# Ubergraph: integrating OBO ontologies into a unified semantic graph

James P. Balhoff[1], Ugur Bayindir[2], Anita R. Caron[2], Nicolas Matentzoglu[3], David Osumi-Sutherland[2] and Christopher J. Mungall[4]

[1] *Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC USA*
[2] *European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Cambridge, UK*
[3] *Semanticly, Athens, Greece*
[4] *Environmental Genomics and Systems Biology Division, Lawrence Berkeley National Laboratory, Berkeley CA, 94720*

**Abstract**

OBO library ontologies include a wealth of inter-ontology semantic links, which require OWL reasoning to utilize fully. Merging and reasoning over a large suite of ontologies is resource intensive, and challenging for many users. Here we present Ubergraph, an RDF triplestore and public SPARQL query endpoint, which includes a novel approach to precomputing OWL inferences, stored in a readily traversable knowledge graph. Ubergraph's "relation graphs" allow users to perform SPARQL queries which make use of the semantics of the included ontologies. Ubergraph currently includes 39 OBO library ontologies. We describe several use cases enabled by Ubergraph such as ontology browsing and entailment validation.

**Keywords**

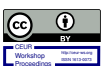
ontology, knowledge graph, reasoning, OWL, RDF, SPARQL, semantic web

## 1. Introduction

The mission of the Open Biological and Biomedical Ontology (OBO) Foundry is to "develop a family of interoperable ontologies that are both logically well-formed and scientifically accurate" [1–3]. Ideally, each ontology in the Foundry covers a specific scope (e.g., animal anatomy, chemistry, cellular processes) and serves as the reference classification for that domain. This allows developers of each ontology to focus on the domain of their expertise, while reusing the work of other experts within their respective domains. Beyond simply dividing up responsibilities for ontology development and reducing overlapping effort, OBO ontologies are encouraged to directly reference concepts from other OBO ontologies within their own logical axioms, allowing automatic classification of terms that makes use of the semantics of those ontologies.

As one example, the Gene Ontology (GO) [4] contains many concepts describing cellular processes acting on particular chemicals, such as various types of metabolism, catabolism, biosynthesis, and transport. As one would expect, GO:0006006 'glucose metabolic process' is a subclass of GO:0005975 'carbohydrate metabolic process'. Rather than manually classifying these GO terms based on their chemical specification, which would need to be done for each kind of GO process, the GO logically defines these concepts using terms from the ChEBI chemical ontology [5], which provides the fact that CHEBI:17234 'glucose' is a type of CHEBI:16646 'carbohydrate'. A standard OWL reasoner can

ICBO 2020, September 25–28, 2022, Ann Arbor, MI, USA
EMAIL: balhoff@renci.org (A. 1); ugur@ebi.ac.uk (A. 2); anitac@ebi.ac.uk (A. 3); nico@semanticly.ai (A. 4); dosumis@ebi.ac.uk (A. 5); cjmungall@lbl.gov (A. 6)
ORCID: 0000-0002-8688-6599 (A. 1); 0000-0002-6012-3729 (A. 2); 0000-0002-6523-4866 (A. 3); 0000-0002-7356-1779 (A. 4); 0000-0002-7073-9172 (A. 5); 0000-0002-6601-2165 (A. 6)



automatically compute the consequent relationships between the GO concepts [6].

OWL axioms making use of external terms typically take the form of complex equivalence definitions bringing together concepts from multiple ontologies. For example, the logical definition of HP:0002446 'Astrocytosis' from the Human Phenotype Ontology [7] refers to terms from four other independently developed OBO ontologies:

```
HP:'Astrocytosis' EquivalentTo
(RO:'has part' some (PATO:'increased
rate' and (RO:'characteristic of part
of' some (GO:'cell growth' and
(RO:'occurs in' some
CL:'astrocyte'))) and (RO:'has
modifier' some PATO:'abnormal')))
```

The meaning of this axiom ("a phenotype including an abnormally increased rate of cell growth in astrocytes") depends on the contents of all five ontologies. As such, for many applications it can be advantageous to treat a collection of OBO library ontologies as one large, unified ontology. One way to accomplish this would be to import all the needed ontologies into a Protégé ontology editing workspace, and run an OWL reasoner. Protégé provides a query interface allowing the user to submit Description Logic (DL) queries to the reasoner; for example, one could find all the cell types (from CL) that are part of the liver (from UBERON): *subclasses* of `(CL:'cell' and (RO:'part of' some UBERON:'liver'))`. By using OWL reasoning, such cell types may match the query indirectly via various kinds of inferences, such as the transitive characteristic of 'part of'.

For a large aggregation of ontologies, that approach inconveniently requires downloading multiple gigabytes of ontologies, providing tens of gigabytes of computer memory, and at least 10–20 minutes of loading and reasoning time using the ELK reasoner. A graph database with a web-based query interface allows users to more efficiently access an integrated semantic knowledge graph. Here we introduce Ubergraph, an RDF triplestore which provides a SPARQL query endpoint to an integrated suite of OBO ontologies, and includes precomputed inferred edges allowing logically complete queries over those ontologies for a subset of OWL.

## 2. Features

The Ubergraph triplestore is a Blazegraph RDF database which is generated weekly from its source ontologies via an open-source workflow. A public SPARQL endpoint (for programmatic access) is available at `https://ubergraph.apps.renci.org/sparql`. Currently, Ubergraph incorporates 39 OBO library ontologies (Fig. 1). The triplestore is organized into a number of different graphs, which enable querying specific ontologies, or including or excluding particular inferences. Each included ontology is stored within a graph named by its ontology IRI. Additional graphs provide precomputed triples which enable a range of functionality making Ubergraph more than simply the sum of the loaded ontologies.

### 2.1. Relation graphs

Two "relation graphs" stored in their own named graphs are at the heart of Ubergraph's utility. Here, a relation graph is an RDF dataset representing certain OWL axioms as convenient graph edges. Many ontology users think of ontologies as directed graphs, containing edges like "index_finger is_a finger" and "finger part_of hand". However, the OWL representation of these statements is somewhat more complex, and particularly complex when stored as an RDF graph. While the first statement is represented by a single straightforward triple:

```
<index_finger> rdfs:subClassOf <finger>
```

the second statement is stored as four triples representing the OWL axiom `<finger> SubClassOf (<part_of> some <hand>)`, using a generated blank node, here, `_:node1`:

```
<finger> rdfs:subClassOf _:node1
_:node1 rdf:type owl:Restriction
_:node1 owl:onProperty <part_of>
_:node1 owl:someValuesFrom <hand>
```

A user writing a SPARQL query for the parts of the hand will need to be aware of the OWL serialization in order to match this complex triple pattern. There are further challenges: a user would expect that when querying for parts of the hand they would receive not only 'finger' but any concepts stated to be parts (e.g., fingernails) or subclasses of 'finger'. SPARQL property paths

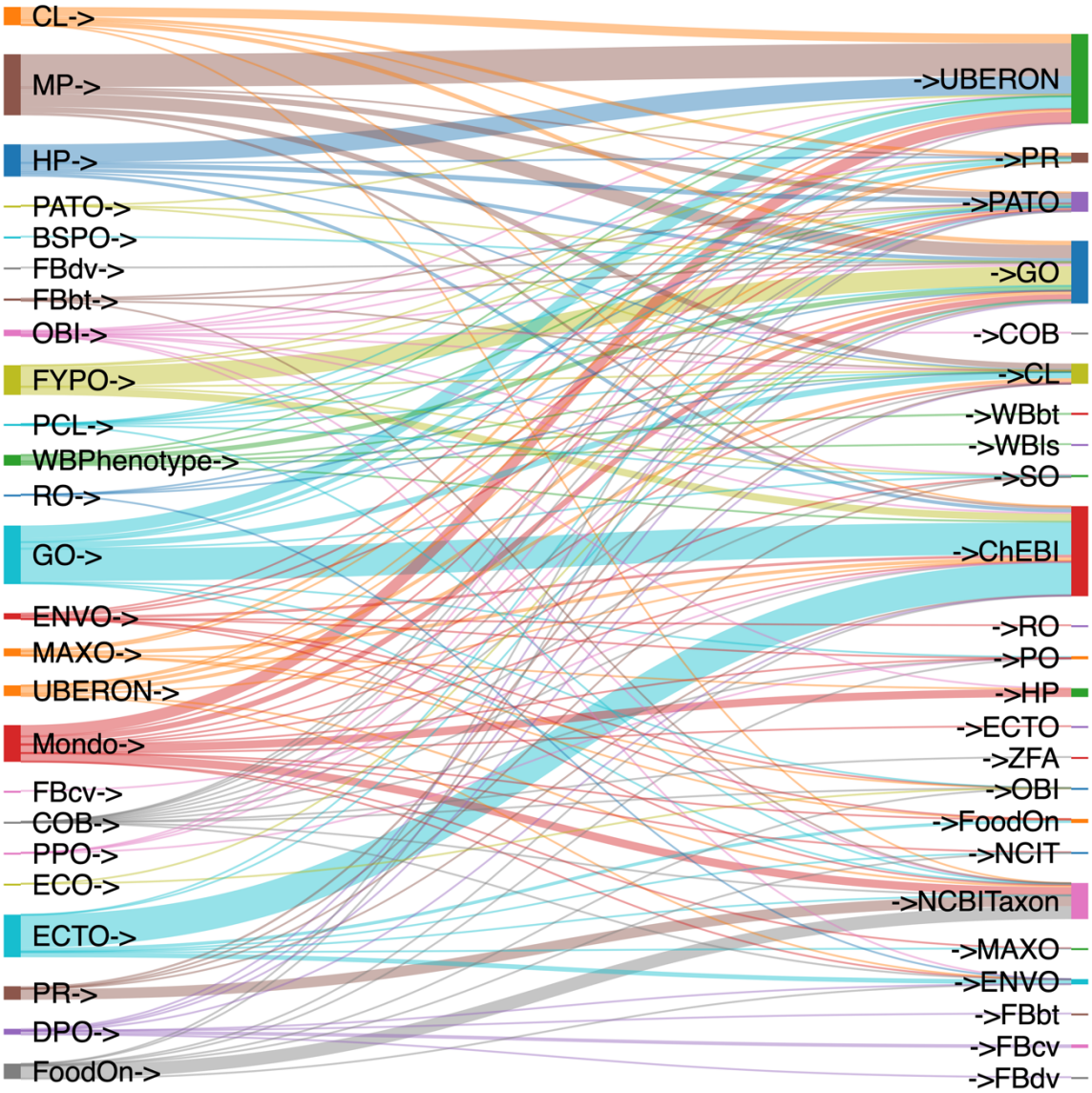



**Figure. 1**: Mutually referential ontologies currently included in Ubergraph. Widths of lines are proportional to the number of OWL classes from each ontology on the right used within axioms in each ontology on the left. Large ontologies on the left side reuse many external terms. Large ontologies on the right side are frequently referenced by other ontologies. Additional OBO ontologies included in Ubergraph but which lack incoming or outgoing references are: APO, EMAPA, MA, MI, MMO, MmusDv.

cannot be employed to retrieve nodes linked by a chain of properties over such OWL expressions. An edge of interest may be embedded deeply within an even more complex OWL equivalence axiom (e.g. 'Astrocyte', above). A user would also expect to find terms such as 'hand_skeleton', related to 'hand' via a different relation such as 'skeleton_of' (an UBERON relation defined as a subproperty of 'part_of').

Relation graphs simplify access to this knowledge by storing the most commonly required OWL patterns as single RDF triples: every SubClassOf relation is represented by a single triple in the standard way, and axioms of the form `<A> **SubClassOf** (<R> **some** <B>)` (existential relations) are converted to simple `<A> <R> <B>` triples:

```
<index_finger> rdfs:subClassOf <finger>
<finger> <part_of> <hand>
```

Beyond this syntactic simplification, the Ubergraph build pipeline uses an OWL reasoner to compute and store every existential relation implied by the input ontology. Given the property hierarchy `<skeleton_of> subPropertyOf`

<part_of> subPropertyOf <overlaps>, all these triples between 'hand_skeleton' and 'hand' would be stored:

```
<hand_skeleton> <skeleton_of> <hand>
<hand_skeleton> <part_of> <hand>
<hand_skeleton> <overlaps> <hand>
```

The relation graph precomputation allows straightforward SPARQL queries to provide fast results consistent with the full semantics of the input ontologies (according to the OWL EL profile). For example, here is a SPARQL query for cell types (CL:0000000) specific to organs (UBERON:0000062) of the abdomen (UBERON:0000916):

```
PREFIX rdfs:
<http://www.w3.org/2000/01/rdf-schema#>
PREFIX cell:
<http://purl.obolibrary.org/obo/CL_0000
000>
PREFIX organ:
<http://purl.obolibrary.org/obo/UBERON_
0000062>
PREFIX abdomen:
<http://purl.obolibrary.org/obo/UBERON_
0000916>
PREFIX part_of:
<http://purl.obolibrary.org/obo/BFO_000
0050>
SELECT DISTINCT ?cell ?organ
WHERE {
  ?cell rdfs:subClassOf cell: .
  ?cell part_of: ?organ .
  ?organ rdfs:subClassOf organ: .
  ?organ part_of: abdomen: .
}
```

The SPARQL query language enables additional kinds of queries unsupported by DL query interfaces, as they require the use of features such as variables or negation as failure, such as:

"Of what is the adrenal gland a part?"

```
adrenal_gland: part_of: ?x
```

"Which bone elements are not entailed to be part of the skeletal system?" (none, in Uberon)

```
?bone rdfs:subClassOf bone_element:
FILTER NOT EXISTS {
  ?bone part_of: skeletal_system: .
}
```

These kinds of queries are useful for checking and exploring the inferences entailed by the input ontologies.

The precomputed subclass and existential relations are stored in two named graphs. The *redundant graph* contains the complete set of entailed relations between terms. The majority of these are not directly asserted in the ontology, but instead implied by equivalence axioms, subproperty axioms, property chains, and other OWL axioms. The *nonredundant graph* contains a subset of the redundant graph. This graph contains only "direct" edges, removing edges such as the following:

- <A> rdfs:subClassOf <C>, where both <A> rdfs:subClassOf <B> and <B> rdfs:subClassOf <C> exist
- <A> <R> <C>, where <A> <R> <B> and <B> rdfs:subClassOf <C> exist
- <A> <R> <C>, where <B> <R> <C> and <A> rdfs:subClassOf <B> exist
- <A> <R> <C>, where <A> <R> <B> and <B> <R> <C> exist, and the ontology declares <R> to be a transitive property
- <A> <S> <B>, where <A> <R> <B> exists, and the ontology declares <R> to be a subproperty of <S>

The nonredundant relation graph is particularly useful for SPARQL CONSTRUCT queries which output the graph neighborhood around a term, such as when powering an ontology browsing interface (Fig. 2).

## 2.2. Ontology graph

An additional named graph stores the result of classifying the merged axioms of the input ontologies using the ELK reasoner [8]. This graph includes all the term annotation axioms (such as labels and definitions) provided by the source ontologies. It also includes a generated triple for each term, connecting it to its source ontology based on its OBO ID space, e.g., obo:UBERON_4100121 rdfs:isDefinedBy obo:uberon.owl. These triples allow much more efficient filtering of results by ID space as compared with string-based SPARQL filters. The ontology graph also includes triples linking each term to a computed information content score (terms with a greater number of subclasses have lower information content). These scores can be helpful in ranking Ubergraph query result values,

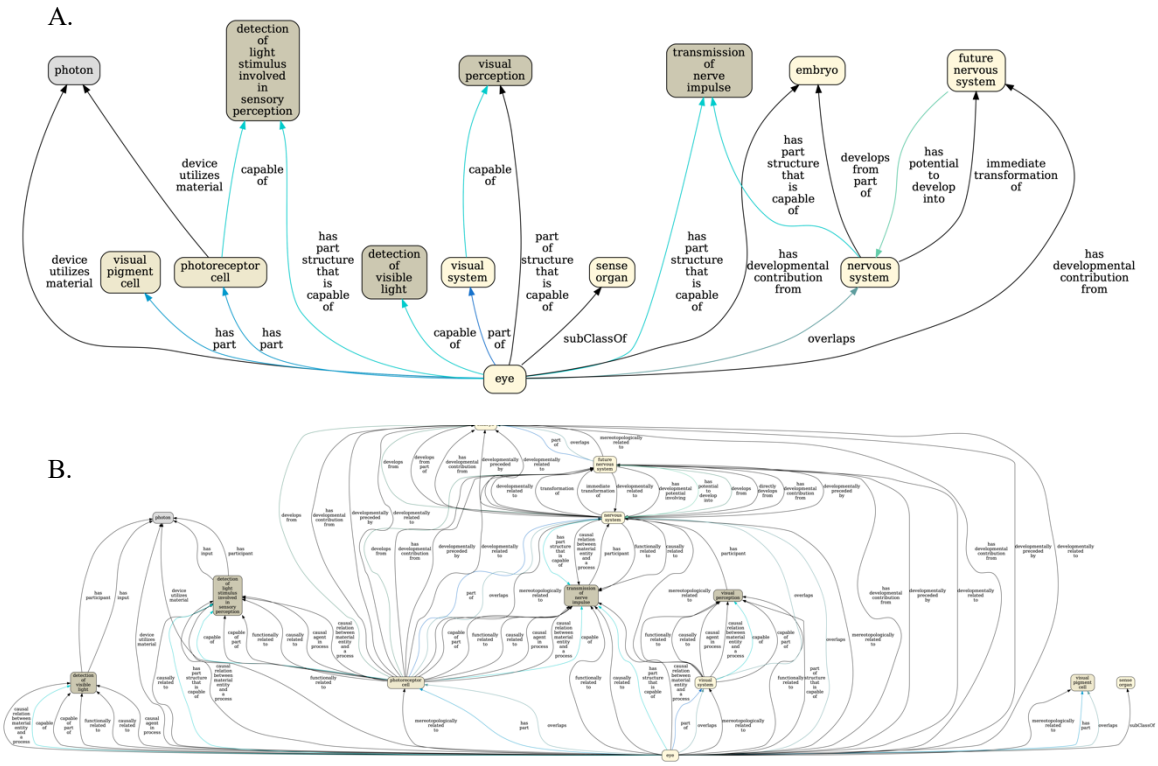



**Figure. 2**: (A) Nodes directly connected to UBERON:0000970 'eye', and edges between them, in the Ubergraph nonredundant graph. (B) Edges in the redundant graph between those same nodes. A depiction of edges between all nodes reachable by one hop from 'eye' in the redundant graph would be even more tangled, including 10,262 edges.

based on their relative graph placement, within downstream applications.

## 2.3. Biolink Model graph

The Biolink Model is a high level data model of biological entities [9]. Ubergraph includes a graph containing an RDF serialization of the Biolink Model, as well as `biolink:category` links from each ontology term to mapped Biolink Model classes, propagated across the subclass hierarchy of both the Biolink Model and the included ontologies. This graph is provided for use by applications built on the Biolink Model, to easily traverse from OBO ontology concepts to corresponding Biolink Model terms.

# 3. Implementation

Ubergraph is constructed using a workflow implemented as a GNU Makefile [10]. The source code is available on GitHub at https://github.com/INCATools/ubergraph. The workflow downloads the source ontologies and computes additional RDF triples constituting the relation graphs and other enrichments using a variety of tools included within a Docker image defined in the Ubergraph repository. The endpoint of the workflow is a Blazegraph [11] database file used to drive the SPARQL endpoint. The public SPARQL service is a Blazegraph server running within an on-premises Kubernetes cluster, provided with 32 GB memory and 8 CPUs.

## 3.1. Merging and reasoning over an integrated set of ontologies

Ontology manipulation is conducted using ROBOT [12] and the Apache Jena [13] 'arq' and 'riot' tools. When available, the Ubergraph build downloads the "base" release of each ontology (e.g., for Uberon, `http://purl.obolibrary.org/obo/uberon/uberon-base.owl`). Base files are a recently adopted convention within the OBO Foundry which facilitate merging and reasoning over sets of ontologies which refer to one another within their axioms. The standard releases of each ontology typically contain imported content from the ontologies they depend on. Merging these files can result in mixtures of axioms and annotations

from slightly different releases of each ontology, possibly resulting in conflicting logical assertions. Base files, on the other hand, contain the axioms native to a given ontology but exclude any imported content. When a base file is not available for an ontology, the Ubergraph build uses the ROBOT tool to approximate a base file for that ontology by removing axioms defining terms from external ID spaces.

## 3.2. Computing relation graphs

The redundant relation graph is computed via a purpose-built tool, 'relation-graph' [14], which uses an efficient, parallel algorithm to perform millions of DL queries using the Whelk OWL reasoner [15]. This process takes approximately 4 hours, using 20 CPUs and 140 GB RAM. The nonredundant relation graph is computed from the redundant graph using a Soufflé Datalog [16] pruning script which implements the redundancy rules described above, taking approximately 70 minutes to complete.

The current Ubergraph release contains 530,834,705 triples, with 48,684,904 triples comprising the included ontologies, and the rest derived by the build process. The majority of these, 318,231,131, constitute the redundant relation graph. The pruning step reduces that number to 4,532,758 triples in the nonredundant relation graph.

# 4. Applications

## 4.1. Programmatic Access

As a standard SPARQL endpoint, Ubergraph can be accessed using any of the typical programmatic means of reaching a REST endpoint, as well as dedicated SPARQL query interfaces such as Yasgui [17]. In addition to generic REST libraries, languages like Python include clients for SPARQL endpoints such as SPARQLWrapper.

For higher level access, it is possible to write software libraries that act as clients to Ubergraph and provide convenient reusable operations. One such library is the new OAK (Ontology Access Kit) library in Python, which provides programmatic access to a variety of endpoints, including Ubergraph [18]. OAK provides both a Python layer and a command line interface (CLI) that allows for various operations over Ubergraph including (1) searching for terms; (2) performing semantic similarity; and (3) accessing and visualization of subgraphs, making use of the obographviz package. For example, the OAK viz command, e.g., `runoak -i ubergraph: viz GO:0001750 -p rdfs:subClassOf,BFO:0000050`, will show all terms traversable via the relation graph from a starting set of terms such as 'photoreceptor outer segment' in GO (Fig. 3).

One additional way to access Ubergraph is through its OpenAPI REST endpoint description [19], autogenerated (using grlc [20]) from a suite of example SPARQL queries stored in the Ubergraph source repository.

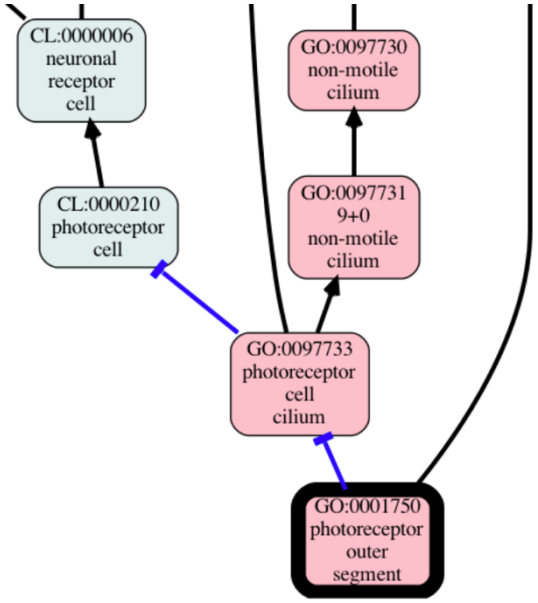



**Figure 3**: OAK visualization of relation graph ancestors of a GO term, traversing inter-ontology links. CL terms are gray, GO terms are pink. Is_a (SubClassOf between named classes) edges in black, part_of (BFO:0000050) in blue.

## 4.2. Projects using Ubergraph

### 4.2.1. Mondo quality control checks

The Mondo disease ontology [21] is a complex development effort that seeks to integrate disparate disease ontologies and terminologies. An automated process creates a skeleton for a large ontology that integrates axioms and annotations from these sources, which is further augmented by complex logical axioms using design patterns [22]. Such an ontology requires extensive quality control checking and reporting

capabilities that need to be reviewed, refined, and shared. Moreover, these reports often span multiple ontologies such as Uberon, GO, and CL. All Mondo reports and quality control checks [23] are implemented using SPARQL. To develop and share these reports, the Mondo team uses Yasgui in combination with the Ubergraph SPARQL endpoint: queries are prototyped, shared between developers, and refined until fit for purpose. The Mondo team makes extensive use of inferred relationships in Ubergraph, which not only reduces query time, but also improves recall for cases where subclasses are only inferable through an entailment regime such as OWL EL.

### 4.2.2. HuBMAP validation

The HuBMAP project is building a human reference atlas leveraging expert input [24]. The atlas covers adult human anatomy. Uberon has many terms that are not useful for this purpose as they refer to other species or developmental stages. It also has many more relationship types than needed. Experts working on this project provide their view of human anatomy using spreadsheets to relate Uberon and Cell Ontology terms. HuBMAP developers use Ubergraph to programmatically test the validity of expert-specified relationships between term pairs against subclass and existential relations from a small set of high level object properties. Reports of non-validating pairs are used to inform corrections to the expert-curated tables, or to improve Uberon and Cell Ontology as applicable. As an example, HuBMAP biologists have mapped 'OFF-bipolar cell' (CL:0000750) to 'inner nuclear layer of retina' (UBERON:0001791), a relation that is currently not present in the Cell Ontology. The validation tool searches for relationships among the terms in the HuBMAP domain and finds that 'OFF-bipolar cell' currently does have a 'part_of' relationship with 'retina' (UBERON:0000966). Having these suggestions and relationships visualized via the validation tool allows editors to consider remodeling the Cell Ontology to have more specific mappings. In this case, an editor may choose to add the following axiom to 'OFF-bipolar cell': `'part of' some 'inner nuclear layer of retina'`.

### 4.2.3. Biomedical Data Translator

The NCATS Biomedical Data Translator program is creating a federated knowledge system capable of integrating existing biomedical data sets, and which will allow users to derive "insights that can accelerate translational research, support clinical care, and leverage clinical expertise to drive research innovations" [25]. Many of the Data Translator knowledge sources express their data with reference to standard identifiers for terms from OBO library ontologies, such as cell types, anatomical locations, and diseases. The ontologies themselves provide the background knowledge giving meaning to the use of those terms. Ubergraph provides the basis for the Ontology Knowledge Provider, a Translator knowledge source which implements the Data Translator knowledge graph API via queries to the Ubergraph SPARQL endpoint [26]. As described above, nodes within Ubergraph are pre-categorized using groupings from the Biolink standard, which is used as a top-level data model by Translator.

## 5. Challenges

Reasoning over a merged collection of mutually referential, but independently developed, ontologies can uncover hidden logical incompatibilities. Some of these incompatibilities are simply the result of stale imported content; while these ontologies are all developed as part of the OBO collaborative community, they move at varying paces and release schedules. In our experience, the use of "base files" (discussed above) helps to avoid many such issues that were frequently encountered in previous attempts to reason across combinations of OBO ontologies, e.g., [27]. Further, combining ontologies together in applications like Ubergraph highlights the need to ensure that quality control checks in ontology release pipelines consider a comprehensive set of external axioms; otherwise they may miss undesired entailments from the use of particular terms. The Ubergraph build pipeline applies additional preprocessing, such as removing disjointness axioms, to minimize the effect of any remaining logical incoherency.

Another issue encountered in a reasoning application like Ubergraph is that the inferences computed from the merged set of ontologies may result in additional intra-ontology subsumptions that individual ontology providers have not vetted. In a real-life example, a new release of ChEBI classified some chemicals as lipids which were not previously so. Combining this release of ChEBI with the logical definitions for metabolic

processes provided by the Gene Ontology resulted in certain GO processes being classified under 'lipid metabolism' which were not grouped as such in the official GO release. To better avoid such discrepancies, but still provide all the inferred relations which make Ubergraph so useful, we are developing protocols for axiom inclusion in the Ubergraph reasoning process, such as transformations of equivalent class axioms into less powerful subclass axioms, and ensuring that the class hierarchy published by each provider is precomputed in the ontology file incorporated into Ubergraph.

## 6. Related work

Online ontology repositories such as Ontobee [28], the EBI Ontology Lookup Service [29], and BioPortal [30] all provide SPARQL endpoints which allow querying over the OBO ontologies they include. However, within these services, users have access to only the complicated RDF serialization of the OWL axioms. Ubergraph's precomputed relation graphs both greatly simplify, and also significantly increase the semantic power of, SPARQL queries over the combined suite of ontologies.

## 7. Conclusions

OBO library ontologies contain a wealth of cross-domain knowledge within their logical axioms. Ubergraph provides a powerful means to access and make use of these connections in a way that preserves and utilizes the full semantics of its constituent ontologies, without requiring users to download multiple gigabytes of ontologies and load these into an OWL reasoner.

## 8. Acknowledgments

This work was supported in part by the U.S. NIH project numbers 5U01HG009453-03 and 3OT2TR003449-01S1.

## 9. References

[1] The Open Biological and Biomedical Ontology (OBO) Foundry. In: The Open Biological and Biomedical Ontology (OBO) Foundry [Internet]. [cited 25 May 2022]. Available: https://obofoundry.org/

[2] Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotechnol. 2007;25: 1251–1255. doi:10.1038/nbt1346

[3] Jackson R, Matentzoglu N, Overton JA, Vita R, Balhoff JP, Buttigieg PL, et al. OBO Foundry in 2021: operationalizing open data principles to evaluate ontologies. Database . 2021;2021. doi:10.1093/database/baab069

[4] Gene Ontology Consortium. The Gene Ontology resource: enriching a GOld mine. Nucleic Acids Res. 2021;49: D325–D334. doi:10.1093/nar/gkaa1113

[5] Hastings J, Owen G, Dekker A, Ennis M, Kale N, Muthukrishnan V, et al. ChEBI in 2016: Improved services and an expanding collection of metabolites. Nucleic Acids Res. 2016;44: D1214–9. doi:10.1093/nar/gkv1031

[6] Hill DP, Adams N, Bada M, Batchelor C, Berardini TZ, Dietze H, et al. Dovetailing biology and chemistry: integrating the Gene Ontology with the ChEBI chemical ontology. BMC Genomics. 2013;14: 513. doi:10.1186/1471-2164-14-513

[7] Köhler S, Gargano M, Matentzoglu N, Carmody LC, Lewis-Smith D, Vasilevsky NA, et al. The Human Phenotype Ontology in 2021. Nucleic Acids Res. 2021;49: D1207–D1217. doi:10.1093/nar/gkaa1043

[8] Kazakov Y, Krötzsch M, Simančík F. The Incredible ELK. J Automat Reason. 2013;53: 1–61. doi:10.1007/s10817-013-9296-3

[9] Unni DR, Moxon SAT, Bada M, Brush M, Bruskiewich R, Clemons P, et al. Biolink Model: A Universal Schema for Knowledge Graphs in Clinical, Biomedical, and Translational Science. arXiv [cs.DB]. 2022. doi:10.48550/arXiv.2203.13906

[10] Make - GNU Project - Free Software Foundation. [cited 27 May 2022]. Available: https://www.gnu.org/software/make/

[11] Blazegraph. In: Blazegraph [Internet]. 2015 [cited 29 May 2016]. Available: https://github.com/blazegraph/database

[12] Jackson RC, Balhoff JP, Douglass E, Harris NL, Mungall CJ, Overton JA. ROBOT: A Tool for Automating Ontology Workflows. BMC Bioinformatics. 2019;20: 407. doi:10.1186/s12859-019-3002-3

[13] Apache Jena. [cited 27 May 2022]. Available: https://jena.apache.org/

[14] Balhoff JP. relation-graph. Github; Available: https://github.com/balhoff/relation-graph
[15] Balhoff JP. Whelk. Github; Available: https://github.com/balhoff/whelk
[16] Soufflé. [cited 27 May 2022]. Available: https://souffle-lang.github.io/index.html
[17] Yasgui. In: Triply [Internet]. 25 Jun 2019 [cited 27 May 2022]. Available: https://triply.cc/docs/yasgui
[18] Mungall C, Harshad, Kalita P, Patil S, Joachimiak M p., Caufield H. INCATools/ontology-access-kit: v0.1.18. 2022. doi:10.5281/zenodo.6574927
[19] OpenAPI Specification v3.1.0. [cited 29 May 2022]. Available: https://spec.openapis.org/oas/latest.html
[20] Peñuela AM. grlc. [cited 29 May 2022]. Available: https://grlc.io/
[21] Vasilevsky NA, Matentzoglu NA, Toro S, Flack JE IV, Hegde H, Unni DR, et al. Mondo: Unifying diseases for the world, by the world. medRxiv. 2022. doi:10.1101/2022.04.13.22273750
[22] Osumi-Sutherland D, Courtot M, Balhoff JP, Mungall C. Dead simple OWL design patterns. J Biomed Semantics. 2017;8: 18. doi:10.1186/s13326-017-0126-0
[23] Quality control tests - Mondo Documentation. [cited 26 May 2022]. Available: https://mondo.readthedocs.io/en/latest/editors-guide/quality-control-tests/
[24] Börner K, Teichmann SA, Quardokus EM, Gee JC, Browne K, Osumi-Sutherland D, et al. Anatomical structures, cell types and biomarkers of the Human Reference Atlas. Nat Cell Biol. 2021;23: 1117–1128. doi:10.1038/s41556-021-00788-6
[25] Biomedical Data Translator Consortium. Toward A Universal Biomedical Data Translator. Clin Transl Sci. 2019;12: 86–90. doi:10.1111/cts.12591
[26] ontology-kp. [cited 27 May 2022]. Available: https://github.com/TranslatorSRI/ontology-kp
[27] Slater LT, Gkoutos GV, Hoehndorf R. Towards semantic interoperability: finding and repairing hidden contradictions in biomedical ontologies. BMC Med Inform Decis Mak. 2020;20: 311. doi:10.1186/s12911-020-01336-2
[28] Xiang, Mungall, Ruttenberg, He. Ontobee: A linked data server and browser for ontology terms. ICBO. 2011. Available: http://ceur-ws.org/Vol-833/paper48.pdf
[29] Jupp, Burdett, Leroy, Parkinson. A new Ontology Lookup Service at EMBL-EBI. SWAT4LS. 2015. Available: http://ceur-ws.org/Vol-1546/paper_29.pdf
[30] Salvadores M, Alexander PR, Musen MA, Noy NF. BioPortal as a Dataset of Linked Biomedical Ontologies and Terminologies in RDF. Semantic Web. 2013;4: 277–284. Available: https://www.ncbi.nlm.nih.gov/pubmed/25214827