tuple $G := (V, E, L, P, U, e, l, p)$, where $V \subseteq$ **Con** is a set of node ids, $E \subseteq$ **Con** is a set of edge ids, $L \subseteq$ **Con** is a set of labels, $P \subseteq$ **Con** is a

set of properties, $U \subseteq \mathbf{Con}$ is a set of values, $e : E \rightarrow V \times V$ maps an edge id to a pair of node ids, $l : V \cup E \rightarrow 2^L$ maps a node or edge id to a set of labels, and $p : V \cup E \rightarrow 2^{P \times U}$ maps a node or edge id to a set of property–value pairs.

*Example B.6.* Returning to Figure 4:

- the set $V$ contains Santiago and Arica;
- the set $E$ contains E1 and E2;
- the set $L$ contains City and flight;
- the set $P$ contains country, capital, company and dist;
- the set $U$ contains Chile, true, false, Sky Airlines, LAN Airlines, and 2000 km;
- the mapping $e$ gives, e.g, $e(\text{E1}) = (\text{Santiago}, \text{Arica})$;
- the mapping $l$ gives, e.g., $l(\text{E1}) = \{\text{flight}\}$ and $l(\text{Santiago}) = \{\text{city}\}$;
- the mapping $p$ gives, e.g., $p(\text{E1}) = \{(\text{company}, \text{Sky Airlines}), \dots, (\text{dist}, 2000 \text{ km})\}$ and $p(\text{Santiago}) = \{(\text{country}, \text{Chile}), \dots, (\text{capital}, \text{true})\}$.

Definition B.5 does not require that the sets $V$, $E$, $L$, $P$ or $U$ to be (pairwise) disjoint: we allow, for example, that values are also nodes. Unlike some previous definitions [14], here we allow a node or edge to have several values for a given property. In practice, systems like Neo4j [332] may rather support this by allowing an array of values. We view such variations as syntactic.

## B.2 Querying

Here we formalise foundational notions relating to queries over graphs, starting with graph patterns, to which we later add relational-style operators and path expressions.

*B.2.1 Graph patterns.* We formalise the notions of graph patterns first for directed edge-labelled graphs, and subsequently for property graphs [14]. For these definitions, we introduce a countably infinite set of *variables* **Var** ranging over (but disjoint from: $\mathbf{Con} \cap \mathbf{Var} = \emptyset$) the set of constants. We refer generically to constants and variables as *terms*, denoted and defined as $\mathbf{Term} = \mathbf{Con} \cup \mathbf{Var}$.

*Definition B.7 (Directed edge-labelled graph pattern).* We define a *directed edge-labelled graph pattern* as a tuple $Q = (V', E', L')$, where $V' \subseteq \mathbf{Term}$ is a set of node terms, $L' \subseteq \mathbf{Term}$ is a set of edge terms, and $E' \subseteq V' \times L' \times V'$ is a set of edges (triple patterns).

*Example B.8.* Returning to the graph pattern of Figure 5:

- the set $V'$ contains the constant Food Festival and variables ?event, ?ven1 and ?ven2;
- the set $L'$ contains the constants type and venue;
- the set $E'$ contains four edges, including (?event, type, Food Festival), etc.

A property graph pattern is defined analogously, allowing variables in any position.

*Definition B.9 (Property graph pattern).* We define a *property graph pattern* as a tuple $Q = (V', E', L', P', U', e', l', p')$, where $V' \subseteq \mathbf{Term}$ is a set of node id terms, $E' \subseteq \mathbf{Term}$ is a set of edge id terms, $L' \subseteq \mathbf{Term}$ is a set of label terms, $P' \subseteq \mathbf{Term}$ is a set of property terms, $U' \subseteq \mathbf{Term}$ is a set of value terms, $e' : E' \rightarrow V' \times V'$ maps an edge id term to a pair of node id terms, $l' : V' \cup E' \rightarrow 2^{L'}$ maps a node or edge id term to a set of label terms, and $p' : V' \cup E' \rightarrow 2^{P' \times U'}$ maps a node or edge id term to a set of pairs of property–value terms.

Towards defining the results of evaluating a graph pattern, we first define a partial mapping $\mu : \mathbf{Var} \rightarrow \mathbf{Con}$ from variables to constants, whose *domain* (the set of variables for which it is defined) is denoted by $\text{dom}(\mu)$. Given a graph pattern $Q$, let $\mathbf{Var}(Q)$ denote the set of all variables appearing in (some recursively nested element of) $Q$. Abusing notation, we denote by $\mu(Q)$ the

image of $Q$ under $\mu$, meaning that any variable $v \in \mathbf{Var}(Q) \cap \mathrm{dom}(\mu)$ is replaced in $Q$ by $\mu(v)$. Observe that when $\mathbf{Var}(Q) \subseteq \mathrm{dom}(\mu)$, then $\mu(Q)$ is a data graph (in the corresponding model of $Q$).

Next, we define the notion of containment between data graphs. For two directed edge-labelled graph pattern $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$, we say that $G_1$ is a *sub-graph* of $G_2$, denoted $G_1 \subseteq G_2$, if and only if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, and $L_1 \subseteq L_2$.[39] Conversely, in property graphs, nodes can often be defined without edges. For two property graphs $G_1 = (V_1, E_1, L_1, P_1, U_1, e_1, l_1, p_1)$ and $G_2 = (V_2, E_2, L_2, P_2, U_2, e_2, l_2, p_2)$, we say that $G_1$ is a *sub-graph* of $G_2$, denoted $G_1 \subseteq G_2$, if and only if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $L_1 \subseteq L_2$, $P_1 \subseteq P_2$, $U_1 \subseteq U_2$, for all $x \in E_1$ it holds that $e_1(x) = e_2(x)$, and for all $y \in E_1 \cup V_1$ it holds that $l_1(y) \subseteq l_2(y)$ and $p_1(y) \subseteq p_2(y)$.

We are now ready to define the evaluation of a graph pattern.

*Definition B.10 (Evaluation of a graph pattern).* Let $Q$ be a graph pattern and let $G$ be a data graph. We then define the *evaluation of graph pattern $Q$ over the data graph $G$*, denoted $Q(G)$, to be the set of mappings $\{\mu \mid \mu(Q) \subseteq G \text{ and } \mathrm{dom}(\mu) = \mathbf{Var}(Q)\}$.

*Example B.11.* Figure 5 enumerates all of the mappings given by the evaluation of the depicted graph pattern over the data graph of Figure 1. Each non-header row indicates a mapping $\mu$.

The final results of evaluating a graph pattern may then vary depending on the choice of semantics: the results under *homomorphism-based semantics* are defined as $Q(G)$. Conversely, under *isomorphism-based* semantics, mappings that send two edge variables to the same constant and/or mappings that send two node variables to the same constant may be excluded from the results. Henceforth we assume the more general *homomorphism-based semantics*.

*B.2.2 Complex graph patterns.* We now define complex graph patterns.

*Definition B.12 (Complex graph pattern).* *Complex graph patterns* are defined recursively:

- If $Q$ is a graph pattern, then $Q$ is a *complex graph pattern*.
- If $Q$ is a complex graph pattern, and $\mathcal{V} \subseteq \mathbf{Var}(Q)$, then $\pi_{\mathcal{V}}(Q)$ is a *complex graph pattern*.
- If $Q$ is a complex graph pattern, and $R$ is a selection condition with boolean and equality connectives $(\wedge, \vee, \neg, =)$, then $\sigma_R(Q)$ is a *complex graph pattern*.
- If $Q_1$ and $Q_2$ are complex graph patterns, then $Q_1 \bowtie Q_2$, $Q_1 \cup Q_2$ and $Q_1 - Q_2$ are also *complex graph patterns*.

Next we define the evaluation of complex graph patterns. First, given a mapping $\mu$, for a set of variables $\mathcal{V} \subseteq \mathbf{Var}$ let $\mu[\mathcal{V}]$ denote the mapping $\mu'$ such that $\mathrm{dom}(\mu') = \mathrm{dom}(\mu) \cap \mathcal{V}$ and $\mu(v) = \mu'(v)$ for all $v \in \mathrm{dom}(\mu')$ (in other words, $\mu[\mathcal{V}]$ projects the variables $\mathcal{V}$ from $\mu$). Furthermore, letting $R$ denote a boolean selection condition and $\mu$ a mapping, by $R \models \mu$ we denote that $\mu$ satisfies the boolean condition. Finally, we define two mappings $\mu_1$ and $\mu_2$ to be *compatible*, denoted $\mu_1 \sim \mu_2$, if and only if $\mu_1(v) = \mu_2(v)$ for all $v \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$ (in other words, they map all common variables to the same constant). We are now ready to provide the definition.

---

[39]Given, for example, $G_1 = (\{a\}, \{(a, b, a)\}, \{b, c\})$ and $G_2 = (\{a, c\}, \{(a, b, a)\}, \{b\})$, we remark that $G_1 \nsubseteq G_2$ and $G_2 \nsubseteq G_1$: the former has a label not used on an edge while the latter has a node without an incident edge. In concrete data models like RDF where such cases of nodes or labels without edges cannot occur, the sub-graph relation $G_1 \subseteq G_2$ holds if and only if $E_1 \subseteq E_2$ holds.

*Definition B.13 (Complex graph pattern evaluation).* Given a complex graph pattern $Q$, if $Q$ is a graph pattern, then $Q(G)$ is defined per Definition B.10. Otherwise:

$$\pi_{\mathcal{V}}(Q)(G) := \{\mu[\mathcal{V}] \mid \mu \in Q(G)\}$$

$$\sigma_R(Q)(G) := \{\mu \mid \mu \in Q(G) \text{ and } R \models \mu\}$$

$$Q_1 \bowtie Q_2(G) := \{\mu_1 \cup \mu_2 \mid \mu_1 \in Q_2(G) \text{ and } \mu_2 \in Q_1(G) \text{ and } \mu_1 \sim \mu_2\}$$

$$Q_1 \cup Q_2(G) := \{\mu \mid \mu \in Q_1(G) \text{ or } \mu \in Q_2(G)\}$$

$$Q_1 - Q_2(G) := \{\mu \mid \mu \in Q_1(G) \text{ and } \mu \notin Q_2(G)\}$$

Based on these query operators, we can also define some additional syntactic operators, such as the *anti-join* ($\triangleright$, aka *not exists*), or the *left-join* ($\bowtie$, aka *optional*):

$$Q_1 \triangleright Q_2(G) := Q_1(G) - \pi_{\mathbf{Var}(Q_1)}(Q_1(G) \bowtie Q_2(G))$$

$$Q_1 \bowtie Q_2(G) := (Q_1(G) \bowtie Q_2(G)) \cup (Q_1(G) \triangleright Q_2(G))$$

We call these operators *syntactic* as they do not add expressivity to the query language.

*Example B.14.* Figure 7 illustrates a complex graph pattern and its evaluation.

*B.2.3 Navigational graph patterns.* We first define path expressions and regular path queries.

*Definition B.15 (Path expression).* A constant (edge label) $c$ is a *path expression*. Furthermore:
- If $r$ is a path expression, then $r^-$ (*inverse*) and $r^*$ (*Kleene star*) are *path expressions*.
- If $r_1$ and $r_2$ are path expressions, then $r_1 \cdot r_2$ (*concatenation*) and $r_1 \mid r_2$ (*disjunction*) are *path expressions*.

We now define the evaluation of a path expression under the SPARQL 1.1-style semantics whereby the endpoints (pairs of start and end nodes) of the path are returned [207].

*Definition B.16 (Path expression evaluation (directed edge-labelled graph)).* Given a directed edge-labelled graph $G = (V, E, L)$ and a path expression $r$, we define the *evaluation of $r$ over $G$*, denoted $r[G]$, as follows:

$$r[G] := \{(u, v) \mid (u, r, v) \in E\} \text{ (for } r \in \mathbf{Con})$$

$$r^-[G] := \{(u, v) \mid (v, u) \in r[G]\}$$

$$r_1 \mid r_2[G] := r_1[G] \cup r_2[G]$$

$$r_1 \cdot r_2[G] := \{(u, v) \mid \exists w \in V : (u, w) \in r_1[G] \text{ and } (w, v) \in r_2[G]\}$$

$$r^*[G] := V \cup \bigcup_{n \in \mathbb{N}^+} r^n[G]$$

where by $r^n$ we denote the $n^{\text{th}}$-concatenation of $r$ (e.g., $r^3 = r \cdot r \cdot r$).

The evaluation of a path expression on a property graph $G = (V, E, L, P, U, e, l, p)$ can be defined analogously by adapting the first definition (in the case that $r \in \mathbf{Con}$) as follows:

$$r[G] := \{(u, v) \mid \exists x \in E : e(x) = (u, v) \text{ and } l(e) = r\}.$$

The rest of the definitions then remain unchanged.

Query languages may support additional operators, some of which are syntactic (e.g., $r^+$ is sometimes used for one-or-more, but can be rewritten as $r \cdot r^*$), while others may add expressivity such as the case of SPARQL [207], which allows a limited form of negation in expressions (e.g., $!r$, with $r$ being a constant or the inverse of a constant, matching any path not labelled $r$).

Next we define a regular path query and its evaluation.

*Definition B.17 (Regular path query).* A *regular path query* is a triple $(x, r, y)$ where $x, y \in$ **Con** $\cup$ **Var** and $r$ is a path expression.

*Definition B.18 (Regular path query evaluation).* Let $G$ denote a directed edge-labelled graph, $c$, $c_1, c_2 \in$ **Con** denote constants and $z, z_1, z_2 \in$ **Var** denote variables. Then the *evaluation of a regular path query* is defined as follows:

$$(c_1, r, c_2)(G) := \{\mu_\emptyset \mid (c_1, c_2) \in r[G]\}$$
$$(c, r, z)(G) := \{\mu \mid \text{dom}(\mu) = \{z\} \text{ and } (c, \mu(z)) \in r[G]\}$$
$$(z, r, c)(G) := \{\mu \mid \text{dom}(\mu) = \{z\} \text{ and } (\mu(z), c) \in r[G]\}$$
$$(z_1, r, z_2)(G) := \{\mu \mid \text{dom}(\mu) = \{z_1, z_2\} \text{ and } (\mu(z_1), \mu(z_2)) \in r[G]\}$$

where $\mu_\emptyset$ denotes the empty mapping such that $\text{dom}(\mu) = \emptyset$ (the join identity).

*Definition B.19 (Navigational graph pattern).* If $Q$ is a graph pattern, then $Q$ is a *navigational graph pattern*. Furthermore, if $Q$ is a navigational graph pattern and $(x, r, y)$ is a regular path query, then $Q \bowtie (x, r, y)$ is a *navigational graph pattern*.

The definition of the evaluation of a navigational graph pattern then follows from the previous definition of a join and the corresponding definition of the evaluation of a regular path query (for a directed edge-labelled graph or a property graph, respectively). Likewise, *complex navigational graph patterns* – and their evaluation – are defined by extending this definition in the natural way with the same operators from Definition B.12 following the same semantics seen in Definition B.13.

## B.3 Schema

Here we formalise notions relating to schemata for graphs. Though we present definitions for directed edge-labelled graphs – which allows for more succinct presentation – the same concepts can be applied to property graphs and other graph models.

*B.3.1 Semantic schema.* We provide definitions that generalise semantic schemata in Appendix B.5.

*B.3.2 Validating schema.* We define shapes following conventions used by Labra Gayo et al. [286].

*Definition B.20 (Shape).* A *shape* $\phi$ is defined as:

| $\phi$ | ::= | $\top$ | true |
|--------|-----|--------|------|
| | \| | $\Delta_N$ | node belongs to the set of nodes $N$ |
| | \| | $\Psi_{\text{cond}}$ | node satisfies the boolean condition cond |
| | \| | $\phi_1 \wedge \phi_2$ | conjunction of shape $\phi_1$ and shape $\phi_2$ |
| | \| | $\neg\phi$ | negation of shape $\phi$ |
| | \| | $@s$ | reference to shape with label $s$ |
| | \| | $\xrightarrow{p} \phi\{min, max\}$ | between *min* and *max* outward edges (inclusive) with label $p$ to nodes satisfying shape $\phi$ |

where $min \in \mathbb{N}_{(0)}$, $max \in \mathbb{N}_{(0)} \cup \{*\}$, with "$*$" indicating unbounded.

*Definition B.21 (Shapes schema).* A *shapes schema* is defined as a tuple $\Sigma := (\Phi, S, \lambda)$ where $\Phi$ is a set of shapes, $S$ is a set of shape labels, and $\lambda : S \to \Phi$ is a total function from labels to shapes.

*Example B.22.* The shapes schema from Figure 12 can be expressed as:

$$\boxed{\text{Event}} \quad \mapsto \quad \xrightarrow{\text{name}} \Delta_{\text{string}}\{1,*\} \wedge \xrightarrow{\text{start}} \Delta_{\text{dateTime}}\{1,1\} \wedge \xrightarrow{\text{end}} \Delta_{\text{dateTime}}\{1,1\}$$
$$\wedge \xrightarrow{\text{type}} \top\{1,*\} \wedge \xrightarrow{\text{venue}} @\boxed{\text{Venue}}\{1,*\}$$

$$\boxed{\text{Venue}} \quad \mapsto \quad @\boxed{\text{Place}} \wedge \xrightarrow{\text{indoor}} \Delta_{\text{boolean}}\{0,1\} \wedge \xrightarrow{\text{city}} @\boxed{\text{City}}\{0,1\}$$

$$\boxed{\text{City}} \quad \mapsto \quad @\boxed{\text{Place}} \wedge \xrightarrow{\text{population}} (\Delta_{\text{int}} \wedge \Psi_{>5000})\{0,1\}$$

$$\boxed{\text{Place}} \quad \mapsto \quad \xrightarrow{\text{lat}} \Delta_{\text{float}}\{0,1\} \wedge \xrightarrow{\text{long}} \Delta_{\text{float}}\{0,1\}$$
$$\wedge \xrightarrow{\text{flight}} @\boxed{\text{Place}}\{0,*\} \wedge \xrightarrow{\text{bus}} @\boxed{\text{Place}}\{0,*\}$$

In a shapes schema, shapes may refer to other shapes, giving rise to a graph that is sometimes known as the *shapes graph* [278]. Figure 12 illustrates a shapes graph of this form.

The semantics of a shape $\phi$ is defined in terms of the evaluation of $\phi$ over the nodes of a graph $G = (V, E, L)$ with respect to a shapes map $\sigma$ associating nodes and shape labels that apply to them.

*Definition B.23 (Shapes map).* Given a graph $G = (V, E, L)$ and a schema $\Sigma = (\Phi, S, \lambda)$, a *shapes map* is a (partial) mapping $\sigma : V \times S \to \{0, 1\}$.

The precise semantics of a shape then depends on whether or not $\sigma$ is a total or partial mapping: whether or not it is defined for every value in $V \times S$. In this paper, we present the semantics for the more straightforward case where $\sigma$ is assumed to be a total shapes map.

*Definition B.24 (Shape evaluation).* Given a shapes schema $\Sigma := (\Phi, S, \lambda)$, the semantics of a shape $\phi \in \Phi$ is defined in terms of a *shape evaluation function* $[\phi]^{G,v,\sigma} \in \{0, 1\}$, for a graph $G = (V, E, L)$, a node $v \in V$ and a total shapes map $\sigma$, such that:

$$
\begin{array}{lcl}
[\top]^{G,v,\sigma} & = & 1 \\
[\Delta_N]^{G,v,\sigma} & = & 1 \text{ iff } v \in N \\
[\Psi_{\text{cond}}]^{G,v,\sigma} & = & 1 \text{ iff } \text{cond}(v) \text{ is true} \\
[\phi_1 \wedge \phi_2]^{G,v,\sigma} & = & \min\{[\phi_1]^{G,v,\sigma}, [\phi_2]^{G,v,\sigma}\} \\
[\neg\phi]^{G,v,\sigma} & = & 1 - [\phi]^{G,v,\sigma} \\
[@s]^{G,v,\sigma} & = & 1 \text{ iff } \sigma(v, s) = 1 \\
[\xrightarrow{p} \phi\{min, max\}]^{G,v,\sigma} & = & 1 \text{ iff } min \le |\{(v, p, u) \in E \mid [\phi]^{G,u,\sigma} = 1\}| \le max
\end{array}
$$

If $[\phi]^{G,v,\sigma} = 1$, then $v$ is said to *satisfy* $\phi$ in $G$ under $\sigma$.

Typically for the purposes of validating a graph with respect to a shapes schema, a *target* is defined that requires certain nodes to satisfy certain shapes.

*Definition B.25 (Shapes target).* Given a directed edge-labelled graph $G = (V, E, L)$ and a shapes schema $\Sigma = (\Phi, S, \lambda)$, a *shapes target* $T$ is a set of pairs of nodes and shape labels: $T \subseteq V \times S$.

The nodes that a shape targets can be selected manually, based on the type(s) of the nodes, based on the results of a graph query, etc. [100, 286].

Lastly, we can define the notion of a valid graph under a given shapes schema and target.

*Definition B.26 (Valid graph).* Given a shapes schema $\Sigma = (\Phi, S, \lambda)$, a graph $G = (V, E, L)$, and a shapes target $T$, we say that $G$ *is valid under* $\Sigma$ *and* $T$ if and only if there exists a shapes map $\sigma$ such that, for all $s \in S$ and $v \in V$ it holds that $\sigma(v, s) = [\lambda(s)]^{G,v,\sigma}$, and $(v, s) \in T$ implies $\sigma(v, s) = 1$.

*Example B.27.* Taking the graph $G$ from Figure 1 and the shapes schema $\Sigma$ from Figure 12, first assume an empty shapes target $T = \{\}$. If we consider a shapes map where (for example) $\sigma(\boxed{\text{EID15}}, \boxed{\text{Event}}) = 1$, $\sigma(\boxed{\text{Santa Lucía}}, \boxed{\text{Venue}}) = 1$, $\sigma(\boxed{\text{Santa Lucía}}, \boxed{\text{Place}}) = 1$, etc., but where $\sigma(\boxed{\text{EID16}}, \boxed{\text{Event}}) = 0$ (as it does not have the required values for start and end), etc., then we

see that $G$ is valid under $\Sigma$ and $T$. However, if we were to define a shapes target $T$ to ensure that the `EVENT` shape targets `EID15` and `EID16` – i.e., to define $T$ such that $\{(($ `EID15` $,$ `EVENT` $), ($ `EID16` $,$ `EVENT` $))\} \subseteq T$ – then the graph would no longer be valid under $\Sigma$ and $T$ since `EID16` does not satisfy `EVENT`.

The semantics we present here assumes that each node in the graph either satisfies or does not satisfy each shape labelled by the schema. More complex semantics – for example, based on Kleene's three-valued logic [100, 286] – have been proposed that support partial shapes maps, where the satisfaction of some nodes for some shapes can be left undefined. Shapes languages in practice may support other forms of constraints, such as counting on paths [278]. In terms of implementing validation with respect to shapes, work has been done on translating constraints into sets of graph queries, whose results are input to a SAT solver for recursive cases [99].

### B.3.3 Emergent schema.
Emergent schemata are often based on the notion of a quotient graph.

*Definition B.28 (Quotient graph).* Given a directed-edge labelled graph $G = (V, E, L)$, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, L)$ is a *quotient graph* of $G$ if and only if:
- $\mathcal{V}$ is a partition of $V$ without the empty set, i.e., $\mathcal{V} \subseteq (2^V - \emptyset)$, $V = \bigcup_{U \in \mathcal{V}} U$, and for all $U \in \mathcal{V}$, $W \in \mathcal{V}$, it holds that $U = W$ or $U \cap W = \emptyset$; *and*
- $\mathcal{E} = \{(U, l, W) \mid U \in \mathcal{V}, W \in \mathcal{V}$ and there exist $u \in U, w \in W$ such that $(u, l, w) \in E\}$.

Intuitively speaking, a quotient graph can merge multiple nodes into one node, where the merged node preserves the edges of its constituent nodes. For an input graph $G = (V, E, L)$, there is an exponential number of potential quotient graphs: as many as there are partitions of the input graphs' nodes. On one extreme, the input graph is a quotient graph of itself (turning nodes like ⓤ into singleton nodes like ⟮{u}⟯). On the other extreme, a single node ⟮V⟯, with all input nodes, and loops $(V, l, V)$ for each edge-label $l$ used in $E$, the set of input edges, is also a quotient graph. Practical quotient graphs typically fall somewhere in between, where the partition $\mathcal{V}$ of $V$ is often defined in terms of an *equivalence relation* $\sim$ on the set $V$ such that $\mathcal{V} := \sim/V$; i.e., $\mathcal{V}$ is defined as the *quotient set* of $V$ with respect to $\sim$; for example, we might define an equivalence relation on nodes such that $u \sim v$ if and only if they have the same set of defined types, where $\sim/V$ is then a partition whose parts contain all nodes with the same types. Another way to induce a quotient graph is to define the partition in a way that preserves some of the topology of the input graph. One way to formally define this idea is through *simulation* and *bisimulation*.

*Definition B.29 (Simulation).* Given two directed-edge labelled graph $G := (V, E, L)$ and $G' := (G', E', L')$, let $R \subseteq V \times V'$ be a relation between the nodes of $G$ and $G'$, respectively. We call $R$ a *simulation* on $G$ and $G'$ if, for all $(v, v') \in R$, the following holds:
- if $(v, p, w) \in E$ then there exists $w'$ such that $(v', p, w') \in E'$ and $(w, w') \in R$.

If a simulation exists on $G$ and $G'$, we say that $G'$ *simulates* $G$, denoted $G \rightsquigarrow G'$.

*Definition B.30 (Bisimulation).* If $R$ is a simulation on $G$ and $G'$, we call it a *bisimulation* if, for all $(v, v') \in R$, the following condition holds:
- if $(v'p, w') \in E'$ then there exists $w$ such that $(v, p, w) \in E$ and $(w, w') \in R$.

If a bisimulation exists on $G$ and $G'$, we say that they are *bisimilar*, denoted $G \approx G'$.

Bisimulation ($\approx$) is then an equivalence relation on graphs. By defining the (bi)simulation relation $R$ in terms of set membership $\in$, every quotient graph simulates its input graph, but does not necessarily bisimulate its input graph. This gives rise to the notion of *bisimilar quotient graphs*.

*Example B.31.* Figures 13 and 14 exemplify quotient graphs for the graph of Figure 1. Figure 13 simulates but is not bisimilar to the data graph. Figure 14 is bisimilar to the data graph. Often the

goal will be to compute the most concise quotient graph that satisfies a given condition; for example, the nodes without outgoing edges in Figure 14 could be merged while preserving bisimilarity.

## B.4 Context

*B.4.1 Annotation domain.* We define an annotation domain per Zimmermann et al. [546].

*Definition B.32 (Annotation domain).* Let $A$ be a set of *annotation values*. An *annotation domain* is defined as an idempotent, commutative semi-ring $D = \langle A, \oplus, \otimes, \bot, \top \rangle$.

This definition can be used to instantiate specific domains of context. Letting $D$ be a semi-ring imposes that, for any values $a, a_1, a_2, a_3$ in $A$, the following hold:

- $(a_1 \oplus a_2) \oplus a_3 = a_1 \oplus (a_2 \oplus a_3)$
- $(\bot \oplus a) = (a \oplus \bot) = l$
- $(a_1 \oplus a_2) = (a_2 \oplus a_1)$
- $(a_1 \oplus a_2) = (a_2 \oplus a_1)$
- $(a_1 \otimes a_2) \otimes a_3 = a_1 \otimes (a_2 \otimes a_3)$
- $(\top \otimes a) = (a \otimes \top) = a$
- $a_1 \otimes (a_2 \oplus a_3) = (a_1 \otimes a_2) \oplus (a_1 \otimes a_3)$
- $(a_1 \oplus a_2) \otimes a_3 = (a_1 \otimes a_3) \oplus (a_2 \otimes a_3)$
- $(\bot \otimes a) = (a \otimes \bot) = \bot$

The requirement that it be a commutative semi-ring imposes the following constraint:

- $(a_1 \otimes a_2) = (a_2 \otimes a_1)$

Finally, the requirement that it be an idempotent semi-ring imposes the following constraint:

- $(a \oplus a) = a$

Idempotence induces a partial order: $a_1 \leq a_2$ if and only if $a_1 \oplus a_2 = a_2$. Imposing these conditions on the annotation domain allow for reasoning and querying to be conducted over the annotation domain in a well-defined manner. Annotated graphs can then be defined in the natural way:

*Definition B.33 (Annotated directed-edge labelled graph).* Letting $D = \langle A, \oplus, \otimes, \bot, \top \rangle$ denote an idempotent, commutative semi-ring, we define an *annotated directed-edge labelled graph* $G :=$ $(V, E_A, L)$ where $V \subset \mathbf{Con}$ is a finite set of nodes, $L \subset \mathbf{Con}$ is a finite set of edge labels, and $E \subseteq V \times L \times V \times A$ is a finite set of edges annotated with values from $A$.

Figure 19 exemplifies query answering on a graph annotated with days of the year. Formally this domain can be defined as follows: $L := 2^{\mathbb{N}_{[1,365]}}, \oplus := \cup, \otimes := \cap, \top := \mathbb{N}_{[1,365]}, \bot := \emptyset$, where one may verify that $D = \langle 2^{\mathbb{N}_{[1,365]}}, \cup, \cap, \mathbb{N}_{[1,365]}, \emptyset \rangle$ is indeed an idempotent, commutative semi-ring.

## B.5 Deductive Knowledge

We provide some formal definitions for concepts relating to deductive knowledge, starting with the notion of an interpretation for a graph. We then describe some logical formalisms by which reasoning can be conducted over graphs, describing rules and Description Logics.

*B.5.1 Graph interpretations.* A *graph interpretation* – or simply *interpretation* – captures the assumptions under which the semantics of a graph can be defined. We define interpretations for directed edge-labelled graphs, though the notion extends naturally to other graph models.

*Definition B.34 (Graph interpretation).* A *(graph) interpretation* $I$ is defined as a pair $I := (\Gamma, \cdot^I)$ where $\Gamma = (V_\Gamma, E_\Gamma, L_\Gamma)$ is a (directed edge-labelled) graph called the *domain graph* and $\cdot^I : \mathbf{Con} \rightarrow V_\Gamma \cup L_\Gamma$ is a partial mapping from constants to terms in the domain graph.

We denote the domain of the mapping $\cdot^I$ by $\mathrm{dom}(\cdot^I)$. For interpretations under the UNA, the mapping $\cdot^I$ is required to be injective, while with no UNA (NUNA), no such requirement is necessary. Interpretations that *satisfy* a graph are then said to be *models* of that graph. We first define this notion for a base case that ignores ontological features.

*Definition B.35 (Graph models).* Let $G := (V, E, L)$ be a directed edge-labelled graph. An interpretation $I := (\Gamma, \cdot^I)$ *satisfies* $G$ if and only if the following hold:

- $V \cup L \subseteq \mathrm{dom}(\cdot^I)$;
- for all $v \in V$, it holds that $v^I \in V_\Gamma$;
- for all $l \in L$, it holds that $l^I \in L_\Gamma$; and
- for all $(u, l, v) \in E$, it holds that $(u^I, l^I, v^I) \in E_\Gamma$.

If $I$ *satisfies* $G$ we call $I$ a *(graph) model* of $G$.

Next we define models under semantics conditions (e.g., of ontology features).

*Definition B.36 (Semantic condition).* Let $2^G$ denote the set of all (directed edge-labelled) graphs. A *semantic condition* is a mapping $\phi : 2^G \rightarrow \{\mathrm{true}, \mathrm{false}\}$. An interpretation $I := (\Gamma, \cdot^I)$ is a model of $G$ under $\phi$ if and only if $I$ is a model of $G$ and $\phi(\Gamma)$. Given a set of semantic conditions $\Phi$, we say that $I$ is a model of $G$ if and only if $I$ is a model of $G$ and for all $\phi \in \Phi$, $\phi(\Gamma)$ is true.

We do not restrict the language used to define semantic conditions, but, for example, we can define the HAS VALUE semantic condition of Table 4 in FOL as follows:

$$\forall c, p, y \Big( \big( \Gamma(c, \mathsf{prop}, p) \wedge \Gamma(c, \mathsf{value}, y) \big) \leftrightarrow \forall x \big( \Gamma(x, \mathsf{type}, c) \leftrightarrow \Gamma(x, p, y) \big) \Big)$$

Here we overload $\Gamma$ as a ternary predicate to capture the edges of $\Gamma$. The above FOL formula defines an if-and-only-if version of the semantic condition for HAS VALUE. The other semantic conditions enumerated in Tables 2–4 can be defined in a similar way [441].[40]

Finally, we can define entailment considering such semantic conditions.

*Definition B.37 (Graph entailment).* Letting $G_1$ and $G_2$ denote two (directed edge-labelled) graphs, and $\Phi$ a set of semantic conditions, we say that $G_1$ *entails* $G_2$ *under* $\Phi$ – denoted $G_1 \models_\Phi G_2$ – if and only if any model of $G_2$ under $\Phi$ is also a model of $G_1$ under $\Phi$.

An example of entailment is discussed in Section 4.2. Note that in a slight abuse of notation, we may simply write $G \models_\Phi (s, p, o)$ to denote that $G$ entails the edge $(s, p, o)$ under $\Phi$.

Under OWA, entailment is as defined as given in Definition B.37. Under CWA, we make the additional assumption that if $G \not\models_\Phi e$, where $e$ is an edge (strictly speaking, a *positive* edge), then $G \models_\Phi \neg e$; in other words, under CWA we assume that any (positive) edges that $G$ does not entail under $\Phi$ can be assumed false according to $G$ and $\Phi$.[41]

*B.5.2 Rules.* Given a graph pattern $Q$ – be it a directed edge-labelled graph pattern per Definition B.7 or a property graph pattern per Definition B.9 – recall that $\mathbf{Var}(Q)$ denotes the variables appearing in $Q$. We can now define the notion of a rule for graphs.

*Definition B.38 (Rule).* A *rule* is a pair $R := (B, H)$ such that $B$ and $H$ are graph patterns and $\mathbf{Var}(H) \subseteq B$. The graph pattern $B$ is called the *body* of the rule while $H$ is called the *head* of the rule.

---

[40]Note that although these tables consider axioms originating in the data graph, it suffices to check their image in the domain graph since $I$ only satisfies $G$ if the edges of $G$ defining the axioms are reflected in $I$.

[41]In FOL, the CWA only applies to positive *facts*, whereas edges in a graph can be used to represent other FOL formulae. If one wished to maintain FOL-compatibility under CWA, additional restrictions on the types of edge $e$ may be needed.

This definition of a rule applies for directed edge-labelled graphs and property graphs by considering the corresponding type of graph pattern. The head is considered to be a conjunction of edges. Given a graph $G$, a rule is *applied* by computing the mappings from the body to the graph and then using those mappings to substitute the variables in $H$. The restriction $\mathbf{Var}(H) \subseteq B$ ensures that the results of this substitution is a graph, with no variables in $H$ left unsubstituted.

*Definition B.39 (Rule application).* Given a rule $R = (B, H)$ and a graph $G$, we define the *application of $R$ over $G$* as the graph $R(G) := \bigcup_{\mu \in B(G)} \mu(H)$.

Given a set of rules $\mathcal{R} := \{R_1, \ldots R_n\}$ and a knowledge graph $G$, towards defining the set of inferences given by the rules over the graph, we denote by $\mathcal{R}(G) := \bigcup_{R \in \mathcal{R}} R(G)$ the union of the application of all rules of $\mathcal{R}$ over $G$, and we denote by $\mathcal{R}^+(G) := \mathcal{R}(G) \cup G$ the extension of $G$ with respect to the application of $\mathcal{R}$. Finally, we denote by $\mathcal{R}^k(G)$ (for $k \in \mathbb{N}^+$) the recursive application of $\mathcal{R}^+(G)$, where $\mathcal{R}^1(G) := \mathcal{R}^+(G)$, and $\mathcal{R}^{i+1}(G) := \mathcal{R}^+(\mathcal{R}^i(G))$). We are now ready to define the *least model*, which captures the inferences possible for $\mathcal{R}$ over $G$.

*Definition B.40 (Least model).* The *least model of $\mathcal{R}$ over $G$* is defined as $\mathcal{R}^*(G) := \bigcup_{k \in \mathbb{N}} (R^k(G))$.

At some point $R^{k'}(G) = R^{k'+1}(G)$: the rule applications reach a fixpoint and we have the least model. Once the least model $\mathcal{R}^*(G)$ is computed, the entailed data can be treated as any other data.

Rules can be used to support graph entailments of the form $G_1 \models_\Phi G_2$. We say that a set of rules $\mathcal{R}$ is *correct* for $\Phi$ if, for any graph $G$, $G \models_\Phi \mathcal{R}^*(G)$. We say that $\mathcal{R}$ is *complete* for $\Phi$ if, for any graph $G$, there does not exist an edge $e$ such that $G \models_\Phi e$ and $e \notin \mathcal{R}^*(G)$. Table 5 exemplifies a correct (but incomplete) set of rules for the semantic conditions laid out by the RDFS standard [66].

Alternatively, rules can be directly specified in a rule language such as Notation3 (N3) [38], Rule Interchange Format (RIF) [270], Semantic Web Rule Language (SWRL) [242], or SPARQL Inferencing Notation (SPIN) [277]. Languages such as SPIN represent rules as graphs, allowing the rules of a knowledge graph to be embedded in the data graph. Taking advantage of this fact, we can then consider a form of graph entailment $G_1 \cup \gamma(\mathcal{R}) \models_\Phi G_2$, where by $\gamma(\mathcal{R})$ we denote the graph representation of rules $\mathcal{R}$. If the set of rules $\mathcal{R}$ is correct and complete for $\Phi$, we may simply write $G_1 \cup \gamma(\mathcal{R}) \models G_2$, indicating that $\Phi$ captures the same semantics for $\gamma(\mathcal{R})$ as applying the rules in $\mathcal{R}$; formally, $G_1 \cup \gamma(\mathcal{R}) \models \mathcal{R}(G_1 \cup \gamma(\mathcal{R}))$ and there does not exist an edge $e$ such that $G_1 \cup \gamma(\mathcal{R}) \models e$ but $e \notin \mathcal{R}^*(G_1 \cup \gamma(\mathcal{R}))$. This allows us to view rules as another form of graph entailment.

*B.5.3 Description Logics.* Table 6 provides definitions for all of the constructs typically found in Description Logics. The syntax column denotes how the construct is expressed in DL. A DL knowledge base then consists of an A-Box, a T-Box, and an R-Box.

*Definition B.41 (DL knowledge base).* A *DL knowledge base* K is defined as a tuple (A, T, R), where A is the *A-Box*: a set of assertional axioms; T is the *T-Box*: a set of class (aka concept/terminological) axioms; and R is the *R-Box*: a set of relation (aka property/role) axioms.

The semantics column defines the meaning of axioms using *interpretations*. These interpretations are typically defined in a slightly different way to those previously defined for graphs, though the idea is roughly the same.

*Definition B.42 (DL interpretation).* A *DL interpretation* $I$ is defined as a pair $(\Delta^I, \cdot^I)$, where $\Delta^I$ is the *interpretation domain*, and $\cdot^I$ is the *interpretation function*. The interpretation domain is a set of individuals. The interpretation function accepts a definition of either an individual $a$, a class $C$, or a relation $R$, mapping them, respectively, to an element of the domain ($a^I \in \Delta^I$), a subset of the domain ($C^I \subseteq \Delta^I$), or a set of pairs from the domain ($R^I \subseteq \Delta^I \times \Delta^I$).

An interpretation $I$ *satisfies* a knowledge-base K if and only if, for all of the syntactic axioms in K, the corresponding semantic conditions in Table 6 hold for $I$. In this case, we call $I$ a *model* of K.

As an example, for K := (A, T, R), let:

- A := {City(Arica), City(Santiago), flight(Arica, Santiago)};
- T := {City ⊑ Place, ∃flight.⊤ ⊑ ∃nearby.Airport};
- R := {flight ⊑ connectsTo}.

For $I = (\Delta^I, \cdot^I)$, let:

- $\Delta^I$ := {⚓, ◭, ✈};
- $\text{Arica}^I$ := ⚓, $\text{Santiago}^I$ := ◭, $\text{AricaAirport}^I$ := ✈;
- $\text{City}^I$ := {⚓, ◭}, $\text{Airport}^I$ := {✈};
- $\text{flight}^I$ := {(⚓, ◭)}, $\text{connectsTo}^I$ := {(⚓, ◭)}, $\text{sells}^I$ := {(✈, ☕)}.

The interpretation $I$ is not a model of K since it does not have that ⚓ is nearby some Airport, nor that ⚓ and ◭ are in the class Place. However, if we *extend* $I$ with the following:

- $\text{Place}^I$ := {⚓, ◭};
- $\text{nearby}^I$ := {(⚓, ✈)}.

Now $I$ is a model of K. Note that although K does not imply that sells(Arica, coffee) while $I$ indicates that ✈ sells ☕, $I$ is still a model of K since K is not assumed to be a complete description of the world, as per the Open World Assumption.

Finally, the notion of a model gives rise to the key notion of entailment.

*Definition B.43.* Given two DL knowledge bases $K_1$ and $K_2$, we define that $K_1$ entails $K_2$, denoted $K_1 \models K_2$, if and only if any model of $K_2$ is a model of $K_1$.

The entailment relation tells us which knowledge bases hold as a logical consequence of which others: if all models of $K_2$ are also models of $K_1$ then, intuitively speaking, $K_2$ says nothing new over $K_1$. For example, let $K_1$ denote the knowledge base K from the previous example, and define a second knowledge base with one assertion: $K_2$ := ({connectsTo(Arica, Santiago)}, {}, {}). Though $K_1$ does not assert this axiom, it does entail $K_2$: to be a model of $K_2$, an interpretation must have that $(\text{Arica}^I, \text{Santiago}^I) \in \text{connectsTo}^I$, but this must also be the case for any interpretation that satisfies $K_1$ since it must have that $(\text{Arica}^I, \text{Santiago}^I) \in \text{flight}^I$ and $\text{flight}^I \subseteq \text{connectsTo}^I$.

Unfortunately, the problem of deciding entailment for knowledge bases expressed in the DL composed of the unrestricted use of all of the axioms of Table 6 combined is undecidable. We could, for example, reduce instances of the Halting Problem to such entailment. Hence DLs in practice restrict use of the features listed in Table 6. Different DLs then apply different restrictions, implying different trade-offs for expressivity and the complexity of the entailment problem. Most DLs are founded on one of the following base DLs (we use indentation to denote derivation):

$\mathcal{ALC}$ (*Attributive Language with Complement* [439]), supports atomic classes, the top and bottom classes, class intersection, class union, class negation, universal restrictions and existential restrictions. Relation and class assertions are also supported.

$\quad$ $\mathcal{S}$ extends $\mathcal{ALC}$ with transitive closure.

These base languages can be extended as follows:

$\mathcal{H}$ adds relation inclusion.

$\quad$ $\mathcal{R}$ adds (limited) complex relation inclusion, as well as relation reflexivity, relation irreflexivity, relation disjointness and the universal relation.

$\mathcal{O}$ adds (limited) nomimals.

$\mathcal{I}$ adds inverse relations.

$\mathcal{F}$ adds (limited) functional properties.

$\mathcal{N}$ adds (limited) number restrictions (subsuming $\mathcal{F}$ given $\top$).

$\mathcal{Q}$ adds (limited) qualified number restrictions (subsuming $\mathcal{N}$ given $\top$).

We use "(limited)" to indicate that such features are often only allowed under certain restrictions to ensure decidability; for example, complex relations (chains) typically cannot be combined with cardinality restrictions. DLs are then typically named per the following scheme, where $[a|b]$ denotes an alternative between $a$ and $b$ and $[c][d]$ denotes a concatenation $cd$:

$$[\mathcal{ALC}|\mathcal{S}][\mathcal{H}|\mathcal{R}][O][I][\mathcal{F}|\mathcal{N}|\mathcal{Q}]$$

Examples include $\mathcal{ALCO}$, $\mathcal{ALCHI}$, $\mathcal{SHIF}$, $\mathcal{SROIQ}$, etc. These languages often apply additional restrictions on class and property axioms to ensure decidability, which we do not discuss here. For further details on Description Logics, we refer to the recent book by Baader et al. [21].

As mentioned in the body of the survey, DLs have been very influential in the definition of OWL, where the OWL 2 DL fragment (roughly) corresponds to the DL $\mathcal{SROIQ}$. For example, the axiom (venue)–domain→(Event) in OWL can be translated to $\exists$venue.$\top \sqsubseteq$ Event, meaning that the class of individuals with some value for venue (in any class) is a sub-class of the class Event. We leave other translations from the OWL axioms of Tables 2–4 to DL as an exercise.[42] Note, however, that axioms like (sub-taxon of)–subp. of→(subc. of) – which given a graph such as (Fred)–type→(Homo sapiens)–sub-taxon of→(Hominini) entails the edge (Fred)–type→(Hominini) – cannot be expressed in DL: "subTaxonOf $\sqsubseteq \sqsubseteq$" is not syntactically valid. Hence only a subset of graphs can be translated into well-formed DL ontologies; we refer to the OWL standard for details [228].

## B.6 Inductive Knowledge

We provide further discussion and formal definitions relating to graph parallel frameworks, knowledge graph embeddings, and graph neural networks, as discussed in Section 5.

*B.6.1 Graph parallel frameworks.* Before defining a graph parallel framework, in the interest of generality, we first define a directed graph labelled with feature vectors, which captures the type of input that such a framework can accept, with vectors assigned to both nodes and edges.

*Definition B.44 (Directed vector-labelled graph).* We define a *directed vector-labelled graph* $G = (V, E, F, \lambda)$, where $V$ is a set of nodes, $E \subseteq V \times V$ is a set of edges, $F$ is a set of feature vectors, and $\lambda : V \cup E \to F$ labels each node and edge with a feature vector.

A directed-edge labelled graph or a property graph may be encoded as a directed vector-labelled graph in a number of ways, depending on the application. The type of node and/or a selection of its attributes may be encoded in the node feature vectors, while the label of an edge and/or a selection of its attributes may be encoded in the edge feature vector (including, for example, weights applied to edges). Typically node feature vectors will all have the same dimensionality, as will edge feature vectors. The directed vector-labelled graph can thus be seen as defining the initial state and features that will be used as input for the graph parallel framework.

*Example B.45.* We define a directed vector-labelled graph in preparation for later computing PageRank using a graph parallel framework. Let $G = (V, E, L)$ denote a directed edge-labelled graph. Let $|E(u)|$ denote the outdegree of node $u \in V$. We then initialise a directed vector-labelled graph $G' = (V, E', F, \lambda)$ such that $E' = \{(x, z) \mid \exists y : (x, y, z) \in E\}$, and for all $u \in V$, we define

$$\lambda(u) := \begin{bmatrix} \frac{1}{|V|} \\ |E'(u)| \\ |V| \end{bmatrix}, \text{ and } \lambda(u, v) := [\ ], \text{ with } F := \{\lambda(u) \mid u \in V\} \cup \{\lambda(u, v) \mid (u, v) \in E'\}, \text{ assigning}$$

---

Table 6. Description Logic semantics

| Name | Syntax | Semantics ($\cdot^I$) |
|------|--------|----------------------|
| CLASS DEFINITIONS | | |
| Atomic Class | $A$ | $A^I$ (a subset of $\Delta^I$) |
| Top Class | $\top$ | $\Delta^I$ |
| Bottom Class | $\bot$ | $\emptyset$ |
| Class Negation | $\neg C$ | $\Delta^I \setminus C^I$ |
| Class Intersection | $C \sqcap D$ | $C^I \cap D^I$ |
| Class Union | $C \sqcup D$ | $C^I \cup D^I$ |
| Nominal | $\{a_1, ..., a_n\}$ | $\{a_1^I, ..., a_n^I\}$ |
| Existential Restriction | $\exists R.C$ | $\{x \mid \exists y : (x, y) \in R^I \text{ and } y \in C^I\}$ |
| Universal Restriction | $\forall R.C$ | $\{x \mid \forall y : (x, y) \in R^I \text{ implies } y \in C^I\}$ |
| Self Restriction | $\exists R.\text{Self}$ | $\{x \mid (x, x) \in R^I\}$ |
| Number Restriction | $\star\, n\, R$ (where $\star \in \{\geq, \leq, =\}$) | $\{x \mid \#\{y : (x, y) \in R^I\} \star n\}$ |
| Qualified Number Restriction | $\star\, n\, R.C$ (where $\star \in \{\geq, \leq, =\}$) | $\{x \mid \#\{y : (x, y) \in R^I \text{ and } y \in C^I\} \star n\}$ |
| CLASS AXIOMS (T-Box) | | |
| Class Inclusion | $C \sqsubseteq D$ | $C^I \subseteq D^I$ |
| RELATION DEFINITIONS | | |
| Relation | $R$ | $R^I$ (a subset of $\Delta^I \times \Delta^I$) |
| Inverse Relation | $R^-$ | $\{(y, x) \mid (x, y) \in R^I\}$ |
| Universal Relation | $\mathsf{U}$ | $\Delta^I \times \Delta^I$ |
| RELATION AXIOMS (R-Box) | | |
| Relation Inclusion | $R \sqsubseteq S$ | $R^I \subseteq S^I$ |
| Complex Relation Inclusion | $R_1 \circ ... \circ R_n \sqsubseteq S$ | $R_1^I \circ ... \circ R_n^I \subseteq S^I$ |
| Transitive Relations | $\text{Trans}(R)$ | $R^I \circ R^I \subseteq R^I$ |
| Functional Relations | $\text{Func}(R)$ | $\{(x, y), (x, z)\} \subseteq R^I \text{ implies } y = z$ |
| Reflexive Relations | $\text{Ref}(R)$ | for all $x \in \Delta^I : (x, x) \in R^I$ |
| Irreflexive Relations | $\text{Irref}(R)$ | for all $x \in \Delta^I : (x, x) \notin R^I$ |
| Symmetric Relations | $\text{Sym}(R)$ | $R^I = (R^-)^I$ |
| Asymmetric Relations | $\text{Asym}(R)$ | $R^I \cap (R^-)^I = \emptyset$ |
| Disjoint Relations | $\text{Disj}(R, S)$ | $R^I \cap S^I = \emptyset$ |
| ASSERTIONAL DEFINITIONS | | |
| Individual | $a$ | $a^I$ (an element of $\Delta^I$) |
| ASSERTIONAL AXIOMS (A-Box) | | |
| Relation Assertion | $R(a, b)$ | $(a^I, b^I) \in R^I$ |
| Negative Relation Assertion | $\neg R(a, b)$ | $(a^I, b^I) \notin R^I$ |
| Class Assertion | $C(a)$ | $a^I \in C^I$ |
| Equality | $a = b$ | $a^I = b^I$ |
| Inequality | $a \neq b$ | $a^I \neq b^I$ |

each node a vector containing its initial PageRank score, the outdegree of the node, and the number of nodes in the graph. Conversely, edge-vectors are not used in this case.

We are now ready to define a graph parallel framework operating over a directed vector-labelled graph. In the following we use $\{\!\{\cdot\}\!\}$ to denote a multiset (an unordered set preserving duplicates),

$2^{S \to \mathbb{N}}$ to denote the set of all multisets containing (only) elements from the set $S$, and $\mathbb{R}^a$ to denote the set of all vectors of dimension $a$ (i.e., the set of all vectors containing $a$ real-valued elements).

*Definition B.46 (Graph parallel framework).* A *graph parallel framework* (*GPF*) is a triple of functions $\mathfrak{G} := (\text{Msg}, \text{Agg}, \text{End})$ such that (with $a, b, c \in \mathbb{N}$):

- $\text{Msg} : \mathbb{R}^a \times \mathbb{R}^b \to \mathbb{R}^c$
- $\text{Agg} : \mathbb{R}^a \times 2^{\mathbb{R}^c \to \mathbb{N}} \to \mathbb{R}^a$
- $\text{End} : 2^{\mathbb{R}^a \to \mathbb{N}} \to \{\text{true}, \text{false}\}$

The function Msg defines what message (i.e., vector) must be passed from a node to a neighbouring node along a particular edge, given the current feature vectors of the node and the edge; the function Agg is used to compute a new feature vector for a node, given its previous feature vector and incoming messages; the function End defines a condition for termination of vector computation. The integers $a$, $b$ and $c$ denote the dimensions of node feature vectors, edge feature vectors, and message vectors, respectively; we assume that $a$ and $b$ correspond with the dimensions of input feature vectors for nodes and edges. Given a GPF $\mathfrak{G} = (\text{Msg}, \text{Agg}, \text{End})$, a directed vector-labelled graph $G = (V, E, F, \lambda)$, and a node $u \in V$, we define the output vector assigned to node $u$ in $G$ by $\mathfrak{G}$ (written $\mathfrak{G}(G, u)$) as follows. First let $\mathbf{n}_u^{(0)} := \lambda(u)$. For all $i \geq 1$, let:

$$M_u^{(i)} := \left\{\!\!\left\{ \text{Msg}\left(\mathbf{n}_v^{(i-1)}, \lambda(v, u)\right) \middle| (v, u) \in E \right\}\!\!\right\}$$

$$\mathbf{n}_u^{(i)} := \text{Agg}\left(\mathbf{n}_u^{(i-1)}, M_u^{(i)}\right)$$

If $j$ is the smallest integer for which $\text{End}(\{\!\{\mathbf{n}_u^{(j)} \mid u \in V\}\!\})$ is true, then $\mathfrak{G}(G, u) := \mathbf{n}_u^{(j)}$.

This particular definition assumes that vectors are dynamically computed for nodes, and that messages are passed only to outgoing neighbours, but the definitions can be readily adapted to consider dynamic vectors for edges, or messages being passed to incoming neighbours, etc. We now provide an example instantiating a GPF to compute PageRank over a directed graph.

*Example B.47.* We take as input the directed vector labelled graph $G' = (V, E, F, \lambda)$ from Example B.45 for a PageRank GPF. First we define the messages passed from $u$ to $v$:

$$\text{Msg}\left(\mathbf{n}_v, \lambda(v, u)\right) := \left[ \frac{d(\mathbf{n}_v)_1}{(\mathbf{n}_v)_2} \right]$$

where $d$ denotes PageRank's constant dampening factor (typically $d := 0.85$) and $(\mathbf{n}_v)_k$ denotes the $k^{\text{th}}$ element of the $\mathbf{n}_v$ vector. In other words, $v$ will pass to $u$ its PageRank score multiplied by the dampening factor and divided by its degree (we do not require $\lambda(v, u)$ in this particular example). Next we define the function for $u$ to aggregate the messages it receives from other nodes:

$$\text{Agg}\left(\mathbf{n}_u, M_u\right) := \begin{bmatrix} \frac{1-d}{(\mathbf{n}_u)_3} + \sum_{\mathbf{m} \in M_u} (\mathbf{m})_1 \\ (\mathbf{n}_u)_2 \\ (\mathbf{n}_u)_3 \end{bmatrix}$$

Here, we sum the scores received from other nodes along with its share of rank from the dampening factor, copying over the node's degree and the total number of nodes for future use. Finally, there are a number of ways that we could define the termination condition; here we simply define:

$$\text{End}(\{\!\{\mathbf{n}_u^{(i)} \mid u \in V\}\!\}) := (i \geq z)$$

where $z$ is a fixed number of iterations, at which point the process stops.

We may note in this example that the total number of nodes is duplicated in the vector for each node of the graph. Part of the benefit of GPFs is that only local information in the neighbourhood of the node is required for each computation step. In practice, such frameworks may allow additional features, such as global computation steps whose results are made available to all nodes [314], operations that dynamically modify the graph [314], etc.

*B.6.2   Knowledge graph embeddings.* As discussed in Section 5.2, knowledge graph embeddings represent graphs in a low-dimensional numeric space.[43] Before defining the key notions, we introduce mathematical objects related to tensor calculus, on which embeddings heavily rely.

*Definition B.48 (Vector, matrix, tensor, order, mode).* For any positive integer $n$, a *vector* of dimension $a$ is a family of real numbers indexed by integers in $\{1, \ldots, a\}$. For $a$ and $b$ positive integers, an $(a, b)$-matrix is a family of real numbers indexed by pairs of integers in $\{1, \ldots, a\} \times \{1, \ldots, b\}$. A tensor is a family of real numbers indexed by a finite sequence of integers such that there exist positive numbers $a_1, \ldots, a_n$ such that the indices are all the numbers in $\{1, \ldots, a_1\} \times \ldots \times \{1, \ldots, a_n\}$. The number $n$ is called the *order* of the tensor, the subindices $i \in \{1, \ldots, n\}$ indicate the *mode* of a tensor, and each $a_i$ defines the dimension of the $i^{\text{th}}$ mode. A 1-order tensor is a vector and a 2-order tensor is a matrix. We denote the set of all tensors as $\mathbb{T}$.

For specific dimensions $a_1, \ldots, a_n$ of modes, a tensor is an element of $(\cdots(\mathbb{R}^{a_1})\cdots)^{a_n}$ but we write $\mathbb{R}^{a_1, \ldots, a_n}$ to simplify the notation. We use lower-case bold font to denote vectors ($\mathbf{x} \in \mathbb{R}^a$), upper-case bold font to denote matrices ($\mathbf{X} \in \mathbb{R}^{a,b}$) and calligraphic font to denote tensors ($\mathcal{X} \in \mathbb{R}^{a_1, \ldots, a_n}$).

Now we are ready to abstractly define knowledge graph embeddings.

*Definition B.49 (Knowledge graph embedding).* Given a directed edge-labelled graph $G = (V, E, L)$, a *knowledge graph embedding of $G$* is a pair of mappings $(\varepsilon, \rho)$ such that $\varepsilon : V \to \mathbb{T}$ and $\rho : L \to \mathbb{T}$.

In the most typical case, $\epsilon$ and $\rho$ map nodes and edge-labels, respectively, to vectors of fixed dimension. In some cases, however, they may map to matrices. Given this abstract notion of a knowledge graph embedding, we can then define a plausibility score.

*Definition B.50 (Plausibility).* A *plausibility scoring function* is a partial function $\phi : \mathbb{T} \times \mathbb{T} \times \mathbb{T} \to \mathbb{R}$. Given a directed edge-labelled graph $G = (V, E, L)$, an edge $(s, p, o) \in V \times L \times V$, and a knowledge graph embedding $(\varepsilon, \rho)$ of $G$, the plausibility of $(s, p, o)$ is given as $\phi(\varepsilon(s), \rho(p), \varepsilon(o))$.

Edges with higher scores are considered to be more plausible. Given a graph $G = (V, E, L)$, we assume a set of positive edges $E^+$ and a set of negative edges $E^-$. Positive edges are often simply the edges in the graph: $E^+ := E$. Negative edges use the vocabulary of $G$ (i.e., $E^- \subseteq V \times L \times V$) and typically are defined by taking edges $(s, p, o)$ from $E$ and changing one of the terms of each edge – most often, but not always, one of the nodes – such that the edge is no longer in $E$. Given sets of positive and negative edges, and a plausibility scoring function, the objective is then to find the embedding that maximises the plausibility of edges in $E^+$ while minimising the plausibility of edges in $E^-$. Specific knowledge graph embeddings then instantiate the type of embedding considered and the plausibility scoring function in (a wide variety of) different ways.

In Table 7, we define the plausibility scoring function used by different models for knowledge graph embeddings, and further provide details of the types of embeddings considered. To simplify the definitions of embeddings given in Table 7, we will use $\mathbf{e}_x$ to denote $\varepsilon(x)$ when it is a vector,

---

[43]To the best of our knowledge, the term "*knowledge graph embedding*" was coined by Wang et al. [521] in order to distinguish the case from a "graph embedding" that considers a single relation (i.e., an undirected or directed graph). Earlier papers rather used the phrase "*multi-relational data*" [59, 185, 364].

Table 7. Details for selected knowledge graph embeddings, including the plausibility scoring function $\phi(\varepsilon(s), \rho(p), \varepsilon(o))$ for edge $\boxed{s} - p \blacktriangleright \boxed{o}$, and other conditions applied

| Model | $\phi(\varepsilon(s), \rho(p), \varepsilon(o))$ | Conditions (for all $x \in V, y \in L$) |
|---|---|---|
| TransE [59] | $-\|\mathbf{e}_s + \mathbf{r}_p - \mathbf{e}_o\|_q$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, q \in \{1, 2\}, \|\mathbf{e}_x\|_2 = 1$ |
| TransH [521] | $-\|(\mathbf{e}_s - (\mathbf{e}_s^T\mathbf{w}_p)\mathbf{w}_p) + \mathbf{r}_p - (\mathbf{e}_o - (\mathbf{e}_o^T\mathbf{w}_p)\mathbf{w}_p)\|_2^2$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d,$ $\|\mathbf{w}_y\|_2 = 1, \frac{\mathbf{w}_y^T\mathbf{r}_y}{\|\mathbf{r}_y\|_2} \approx 0, \|\mathbf{e}_x\|_2 \leq 1$ |
| TransR [256] | $-\|\mathbf{W}_p\mathbf{e}_s + \mathbf{r}_p - \mathbf{W}_p\mathbf{e}_o\|_2^2$ | $\mathbf{e}_x \in \mathbb{R}^{d_e}, \mathbf{r}_y \in \mathbb{R}^{d_r}, \mathbf{W}_y \in \mathbb{R}^{d_r, d_e},$ $\|\mathbf{e}_x\|_2 \leq 1, \|\mathbf{r}_y\|_2 \leq 1, \|\mathbf{W}_y\mathbf{e}_x\|_2 \leq 1$ |
| TransD [256] | $-\|(\mathbf{w}_p \otimes \mathbf{w}_s + \mathbf{I})\mathbf{e}_s + \mathbf{r}_p - (\mathbf{w}_p \otimes \mathbf{w}_o + \mathbf{I})\mathbf{e}_o\|_2^2$ | $\mathbf{e}_x \in \mathbb{R}^{d_e}, \mathbf{r}_y \in \mathbb{R}^{d_r}, \mathbf{w}_x \in \mathbb{R}^{d_e}, \mathbf{w}_y \in \mathbb{R}^{d_r},$ $\|\mathbf{e}_x\|_2 \leq 1, \|\mathbf{r}_y\|_2 \leq 1, \|(\mathbf{w}_y \otimes \mathbf{w}_x + \mathbf{I})\mathbf{e}_x\|_2 \leq 1$ |
| RESCAL [364] | $\mathbf{e}_s^T\mathbf{R}_p\mathbf{e}_o$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{R}_y \in \mathbb{R}^{d,d}, \|\mathbf{e}_x\|_2 \leq 1, \|\mathbf{R}_y\|_{2,2} \leq 1$ |
| DistMult [533] | $\mathbf{e}_s^T\mathbf{r}_p^D\mathbf{e}_o$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \|\mathbf{e}_x\|_2 = 1, \|\mathbf{r}_y\|_2 \leq 1$ |
| HolE [363] | $\mathbf{r}_p^T(\mathbf{e}_s \star \mathbf{e}_o)$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \|\mathbf{e}_x\|_2 \leq 1, \|\mathbf{r}_y\|_2 \leq 1$ |
| ComplEx [499] | $\mathrm{Re}(\mathbf{e}_s^T\mathbf{r}_p^D\overline{\mathbf{e}}_o)$ | $\mathbf{e}_x \in \mathbb{C}^d, \mathbf{r}_y \in \mathbb{C}^d, \|\mathbf{e}_x\|_2 \leq 1, \|\mathbf{r}_y\|_2 \leq 1$ |
| SimplE [266] | $\frac{\mathbf{e}_s^T\mathbf{r}_p^D\mathbf{w}_o + \mathbf{e}_o^T\mathbf{w}_p^D\mathbf{w}_s}{2}$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \mathbf{w}_x \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d,$ $\|\mathbf{e}_x\|_2 \leq 1, \|\mathbf{w}_x\|_2 \leq 1, \|\mathbf{r}_y\|_2 \leq 1, \|\mathbf{w}_y\|_2 \leq 1$ |
| TuckER [27] | $\mathcal{W} \otimes_1 \mathbf{e}_s^T \otimes_2 \mathbf{r}_p^T \otimes_3 \mathbf{e}_o^T$ | $\mathbf{e}_x \in \mathbb{R}^{d_e}, \mathbf{r}_y \in \mathbb{R}^{d_r}, \mathcal{W} \in \mathbb{R}^{d_e, d_r, d_e}$ |
| SME Linear [185] | $(\mathbf{V}\mathbf{e}_s + \mathbf{V}'\mathbf{r}_p + \mathbf{v})^T(\mathbf{W}\mathbf{e}_o + \mathbf{W}'\mathbf{r}_p + \mathbf{w})$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \mathbf{v} \in \mathbb{R}^w, \mathbf{w} \in \mathbb{R}^w, \|\mathbf{e}_x\|_2 = 1,$ $\mathbf{V} \in \mathbb{R}^{w,d}, \mathbf{V}' \in \mathbb{R}^{w,d}, \mathbf{W} \in \mathbb{R}^{w,d}, \mathbf{W}' \in \mathbb{R}^{w,d}$ |
| SME Bilinear [185] | $((\mathcal{V} \otimes_3 \mathbf{r}_p^T)\mathbf{e}_s + \mathbf{v})^T((\mathcal{W} \otimes_3 \mathbf{r}_p^T)\mathbf{e}_o + \mathbf{w})$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \mathbf{v} \in \mathbb{R}^w, \mathbf{w} \in \mathbb{R}^w, \|\mathbf{e}_x\|_2 = 1,$ $\mathcal{V} \in \mathbb{R}^{w,d,d}, \mathcal{W} \in \mathbb{R}^{w,d,d}$ |
| NTN [463] | $\mathbf{r}_p^T\psi\left(\mathbf{e}_s^T\mathcal{W}\mathbf{e}_o + \mathbf{W}\begin{bmatrix}\mathbf{e}_s\\\mathbf{e}_o\end{bmatrix} + \mathbf{w}\right)$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^w, \mathbf{W} \in \mathbb{R}^{w,2d},$ $\mathcal{W} \in \mathbb{R}^{d,w,d}, \|\mathbf{e}_x\|_2 \leq 1, \|\mathbf{r}_y\|_2 \leq 1,$ $\|\mathbf{w}\|_2 \leq 1, \|\mathbf{W}\|_{2,2} \leq 1, \|\mathcal{W}_{1 \leq i \leq w}^{[:,i,:]}\|_{2,2} \leq 1$ |
| MLP [126] | $\mathbf{v}^T\psi\left(\mathbf{W}\begin{bmatrix}\mathbf{e}_s\\\mathbf{r}_p\\\mathbf{e}_o\end{bmatrix} + \mathbf{w}\right)$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, \mathbf{v} \in \mathbb{R}^w, \mathbf{w} \in \mathbb{R}^w, \mathbf{W} \in \mathbb{R}^{w,3d}$ $\|\mathbf{e}_x\|_2 \leq 1 \, \|\mathbf{r}_y\|_2 \leq 1$ |
| ConvE [122] | $\psi\left(\mathrm{vec}\left(\psi\left(\mathcal{W} * \begin{bmatrix}\mathbf{e}_s^{[a,b]}\\\mathbf{r}_p^{[a,b]}\end{bmatrix}\right)\right)^T\mathbf{W}\right)\mathbf{e}_o$ | $\mathbf{e}_x \in \mathbb{R}^d, \mathbf{r}_y \in \mathbb{R}^d, d = ab,$ $\mathbf{W} \in \mathbb{R}^{w_1(w_2+2a-1)(w_3+b-1),d}, \mathcal{W} \in \mathbb{R}^{w_1,w_2,w_3}$ |
| HypER [26] | $\psi\left(\mathrm{vec}\left(\mathbf{r}_p^T\mathcal{W} * \mathbf{e}_s\right)^T\mathbf{W}\right)\mathbf{e}_o$ | $\mathbf{e}_x \in \mathbb{R}^{d_e}, \mathbf{r}_y \in \mathbb{R}^{d_r}, \mathbf{W} \in \mathbb{R}^{w_2(w_1+d_e-1),d_e},$ $\mathcal{W} \in \mathbb{R}^{d_r,w_1,w_2}$ |

and we will use $\mathbf{r}_y$ to denote $\rho(y)$ when it is a vector and $\mathbf{R}_y$ to denote $\rho(y)$ when it is a matrix. Some models use additional parameters (aka weights) that – although they do not form part of the entity/relation embeddings – are learnt to compute the plausibility score from the embeddings. We denote these as $\mathbf{v}, \mathbf{V}, \mathcal{V}, \mathbf{w}, \mathbf{W}\, \mathcal{W}$ (for vectors, matrices or tensors). We use $d_e$ and $d_r$ to denote the dimensionality chosen for entity embeddings and relation embeddings, respectively. Often it is assumed that $d_e = d_r$, in which case we will write $d$. Sometimes weights may have their own dimensionality, which we denote $w$. The embeddings in Table 7 use a variety of operators on vectors, matrices and tensors. In the interest of keeping the discussion self-contained, we refer to the latter part of this section for definitions of these operators and other conventions used.

The embeddings listed in Table 7 vary in complexity, ranging from simple models such as TransE [59] and DistMult [533], to more complex ones, such as SME Bilinear [185] and ConvE [122]. A trade-off underlies these proposals in terms of the number of parameters used, where more parameters increases computational costs, but increases the expressiveness of the model in terms of the model's capability to capture latent features of the graph. To increase expressivity, many of the models in Table 7 use additional parameters beyond the embeddings themselves. A possible formal guarantee of such models is *full expressiveness*, which, given any disjoint sets of positive edges $E^+$ and negative edges $E^-$, asserts that the model can always correctly partition those edges. On the one hand, for example, DistMult [533] cannot distinguish an edge ⓢ–ⓟ→ⓞ from its inverse ⓞ–ⓟ→ⓢ, so by adding an inverse of an edge in $E^+$ to $E^-$, we can show that it is *not* fully expressive. On the other hand, models such as ComplEx [499], SimplE [266], and TuckER [27] have been proven to be fully expressive given sufficient dimensionality; for example, TuckER [27] with dimensions $d_r = |L|$ and $d_e = |V|$ trivially satisfies full expressivity since its core tensor $\mathcal{W}$ then has sufficient capacity to store the full one-hot encoding of any graph. This formal property is useful to show that the model does not have built-in limitations for numerically representing a graph, though of course in practice the dimensions needed to reach full expressivity are often impractical/undesirable.

Here we have not discussed language models for embedding [92, 416], which are based on a distinct set of principles, or entailment-aware models [120, 200, 520], which add additional scoring constraints on top of the types of models listed in Table 7. For further information on such works, we refer to the survey by Wang et al. [519] and/or the corresponding papers.

We continue by defining in detail the operators and conventions used in Table 7. We start with the conventions used, thereafter defining the pertinent operators.

- We use indexed parentheses – such as $(\mathbf{x})_i$, $(\mathbf{X})_{ij}$, or $(\mathcal{X})_{i_1 \ldots i_n}$ – to denote elements of vectors, matrices, and tensors, respectively. If a vector $\mathbf{x} \in \mathbb{R}^a$ is used in a context that requires a matrix, the vector is interpreted as an $(a, 1)$-matrix (i.e., a column vector) and can be turned into a row vector (i.e., a $(1, a)$-matrix) using the transpose operation $\mathbf{x}^T$. We use $\mathbf{x}^D \in \mathbb{R}^{a,a}$ to denote the diagonal matrix with the values of the vector $\mathbf{x} \in \mathbb{R}^a$ on its diagonal. We denote the identity matrix by $\mathbf{I}$ such that if $j = k$, then $(\mathbf{I})_{jk} = 1$; otherwise $(\mathbf{I})_{jk} = 0$.

- We denote by $\begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_n \end{bmatrix}$ the vertical stacking of matrices $\mathbf{X}_1, \ldots, \mathbf{X}_n$ with the same number of columns. Given a vector $\mathbf{x} \in \mathbb{R}^{ab}$, we denote by $\mathbf{x}^{[a,b]} \in \mathbb{R}^{a,b}$ the "reshaping" of $\mathbf{x}$ into an $(a, b)$-matrix such that $(\mathbf{x}^{[a,b]})_{ij} = (\mathbf{x})_{(i+a(j-1))}$. Conversely, given a matrix $\mathbf{X} \in \mathbb{R}^{a,b}$, we denote by $\text{vec}(\mathbf{X}) \in \mathbb{R}^{ab}$ the *vectorisation* of $\mathbf{X}$ such that $\text{vec}(\mathbf{X})_k = (\mathbf{X})_{ij}$ where $i = ((k - 1) \bmod m) + 1$ and $j = \frac{k-i}{m} + 1$ (observe that $\text{vec}(\mathbf{x}^{[a,b]}) = \mathbf{x}$).

- Given a tensor $\mathcal{X} \in \mathbb{R}^{a,b,c}$, we denote by $\mathcal{X}^{[i::]} \in \mathbb{R}^{b,c}$, the $i^{\text{th}}$ *slice* of tensor $\mathcal{X}$ along the first mode; for example, given $\mathcal{X} \in \mathbb{R}^{5,2,3}$, then $\mathcal{X}^{[4::]}$ returns the $(2, 3)$-matrix consisting of the elements $\begin{bmatrix} (\mathcal{X})_{411} & (\mathcal{X})_{412} & (\mathcal{X})_{413} \\ (\mathcal{X})_{421} & (\mathcal{X})_{422} & (\mathcal{X})_{423} \end{bmatrix}$. Analogously, we use $\mathcal{X}^{[:i:]} \in \mathbb{R}^{a,c}$ and $\mathcal{X}^{[::i]} \in \mathbb{R}^{b,c}$ to indicate the $i^{\text{th}}$ slice along the second and third modes of $\mathcal{X}$, respectively.

- We denote by $\psi(\mathcal{X})$ the element-wise application of a function $\psi$ to the tensor $\mathcal{X}$, such that $(\psi(\mathcal{X}))_{in_1 \ldots i_n} = \psi(\mathcal{X}_{i_1 \ldots i_n})$. Common choices for $\psi$ include a sigmoid function (e.g., the logistic function $\psi(x) = \frac{1}{1+e^{-x}}$ or the hyperbolic tangent function $\psi(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$), the rectifier ($\psi(x) = \max(0, x)$), softplus ($\psi(x) = \ln(1 + e^x)$), etc.

The first and most elemental operation we consider is that of matrix multiplication.

*Definition B.51 (Matrix multiplication).* The *multiplication of matrices* $\mathbf{X} \in \mathbb{R}^{a,b}$ and $\mathbf{Y} \in \mathbb{R}^{b,c}$ is a matrix $\mathbf{XY} \in \mathbb{R}^{a,c}$ such that $(\mathbf{XY})_{ij} = \sum_{k=1}^{b} (\mathbf{X})_{ik} (\mathbf{Y})_{kj}$. The matrix multiplication of two tensors $\mathcal{X} \in \mathbb{R}^{a_1,\ldots,a_m,c}$ and $\mathcal{Y} \in \mathbb{R}^{c,b_1,\ldots,b_n}$ is a tensor $\mathcal{X}\mathcal{Y} \in \mathbb{R}^{a_1,\ldots,a_m,b_1,\ldots,b_n}$ such that $(\mathcal{X}\mathcal{Y})_{i_1\ldots i_m i_{m+1}\ldots i_{m+n}} = \sum_{k=1}^{c} (\mathcal{X})_{i_1\ldots i_m k} (\mathcal{Y})_{k i_{m+1} i_{m+n}}$.

For convenience, we may implicitly add or remove modes with dimension 1 for the purposes of matrix multiplication and other operators; for example, given two vectors $\mathbf{x} \in \mathbb{R}^a$ and $\mathbf{y} \in \mathbb{R}^a$, we denote by $\mathbf{x}^\mathsf{T}\mathbf{y}$ (aka the dot or inner product) the multiplication of matrix $\mathbf{x}^\mathsf{T} \in \mathbb{R}^{1,a}$ with $\mathbf{y} \in \mathbb{R}^{a,1}$ such that $\mathbf{x}^\mathsf{T}\mathbf{y} \in \mathbb{R}^{1,1}$ (i.e., a scalar in $\mathbb{R}$); conversely, $\mathbf{xy}^\mathsf{T} \in \mathbb{R}^{a,a}$ (the outer product).

Constraints on embeddings are sometimes given in terms of norms, defined next.

*Definition B.52 ($L^p$-norm, $L^{p,q}$-norm).* For $p \in \mathbb{R}$, the *$L^p$-norm* of a vector $\mathbf{x} \in \mathbb{R}^a$ is the scalar $\|\mathbf{x}\|_p := (|(\mathbf{x})_1|^p + \ldots + |(\mathbf{x})_a|^p)^{\frac{1}{p}}$, where $|(\mathbf{x})_i|$ denotes the absolute value of the $i^{\text{th}}$ element of $\mathbf{x}$. For $p, q \in \mathbb{R}$, the *$L^{p,q}$-norm* of a matrix $\mathbf{X} \in \mathbb{R}^{a,b}$ is the scalar $\|\mathbf{X}\|_{p,q} := \left( \sum_{j=1}^{b} \left( \sum_{i=1}^{a} |(\mathbf{X})_{ij}|^p \right)^{\frac{q}{p}} \right)^{\frac{1}{q}}$.

The $L^1$ norm (i.e., $\|\mathbf{x}\|_1$) is thus simply the sum of the absolute values of $\mathbf{x}$, while the $L^2$ norm (i.e., $\|\mathbf{x}\|_2$) is the (Euclidean) length of the vector. The Frobenius norm of the matrix $\mathbf{X}$ then equates to $\|\mathbf{X}\|_{2,2} = \left( \sum_{j=1}^{b} \left( \sum_{i=1}^{a} |(\mathbf{X})_{ij}|^2 \right) \right)^{\frac{1}{2}}$; i.e., the square root of the sum of the squares of all elements.

As discussed in Section 5.2, tensor decompositions are an important concept for many embeddings, and at the heart of such decompositions is the tensor product.

*Definition B.53 (Tensor product).* Given two tensors $\mathcal{X} \in \mathbb{R}^{a_1,\ldots,a_m}$ and $\mathcal{Y} \in \mathbb{R}^{b_1,\ldots,b_n}$, the *tensor product* $\mathcal{X} \otimes \mathcal{Y}$ is defined as a tensor in $\mathbb{R}^{a_1,\ldots,a_m,b_1,\ldots,b_n}$, with each element computed as $(\mathcal{X} \otimes \mathcal{Y})_{i_1\ldots i_m j_1\ldots j_n} := (\mathcal{X})_{i_1\ldots i_m} (\mathcal{Y})_{j_1\ldots j_n}$.[44]

To illustrate the tensor product, assume that $\mathcal{X} \in \mathbb{R}^{2,3}$ and $\mathcal{Y} \in \mathbb{R}^{3,4,5}$. The result of $\mathcal{X} \otimes \mathcal{Y}$ will be a tensor in $\mathbb{R}^{2,3,3,4,5}$. Element $(\mathcal{X} \otimes \mathcal{Y})_{12345}$ will be computed by multiplying $(\mathcal{X})_{12}$ and $(\mathcal{Y})_{345}$.

An *n*-mode product is used by other embeddings to transform elements along a mode of a tensor.

*Definition B.54 (n-mode product).* For a positive integer $n$, a tensor $\mathcal{X} \in \mathbb{R}^{a_1,\ldots,a_{n-1},a_n,a_{n+1},\ldots,a_m}$ and matrix $\mathbf{Y} \in \mathbb{R}^{b,a_n}$, the *n-mode product* of $\mathcal{X}$ and $\mathbf{Y}$ is the tensor $\mathcal{X} \otimes_n \mathbf{Y} \in \mathbb{R}^{a_1,\ldots,a_{n-1},b,a_{n+1},\ldots,a_m}$ such that $(\mathcal{X} \otimes_n \mathbf{Y})_{i_1\ldots i_{n-1} j i_{n+1}\ldots i_m} := \sum_{k=1}^{a_n} (\mathcal{X})_{i_1\ldots i_{n-1} k i_{n+1}\ldots i_m} (\mathbf{Y})_{jk}$.

To illustrate, let us assume that $\mathcal{X} \in \mathbb{R}^{2,3,4}$ and $\mathbf{Y} \in \mathbb{R}^{5,3}$. The result of $\mathcal{X} \otimes_2 \mathbf{Y}$ will be a tensor in $\mathbb{R}^{2,5,4}$, where, for example, $(\mathcal{X} \otimes_2 \mathbf{Y})_{142}$ will be given as $(\mathcal{X})_{112}(\mathbf{Y})_{41} + (\mathcal{X})_{122}(\mathbf{Y})_{42} + (\mathcal{X})_{132}(\mathbf{Y})_{43}$. Observe that if $\mathbf{y} \in \mathbb{R}^{a_n}$ – i.e., if $\mathbf{y}$ is a (column) vector – then the *n*-mode tensor product $\mathcal{X} \otimes_n \mathbf{y}^\mathsf{T}$ "flattens" the $n^{\text{th}}$ mode of $\mathcal{X}$ to one dimension, effectively reducing the order of $\mathcal{X}$ by one.

One embedding technique – HolE [363] – uses a circular correlation operator.

*Definition B.55 (Circular correlation).* The *circular correlation* of vector $\mathbf{x} \in \mathbb{R}^a$ with $\mathbf{y} \in \mathbb{R}^a$ is the vector $\mathbf{x} \star \mathbf{y} \in \mathbb{R}^a$ such that $(\mathbf{x} \star \mathbf{y})_k := \sum_{i=1}^{a} (\mathbf{x})_i (\mathbf{y})_{(((k+i-2) \bmod a)+1)}$.

Each element of $\mathbf{x} \star \mathbf{y}$ is the sum of $a$ elements along a diagonal of the outer product $\mathbf{x} \otimes \mathbf{y}$ that "wraps" if not the primary diagonal. Assuming $a = 5$, then $(\mathbf{x} \star \mathbf{y})_1 = (\mathbf{x})_1(\mathbf{y})_1 + (\mathbf{x})_2(\mathbf{y})_2 + (\mathbf{x})_3(\mathbf{y})_3 + (\mathbf{x})_4(\mathbf{y})_4 + (\mathbf{x})_5(\mathbf{y})_5$, or a case that wraps: $(\mathbf{x} \star \mathbf{y})_4 = (\mathbf{x})_1(\mathbf{y})_4 + (\mathbf{x})_2(\mathbf{y})_5 + (\mathbf{x})_3(\mathbf{y})_1 + (\mathbf{x})_4(\mathbf{y})_2 + (\mathbf{x})_5(\mathbf{y})_3$.

---

[44]Please note that "⊗" is used here in an unrelated sense to its use in Definition B.32.

Another embedding technique – ComplEx [499] – uses *complex vectors*. With a slight abuse of notation, the definitions of vectors, matrices and tensors can be modified by replacing the set of real numbers $\mathbb{R}$ by the set of complex numbers $\mathbb{C}$, giving rise to complex vectors, complex matrices, and complex tensors. In this case, we denote by $\mathrm{Re}(\cdot)$ the real part of a complex number. Given a complex vector $\mathbf{x} \in \mathbb{C}^I$, we denote by $\bar{\mathbf{x}}$ its complex conjugate (swapping the sign of the imaginary part of each element). Complex analogues of the aforementioned operators can then be defined by replacing the multiplication and addition of real numbers with complex numbers, where ComplEx [499] specifically uses complex matrix multiplication.

Finally, a couple of neural models that we include – namely ConvE [122] and HypER [26] – are based on convolutional architectures using the convolution operator.

*Definition B.56 (Convolution).* Given two matrices $\mathbf{X} \in \mathbb{R}^{a,b}$ and $\mathbf{Y} \in \mathbb{R}^{e,f}$, the *convolution* of $\mathbf{X}$ and $\mathbf{Y}$ is the matrix $\mathbf{X} * \mathbf{Y} \in \mathbb{R}^{(a+e-1),(b+f-1)}$ such that $(\mathbf{X}*\mathbf{Y})_{ij} = \sum_{k=1}^{a}\sum_{l=1}^{b}(\mathbf{X})_{kl}(\mathbf{Y})_{(i+k-a)(j+l-b)}$.[45] In cases where $(i+k-a) < 1$, $(j+l-b) < 1$, $(i+k-a) > e$ or $(j+l-b) > f$ (i.e., where $(\mathbf{Y})_{(i+k-a)(j+l-b)}$ lies outside the bounds of $\mathbf{Y}$), we say that $(\mathbf{Y})_{(i+k-a)(j+l-b)} = 0$.

Intuitively speaking, the convolution operator overlays $\mathbf{X}$ in every possible way over $\mathbf{Y}$ such that at least one pair of elements $(\mathbf{X})_{ij}, (\mathbf{Y})_{lk}$ overlaps, summing the products of pairs of overlapping elements to generate an element of the result. Elements of $\mathbf{X}$ extending beyond $\mathbf{Y}$ are ignored (equivalently we can consider $\mathbf{Y}$ to be "zero-padded" outside its borders). To illustrate, given $\mathbf{X} \in \mathbb{R}^{3,3}$ and $\mathbf{Y} \in \mathbb{R}^{4,5}$, then $\mathbf{X} * \mathbf{Y} \in \mathbb{R}^{6,7}$, where, for example, $(\mathbf{X}*\mathbf{Y})_{11} = (\mathbf{X})_{33}(\mathbf{Y})_{11}$ (with the bottom right corner of $\mathbf{X}$ overlapping the top left corner of $\mathbf{Y}$), while $(\mathbf{X}*\mathbf{Y})_{34} = (\mathbf{X})_{11}(\mathbf{Y})_{12} + (\mathbf{X})_{12}(\mathbf{Y})_{13} + (\mathbf{X})_{13}(\mathbf{Y})_{14} + (\mathbf{X})_{21}(\mathbf{Y})_{22} + (\mathbf{X})_{22}(\mathbf{Y})_{23} + (\mathbf{X})_{23}(\mathbf{Y})_{24} + (\mathbf{X})_{31}(\mathbf{Y})_{32} + (\mathbf{X})_{32}(\mathbf{Y})_{33} + (\mathbf{X})_{33}(\mathbf{Y})_{34}$ (with $(\mathbf{X})_{22}$ – the centre of $\mathbf{X}$ – overlapping $(\mathbf{Y})_{23}$).[46] In a convolution $\mathbf{X} * \mathbf{Y}$, the matrix $\mathbf{X}$ is often called the "kernel" (or "filter"). Often several kernels are used in order to apply multiple convolutions. Given a tensor $\mathcal{X} \in \mathbb{R}^{c,a,b}$ (representing $c$ $(a,b)$-kernels) and a matrix $\mathbf{Y} \in \mathbb{R}^{e,f}$, we denote by $\mathcal{X} * \mathbf{Y} \in \mathbb{R}^{c,(a+e-1),(b+f-1)}$ the result of the convolutions of the $c$ first-mode slices of $\mathcal{X}$ over $\mathbf{Y}$ such that $(\mathcal{X} * \mathbf{Y})^{[i\cdots]} = \mathcal{X}^{[i\cdots]} * \mathbf{Y}$ for $1 \le i \le c$, yielding a tensor of results for $c$ convolutions.

*B.6.3 Graph neural networks.* We now provide high-level definitions for graph neural networks (GNNs) inspired by (for example) the definitions provided by Xu et al. [531]. We assume that the GNN accepts a directed vector-labelled graph as input (see Definition B.44).

We first abstractly define a recursive graph neural network.

*Definition B.57 (Recursive graph neural network).* A *recursive graph neural network* (*RecGNN*) is a pair of functions $\mathfrak{R} := (\textsc{Agg}, \textsc{Out})$, such that (with $a, b, c \in \mathbb{N}$):

- $\textsc{Agg} : \mathbb{R}^a \times 2^{(\mathbb{R}^a \times \mathbb{R}^b) \to \mathbb{N}} \to \mathbb{R}^a$
- $\textsc{Out} : \mathbb{R}^a \to \mathbb{R}^c$

The function $\textsc{Agg}$ computes a new feature vector for a node, given its previous feature vector and the feature vectors of the nodes and edges forming its neighbourhood; the function $\textsc{Out}$ transforms the final feature vector computed by $\textsc{Agg}$ for a node to the output vector for that node. We assume that $a$ and $b$ correspond to the dimensions of the input node and edge vectors, respectively, while $c$ denotes the dimension of the output vector for each node. Given a RecGNN $\mathfrak{R} = (\textsc{Agg}, \textsc{Out})$,

---

[45]We define the convolution operator per the convention for convolutional neural networks. Strictly speaking, the operator should be called *cross-correlation*, where traditional convolution requires the matrix $\mathbf{X}$ to be initially "rotated" by 180°. Since in our settings the matrix $\mathbf{X}$ is learnt, rather than given, the rotation is redundant.

[46]Models applying convolutions may differ regarding how edge cases are handled, or on the "stride" of the convolution applied, where, for example, a stride of 3 for $(\mathbf{X} * \mathbf{Y})$ would see the kernel $\mathbf{X}$ centred only on elements $(\mathbf{Y})_{ij}$ such that $i \bmod 3 = 0$ and $j \bmod 3 = 0$, reducing the number of output elements by a factor of 9. We do not consider such details here.

a directed vector-labelled graph $G = (V, E, F, \lambda)$, and a node $u \in V$, we define the output vector assigned to node $u$ in $G$ by $\mathfrak{R}$ (written $\mathfrak{R}(G, u)$) as follows. First let $\mathbf{n}_u^{(0)} := \lambda(u)$. For all $i \geq 1$, let:

$$\mathbf{n}_u^{(i)} := \text{Agg}\left(\mathbf{n}_u^{(i-1)}, \{\!\{(\mathbf{n}_v^{(i-1)}, \lambda(v, u)) \mid (v, u) \in E\}\!\}\right)$$

If $j \geq 1$ is an integer such that $\mathbf{n}_u^{(j)} = \mathbf{n}_u^{(j-1)}$ for all $u \in V$, then $\mathfrak{R}(G, u) := \text{Out}(\mathbf{n}_u^{(j)})$.

In a RecGNN, the same aggregation function (Agg) is applied recursively until a fixpoint is reached, at which point an output function (Out) creates the final output vector for each node. While in practice RecGNNs will often consider a static feature vector and a dynamic state vector [437], we can more concisely encode this as one vector, where part may remain static throughout the aggregation process representing input features, and part may be dynamically computed representing the state. In practice, Agg and Out are often based on parametric combinations of vectors, with the parameters learnt based on a sample of output vectors for labelled nodes.

*Example B.58.* The aggregation function for the GNN of Scarselli et al. [437] is given as:

$$\text{Agg}(\mathbf{n}_u, N) := \sum_{(\mathbf{n}_v, \mathbf{a}_{vu}) \in N} f_{\mathbf{w}}(\mathbf{n}_u, \mathbf{n}_v, \mathbf{a}_{vu})$$

where $f_{\mathbf{w}}(\cdot)$ is a contraction function with parameters $\mathbf{w}$. The output function is defined as:

$$\text{Out}(\mathbf{n}_u) := g_{\mathbf{w}'}(\mathbf{n}_u)$$

where again $g_{\mathbf{w}'}(\cdot)$ is a function with parameters $\mathbf{w}'$. Given a set of nodes labelled with their expected output vectors, the parameters $\mathbf{w}$ and $\mathbf{w}'$ are learnt.

There are notable similarities between graph parallel frameworks (GPFs; see Definition B.46) and RecGNNs. While we defined GPFs using separate Msg and Agg functions, this is not essential: conceptually they could be defined in a similar way to RecGNN, with a single Agg function that "pulls" information from its neighbours (we maintain Msg to more closely reflect how GPFs are defined/implemented in practice). The key difference between GPFs and GNNs is that in the former, the functions are defined by the user, while in the latter, the functions are generally learnt from labelled examples. Another difference arises from the termination condition present in GPFs, though often the GPF's termination condition will – like in RecGNNs – reflect convergence to a fixpoint.

Next we abstractly define a non-recursive graph neural network.

*Definition B.59 (Non-recursive graph neural network).* A *non-recursive graph neural network* (NRecGNN) with $l$ layers is an $l$-tuple of functions $\mathfrak{R} := (\text{Agg}^{(1)}, \ldots, \text{Agg}^{(l)})$, such that, for $1 \leq k \leq l$ (with $a_0, \ldots a_l, b \in \mathbb{N}$), $\text{Agg}^{(k)} : \mathbb{R}^{a_{k-1}} \times 2^{(\mathbb{R}^{a_{k-1}} \times \mathbb{R}^b) \to \mathbb{N}} \to \mathbb{R}^{a_k}$.

Each function $\text{Agg}^{(k)}$ (as before) computes a new feature vector for a node, given its previous feature vector and the feature vectors of the nodes and edges forming its neighbourhood. We assume that $a_0$ and $b$ correspond to the dimensions of the input node and edge vectors, respectively, where each function $\text{Agg}^{(k)}$ for $2 \leq k \leq l$ accepts as input node vectors of the same dimension as the output of the function $\text{Agg}^{(k-1)}$. Given an NRecGNN $\mathfrak{R} = (\text{Agg}^{(1)}, \ldots, \text{Agg}^{(l)})$, a directed vector-labelled graph $G = (V, E, F, \lambda)$, and a node $u \in V$, we define the output vector assigned to node $u$ in $G$ by $\mathfrak{R}$ (written $\mathfrak{R}(G, u)$) as follows. First let $\mathbf{n}_u^{(0)} := \lambda(u)$. For all $i \geq 1$, let:

$$\mathbf{n}_u^{(i)} := \text{Agg}^{(i)}\left(\mathbf{n}_u^{(i-1)}, \{\!\{(\mathbf{n}_v^{(i-1)}, \lambda(v, u)) \mid (v, u) \in E\}\!\}\right)$$

Then $\mathfrak{R}(G, u) := \mathbf{n}_u^{(l)}$.

In an $l$-layer NRecGNN, a different aggregation function can be applied at each step (i.e., in each layer), up to a fixed number of steps $l$. We do not consider a separate Out function as it can be

combined with the final aggregation function $\text{AGG}^{(l)}$. When the aggregation functions are based on a convolutional operator, we call the result a *convolutional graph neural network* (*ConvGNN*). We refer to the survey by Wu et al. [527] for discussion of ConvGNNs proposed in the literature.

We have considered GNNs that define the neighbourhood of a node based on its incoming edges. However, these definitions can be adapted to also consider outgoing neighbours by either adding inverse edges to the directed vector-labelled graph in pre-processing, or by adding outgoing neighbours as arguments to the $\text{AGG}(\cdot)$ function. More generally, GNNs (and indeed GPFs) relying solely on the neighbourhood of each node have limited expressivity in terms of their ability to distinguish nodes and graphs [531]; for example, Barceló et al. [29] show that such NRecGNNs have a similar expressiveness for classifying nodes as the $\mathcal{ALCQ}$ Description Logic discussed in Section B.5.3. More expressive GNN variants have been proposed that allow the aggregation functions to access and update a globally shared vector [29]. We refer to the papers by Xu et al. [531] and Barceló et al. [29] for further discussion on the expressivity of GNNs.

*B.6.4 Symbolic learning.* We provide some abstract formal definitions for the tasks of *rule mining* and *axiom mining* over graphs, which we generically call *hypothesis mining*. First we introduce *hypothesis induction*: a task that captures a more abstract (ideal) case for hypothesis mining.

*Definition B.60 (Hypothesis induction).* The task of *hypothesis induction* assumes a particular graph entailment relation $\models_\Phi$ (see Definition B.37; hereafter simply $\models$). Given *background knowledge* in the form of a knowledge graph $G$ (a directed edge-labelled graph, possibly extended with rules or ontologies), a set of *positive edges* $E^+$ such that $G$ does not entail any edge in $E^+$ (i.e., for all $e^+ \in E^+$, $G \not\models e^+$) and $E^+$ does not contradict $G$ (i.e., there is a model of $G \cup E^+$), and a set of *negative edges* $E^-$ such that $G$ does not entail any edge in $E^-$ (i.e., for all $e^- \in E^-$, $G \not\models e^-$), the task is to find a set of *hypotheses* (i.e., a set of directed edge-labelled graphs) $\Psi$ such that:

- $G \not\models \psi$ for all $\psi \in \Psi$ (the background knowledge does not entail any hypothesis)
- $G \cup \Psi^* \models E^+$ (the background knowledge and hypotheses entail all positive edges);
- for all $e^- \in E^-$, $G \cup \Psi^* \not\models e^-$ (the background knowledge and hypotheses do not entail any negative edge);
- $G \cup \Psi^* \cup E^+$ has a model (the background knowledge, hypotheses and positive edges taken together do not contain a contradiction);
- for all $e^+ \in E^+$, $\Psi^* \not\models e^+$ (the hypotheses alone do not entail a positive edge).

where by $\Psi^* \coloneqq \cup_{\psi \in \Psi} \psi$ we denote the union of all graphs in $\Psi$.

*Example B.61.* Let us assume ontological entailment $\models$ with semantic conditions $\Phi$ as defined in Tables 2–4. Given the graph of Figure 27 as the background knowledge $G$, along with

- a set of positive edges $E^+ = \{$ SCL —flight→ ARI , SCL -domestic flight→ ARI $\}$, and
- a set of negative edges $E^- = \{$ ARI —flight→ LIM , SCL -domestic flight→ LIM $\}$,

then a set of hypotheses $\Psi = \{$ flight —type→ Symmetric , domestic flight —type→ Symmetric $\}$ would entail all positive edges in $E^+$ and no negative edges in $E^-$ when combined with $G$.

This task represents a somewhat idealised case. Often there is no set of positive edges distinct from the background knowledge itself. Furthermore, hypotheses not entailing a few positive edges, or entailing a few negative edges, may still be useful. The task of *hypothesis mining* rather accepts as input the background knowledge $G$ and a set of negative edges $E^-$ (such that for all $e^- \in E^-$, $G \not\models e^-$), and attempts to *score* individual hypotheses $\psi$ (such that $G \not\models \psi$) in terms of their ability to "explain" $G$ while minimising the number of elements of $E^-$ entailed by $G$ and $\psi$.

We can now abstractly define the task of hypothesis mining.

*Definition B.62 (Hypothesis mining).* Given a knowledge graph $G$, a set of negative edges $E^-$, a scoring function $\sigma$, and a threshold $min_\sigma$, the goal of *hypothesis mining* is to identify a set of hypotheses $\{\psi \mid G \not\models \psi \text{ and } \sigma(\psi, G, E^-) \geq min_\sigma\}$.

There are two main scoring functions used for $\sigma$ in the literature: *support* and *confidence*.

*Definition B.63 (Hypothesis support and confidence).* Given a knowledge graph $G = (V, E, L)$ and a hypothesis $\psi$, the *positive support* of $\psi$ is defined as follows:

$$\sigma^+(\psi, G) := |\{e \in E \mid G' \not\models e \text{ and } G' \cup \psi \models e\}|$$

where $G'$ denotes $G$ with the edge $e$ removed. Further given a set of negative edges $E^-$, the *negative support* of $\psi$ is defined as follows:

$$\sigma^-(\psi, G, E^-) := |\{e^- \in E^- \mid G \cup \psi \models e^-\}|$$

Finally, the *confidence* of $\psi$ is defined as $\sigma^\pm(\psi, G, E^-) := \frac{\sigma^+(\psi, G)}{\sigma^+(\psi, G) + \sigma^-(\psi, G, E^-)}$.

We have yet to define how the set of negative edges are defined, which, in the context of a knowledge graph $G$, depends on which assumption is applied:

- *Closed world assumption (CWA)*: For any (positive) edge $e$, $G \not\models e$ if and only if $G \models \neg e$. Under CWA, any edge $e$ not entailed by $G$ can be considered a negative edge.
- *Open world assumption*: For a (positive) edge $e$, $G \not\models e$ does not necessarily imply $G \models \neg e$. Under OWA, the negation of an edge must be entailed by $G$ for it to be considered negative.
- *Partial completeness assumption (PCA)*: If there exists $(s, p, o)$ such that $G \models (s, p, o)$, then for all $o'$ such that $G \not\models (s, p, o')$, it holds that $G \models \neg(s, p, o')$. Under PCA, if $G$ entails some outgoing edge(s) labelled $p$ from a node $s$, then such edges are assumed to be complete, and any edge $(s, p, o)$ not entailed by $G$ can be considered a negative edge.

Knowledge graphs are generally incomplete – in fact, one of the main applications of hypothesis mining is to try to improve the completeness of the knowledge graph – and thus it would appear unwise to assume that any edge that is not currently entailed is false/negative. We can thus rule out CWA. Conversely, under OWA, potentially few (or no) negative edges might be entailed by the given ontologies/rules, and thus hypotheses may end up having low negative support despite entailing many edges that do not make sense in practice. Hence the PCA can be adopted as a heuristic to increase the number of negative edges and apply more sensible scoring of hypotheses.

Different implementations of hypothesis mining may consider different logical languages. Rule mining, for example, mines hypotheses expressed either as monotonic rules (with positive edges) or non-monotonic edges (possibly with negated edges). On the other hand, axiom mining considers hypotheses expressed in a logical language such as Description Logics. Particular implementations may, for practical reasons, impose further syntactic restrictions on the hypotheses generated, such as to impose thresholds on their length, on the symbols they use, or on other structural properties (such as "closed rules" in the case of the AMIE rule mining system [164]; see Section 5.4). Systems may further implement different search strategies for hypotheses. Systems such as AMIE [164], RuLES [230], CARL [381], DL-Learner [70], etc., propose *discrete mining* that recursively generates candidate formulae through refinement/genetic operators that are then scored and checked for threshold criteria, thus navigating a branching search space. On the other hand, systems such as NeuralLP [534] and DRUM [430] apply *differentiable mining* that allows for learning (path-like) rules and their scores in a more continuous fashion (e.g., using gradient descent). We refer to Section 5.4 for further discussion and examples of such techniques for mining hypotheses.