Exploring the Universe with Artificial Intelligence (UniversAI)
Proceedings IAU Symposium No. 397, 2026
I. Liodakis & V. Efthymiou, eds.
doi:10.1017/S1743921325104559

Ontology-Based Data Access in Ten Questions

Meghyn Bienvenu

Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France
email: meghyn.bienvenu@labri.fr

Abstract. This extended abstract, structured as a series of ten questions, briefly summarizes the content of my keynote talk, which aimed to introduce astrophysicists to the area of ontology-based data access, a promising paradigm for designing intelligent information systems which has been extensively studied within the symbolic AI and database communities over past twenty years.

Keywords. Ontologies, ontology-based data access, data integration, query answering, knowledge representation and reasoning, semantic technologies

1. What is an Ontology?

In computer science, an ontology refers to a formal description of a shared conceptualization of some domain Gruber (2018), which consists of a vocabulary (list of terms or names) and formal statements that specify semantic relationships holding between the terms (e.g. asserting that one term is a specialization or synonym of another, declaring two terms as being disjoint, or defining a term in terms of other terms). By supplying a useful vocabulary for talking about a domain and constraining the meaning of its terms to avoid ambiguity, ontologies constitute a key tool for sharing, organizing, and querying information and data.

2. OK, but what exactly can I find in an ontology?

The vocabulary of an ontology is typically made up of names for *classes or concepts* (unary relations designating sets of entities) and *roles or properties* (binary relations that link two entities or relate an entity to some data value). For example, an ontology about the academic domain might include names for different classes of professors, students, and courses, e.g. `FullProfessor`, `AssociateProfessor`, `Student`, `AstronomyCourse`, `IntroCourse`, names for roles that link professors to the course they teach (`Teaches`), students to the courses they take (`TakesCourse`), or connecting a course to its prerequisite courses (`HasPrerequisite`), and names for data properties linking e.g. a professor to their salary (e.g. `HasSalary`) or to the date they were hired (e.g. `HasStartDate`). Typical ontology statements (also called axioms) describe the following semantic relationships:

- *subclass* relationship designating that one class is a subclass (specialization) of another, e.g. '`FullProfessor` is a subclass of `Professor`' means every full professor is a professor
- *subproperty* relationships: we might have a more general property `hasDate` and include an axiom '`HasStartDate` is a subproperty of `HasDate`'
- *disjointness* statements that stipulate that two classes or two properties cannot overlap, e.g. '`FullProfessor` and `AssociateProfessor` are disjoint classes' (one cannot be

© The Author(s), 2026. Published by Cambridge University Press on behalf of The International Astronomical Union.

https://doi.org/10.1017/S1743921325104559 Published online by Cambridge University Press

both full and associate professor at the same time) and '`Teaches` and `TakesCourse` are disjoint properties' (one cannot both teach a course and take the same course')

• *domain and range typing* that say which kinds of entities / values may occur in the first or second argument of properties, e.g. 'the domain of `TakesCourse` is `Student`' and 'the range of `TakesCourse` is `Course`' tell us that `TakesCourse` contains student-course pairs

• *mandatory participation constraints* that enforce that members of a given class must participate in some property, e.g. '`Professor` must participate in `Teaches`' (every professor must occur in the first argument of `Teaches`, i.e. must teach something) or '`Course` participates in the *inverse* of `Teaches`' (every course must occur in the second argument of `Teaches`, i.e. be taught by someone)

• *functionality* of properties, e.g. '`HasSalary` is functional' (there cannot be two different salaries for a same individual)

• *transitivity* of properties, e.g. declaring '`HasPrerequisite` is transitive' ensures we can identify both direct and indirect prerequisites

We should insist that the preceding items only scratch the surface of what ontologies can express, and some ontology languages allow for considerably more complex statements. For example, we might use axioms to define the class `PartTimeStudent` as those students who take between one and three courses, to state that a `GraduateCourse` can only be taught by a `Professor`, or to define `IntroCourse` as a course that does *not* have any prerequisite.

3. What is ontology-based data access?

Ontology-based data access (OBDA), alternatively known as ontology-mediated query answering or ontological query answering, uses an ontology to enrich one or more data sources. In some OBDA scenarios, one further employs a set of *mappings* to connect the data and ontology vocabularies. For instance, a mapping rule might express that if a data source contains a table `Employee`[*empID, fname, lname, job, deptID*], then we should add an employee id number to the ontology class `FullProfessor` provided that the employee id appears in a row of the `Employee` table whose *job* attribute has value 'FP'. When a user query arrives, the task of the OBDA system is to combine the information contained in the data source(s), ontology, and mappings in order to compute the query answers. The OBDA approach has been introduced by Poggi et al. (2008), and subsequent developments have been surveyed by Bienvenu and Ortiz (2015) and Xiao et al. (2018).

What purpose does the ontology serve? First, it can simplify query formulation by allowing users to write their queries using the ontology vocabulary, which has been designed to match users' conceptualization of the domain and will typically provide terms at different levels of generality. For example, a user could directly query for all `ScienceCourse`s, rather than needing to ask 'list all of the `AstronomyCourse`s and `CSCourse`s and `ChemistryCourse`s...'. Furthermore, the ontology shields users from the way the low-level details on how data is actually stored (e.g. no need to know that value 'FP' in the *job* column is used to indicate `FullProfessor`), which is particularly helpful in data integration settings where users need to query data from several heterogeneously-structured sources. Finally, the domain knowledge captured in the ontology enables the use of automated reasoning to identify data inconsistencies (e.g. a person holding two incompatible positions, or having multiple salaries) and provide users with more complete answers to queries (e.g. being able to infer both direct and indirect prerequisites for a given course, or to extract all employees across all categories).

https://doi.org/10.1017/S1743921325104559 Published online by Cambridge University Press

4. What ontology languages are available?

Ontologies are typically specified using some logic-based languages. While full first-order logic (FOL) is an appealing choice, due to its high expressivity, it is known to be difficult to handle computationally (e.g. it is impossible to design an algorithm, that always halts, for testing if a given set of FOL statements is consistent, i.e. does not contain any contradiction). For this reason, it is standard to employ decidable fragments of FOL, which offer better computational properties, such as description logics Baader et al. (2017), existential rules Mugnier and Thomazo (2014), or Datalog Abiteboul et al. (1995). *Description logics* (DLs) are a popular family of ontology languages that employ a set of constructors to build up complex class expressions from basic ones, and there are a whole range of different DLs to suit different modelling needs. *Existential rules* are another well-studied family of decidable logics used for specifying ontologies. They natively support higher-arity relations (whereas DLs only allow concepts and roles, with one or two arguments) and likewise comprise different fragments of varying expressivities. *Datalog* is another rule-based language, extensively studied in both the database and AI communities, which can be used to define ontologies. Unlike DLs and existential rules, Datalog can only be used to reason over named entities (thus preventing the expression of axioms like 'every graduate student must have a supervisor', allowing one to infer the existence of a potentially unknown supervisor).

While logical languages are well suited for theoretical investigation and offer a clear semantics, they are hardly user-friendly. For this reason, practical ontology languages have been developed to hide the logical symbols and mechanics from users. In particular, *OWL* (Ontology Web Language) is a well-known ontology language that has been standardized by the OWL Working Group (2012). OWL basically corresponds to a highly expressive description logic but with more convenient syntax and some extra bells and whistles. For simple ontologies, it is more common to employ *RDFS* (Resource Description Framework Schema), another W3C standard Brickley and Guha (2014), which offers support for subclass, subproperty, domain, and range axioms.

5. What kind of data can be accessed?

The OBDA approach typically assumes that the source data is provided in some structured format. If the ontology has been developed from the start, then application data may be provided as a set of unary and binary facts (often called an ABox) using the ontology predicates, such as $\texttt{AssistantProfessor}(pers357)$ and $\texttt{Teaches}(pers357, cs100)$. Another related and commonly supported data format is *RDF* (Resource Description Framework) Cyganiak et al. (2014), in which the data is given as a set of (subject, predicate, object) triples. For instance, the two preceding facts would be represented in RDF by triples $(pers357, \texttt{rdf:type}, \texttt{AssistantProfessor})$ and $(pers357, \texttt{Teaches}, cs100)$. Much real-world data still resides in *relational databases*, where the data is stored in one or more tables, with each table having a fixed number of attributes (aka columns, arguments). For instance, our earlier example considered a relational table `Employee` having attributes *empID* (employee id), *fname*, *lname* (first and last names), *job* (value indicating job type), and *deptID* (id of employee's department). Relational data is also supported by OBDA systems, with mappings used, if needed, to translate between the database and ontology vocabularies. Other formats that have been explored in the literature and can be handled by some systems include key-value stores and graph databases.

6. What kind of queries are supported?

Most work on OBDA assumes that the user query is either an *atomic query*, asking to find all members of a given class or property, or a *conjunctive query*, asking for all tuples

https://doi.org/10.1017/S1743921325104559 Published online by Cambridge University Press

of constants that satisfy a given conjunctive condition. Atomic queries are not as simple as they might appear at first glance: due to the ontology axioms, determining who is a `Professor` may involve considering several categories of professor and possibly other sufficient conditions for someone to have professor status. Conjunctive queries (CQs) go further by allowing conjunctions of atoms built from the class and property names in the ontology vocabulary, e.g. 'find all students who are taking both an astronomy and computer science course'. In logical syntax, this query can be expressed as follows (we abbreviate `TakesCourse` with `Takes`):

$$q(x) = \exists y, z.\texttt{Student}(x) \wedge \texttt{Takes}(x, y) \wedge \texttt{AstronomyCourse}(y) \wedge \texttt{Takes}(x, z) \wedge \texttt{CSCourse}(z)$$

Observe that the variables y, z corresponding to the pair of courses used to satisfy the query condition are existentially quantified ($\exists$), which means they are not returned as part of the answer and only the values assigned to x (designating the student) will be returned in the output. Conjunctive queries correspond to the SELECT-PROJECT-JOIN fragment of SQL, the query language adopted by relational database systems. They also form the core of SPARQL Prud'hommeaux and Seaborne (2008), the standard query language for querying RDF data, which offers additional features (such as path expressions and aggregation) and constitutes another query language accepted by many OBDA systems.

7. What do you mean by query answer, and how are these answers computed?

As mentioned earlier, in the OBDA approach, we want to exploit all of the available information: in the ontology, the mappings (if present), and the data sources. We are thus interested in computing the *certain answers*, which are the tuples that are guaranteed to satisfy the query conditions based upon the provided information. So for instance, if we want to query for `Professors`, and the data contains $\texttt{AssistantProfessor}(pers357)$, then $pers357$ would be a certain answer as we know that she is professor and, due to the ontology, that every `AssistantProfessor` is also a `Professor`. Likewise, if the data contains $\texttt{Teaches}(pers226, cs500)$ and $\texttt{GraduateCourse}(cs500)$, then we can return $pers226$, using the ontology axiom stating that graduate courses must be taught by professors. While we omit the details here, we note that the notion of certain answer can be formalized using logical entailment, and that every certain answer can be *explained* by pinpointing the facts, ontology axioms, and mapping rules (if present) that together entail the answer.

While there exist many different algorithms for computing certain answers in OBDA, they mostly follow one of two algorithmic paradigms (or a combination of the two):

Query Rewriting The idea is to reduce the task of computing certain answers to the conceptually and algorithmically simpler task of computing the answers to database queries (without an ontology). This is done by (i) a rewriting step in which the input query q, ontology $\mathcal{O}$, and mappings $\mathcal{M}$ (if present) are transformed into a new query q', formulated in some standard database query language, and (ii) an evaluation step, in which q' is evaluated over the data, using existing database systems. To illustrate with a simple example, if the user query asks to 'list all science courses', then the rewritten query might tell the database system to 'return all astronomy courses, all computer science courses, ..., as well as all chemistry courses', as the ontology would provide the information on which kinds of courses are science courses.

Materialization. A completely different approach, alternatively referred to as materialization, saturation, consequence-based reasoning or forward chaining, is to use the

https://doi.org/10.1017/S1743921325104559 Published online by Cambridge University Press

ontology axioms to 'complete' the data by adding all facts that can be inferred. For example, if the data contains `AstronomyCourse`(ast350), then we would add the fact `ScienceCourse`(ast350). This materialization step does not depend on the query and can be done offline as a pre-processing step. When a query arrives, we can forget about the ontology and simply evaluate the (unmodified) query over the completed data.

For simple ontology languages, like RDFS, one may choose to adopt either the query rewriting or materialization approach, whereas other ontology languages might only be able to use one of the approaches or may be best handled using a combination of the two. Query rewriting has the advantage of allowing ontologies to be added on top of existing databases, taking advantage of highly optimized database systems and not requiring any modifications to the data. By contrast, the materialization approach typically offers better performance, as one computes in advance the consequences of the ontology axioms, reducing the computation required at query time. However, if the data changes frequently, then the materialized inferences need to be updated, which incurs additional computational cost. The choice of which approach to use thus depends on the characteristics of the application.

8. What Are Some Prominent Examples of Applications of Ontologies?

To illustrate how ontologies are used in practice, we give two concrete examples of prominent ontologies in the scientific domain:

SNOMED CT The SNOMED CT ontology, maintained by an international non-profit consortium, was designed to provide a comprehensive and rigorously designed shared vocabulary for medicine Donnelly et al. (2006). The ontology comprises over 350,000 terms covering all aspects of clinical healthcare (symptoms, diagnoses, laboratory tests,...) and contains not only subclass axioms to structure the classes into a rich hierarchy but also many definitions and more complex axioms. SNOMED is a widely adopted standard in use in over 80 countries to facilitate access and analysis of healthcare data for patient care and medical research.

Gene Ontology The Gene Ontology, developed by the Gene Ontology Consortium (2023), provides a structured and species-agnostic representation of biological knowledge. The ontology contains about 40,000 concepts about different molecular functions, cellular components, and biological processes. It is primarily used to provide annotations – evidence-based statements relating a particular gene product to ontology terms – thereby making it easier to find and browse for research results. The Gene Ontology has seen considerable uptake in the research community, with 1.6m annotated gene products and upwards of 185k annotated publications.

9. Are there any existing astronomy ontologies?

Compared to the life science area, where numerous ontologies have been developed and are in widespread use, there are relatively few resources available for the astronomy domain. However, the interest of developing ontologies for astronomy has been advocated by researchers such as Borne (2010) and Efthymiou (2022). At present, the most mature resource appears to be the Unified Astronomy Thesaurus (UAT), an open-source, community-supported thesaurus for astronomical concepts and their interrelationships. The UAT builds upon some earlier efforts, and its history, design choices, and aims are described by Frey and Accomazzi (2018). Currently, the UAT contains

https://doi.org/10.1017/S1743921325104559 Published online by Cambridge University Press

about 2300 concepts, organized into 11 top-level categories and structured into a class hierarchy.

10. How can I learn more?

Ontologies are studied by several research communities. The development of new logical languages and algorithms is primarily carried out by researchers working in knowledge representation and reasoning (a subarea of AI) and databases, with the International Conference on the Principles of Knowledge Representation and Reasoning (KR) and the International Description Logic Workshop being good places to look for developments in the area. More practically oriented work, especially using W3C standards like OWL, RDF(S), and SPARQL, demos, and resources are often published at the International Semantic Web Conference (ISWC), the flagship event for the Semantic Web community. Another relevant venue with an interdisciplinary perspective is the International Conference on Formal Ontology in Information Systems (FOIS). Two prominent research schools are the International Reasoning Web Summer School (which publishes lecture notes, which provide useful introductions to the area) and the Interdisciplinary Summer School on Applied Ontology.

To get started, however, it may be better to try out existing tools to see what they can do. For applications requiring fast performance and limited reasoning, it makes sense to start by considering RDFS, which is supported by many implemented systems; for a comparison of systems and features, see Ali et al. (2022). The Protégé ontology editor (and the accompanying Pizza tutorial) can help you get acquainted with richer ontologies (and the OWL language) Musen (2015). The Ontop OBDA system Calvanese et al. (2017) implements the query rewriting approach, with mappings, and comes with a tutorial to get started. It is also worth taking a look at recent rule-based systems for OBDA, such as RDFox Nenov et al. (2015), Vadalog Bellomarini et al. (2022), and Nemo Ivliev et al. (2024).

Acknowledgements The author acknowledges support from the ANR AI Chair INTENDED (ANR-19-CHIA-0014).

References

Abiteboul, S., Hull, R., & Vianu, V. 1995, *Foundations of Databases*. Addison-Wesley.

Ali, W., Saleem, M., Yao, B., Hogan, A., & Ngomo, A. N. 2022, A survey of RDF stores & SPARQL engines for querying knowledge graphs. *VLDB J.*, 31(3), 1–26.

Baader, F., Horrocks, I., Lutz, C., & Sattler, U. 2017, *An Introduction to Description Logic*. Cambridge University Press.

Bellomarini, L., Benedetto, D., Gottlob, G., & Sallinger, E. 2022, Vadalog: A modern architecture for automated reasoning with large knowledge graphs. *Inf. Syst.*, 105, 101528.

Bienvenu, M. & Ortiz, M. 2015, Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web Tutorial Lectures*, LNCS 9203, 218–307.

Borne, K. 2010, Astroinformatics: Data-oriented astronomy research and education. *Earth Science Informatics*, 3, 5–17.

Brickley, D. & Guha, R. 2014, RDF Schema 1.1 - W3C Recommendation. Document available at: https://www.w3.org/TR/rdf-schema/

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., & Xiao, G. 2017, Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3), 471–487.

Donnelly, K. *et al.* 2006, SNOMED-CT: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121, 279.

Efthymiou, V. 2022, Cosmontology: Creating an ontology of the cosmos (short position paper). In *Proceedings of the Workshop on Deep Learning for Knowledge Graphs*.

https://doi.org/10.1017/S1743921325104559 Published online by Cambridge University Press

Frey, K. & Accomazzi, A. 2018, The unified astronomy thesaurus: Semantic metadata for astronomy and astrophysics. *The Astrophysical Journal Supplement Series*, 236(1), 24.

Gene Ontology Consortium 2023, The gene ontology knowledge base in 2023. *Genetics*, 224(1).

Gruber, T. 2018, *Ontology*, pp. 2574–2576. Springer.

Ivliev, A., Gerlach, L., Meusel, S., Steinberg, J., & Krötzsch, M. 2024, Nemo: Your Friendly and Versatile Rule Reasoning Toolkit. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning*, 743–754.

Mugnier, M. & Thomazo, M. 2014, An introduction to ontology-based query answering with existential rules. In *Reasoning Web Tutorial Lectures*, LNCS 8714, 245–278.

Musen, M. A. 2015, The protégé project: A look back and a look forward. *AI Matters*, 1(4), 4–12.

Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., & Banerjee, J. 2015, RDFox: A highly-scalable RDF store. In *Proceedings of the 14th International Semantic Web Conference*, 3–20.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Rosati, R. 2008, Linking data to ontologies. *Journal of Data Semantics*, 10, 133–173.

Prud'hommeaux, E. & Seaborne, A. 2008, SPARQL Query Language for RDF - W3C Recommendation. Document available at: https://www.w3.org/TR/rdf-sparql-query/

Cyganiak, R., Wood, D. Lanthaler, M. 2014, RDF 1.1 Concepts and Abstract Syntax. Document available at: https://www.w3.org/TR/rdf11-concepts/

OWL Working Group. 2012, OWL 2 Web Ontology Language Document Overview (Second Edition) - W3C Recommendation. Document available at: https://www.w3.org/TR/owl2-overview/

Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., & Zakharyaschev, M. 2018, Ontology-based data access: A survey. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*.

https://doi.org/10.1017/S1743921325104559 Published online by Cambridge University Press