Database, 2025, baae133
DOI: https://doi.org/10.1093/database/baae133
Original article

A change language for ontologies and knowledge graphs

Harshad Hegde[1], Jennifer Vendetti[2], Damien Goutte-Gattat[3], J. Harry Caufield[1], John B. Graybeal[2], Nomi L. Harris[1], Naouel Karam[4], Christian Kindermann[2], Nicolas Matentzoglu[5], James A. Overton[6], Mark A. Musen[2], Christopher J. Mungall[1,*]

[1]Environmental Genomics and Systems Biology, Lawrence Berkeley National Laboratory, One Cyclotron Rd., Berkeley, CA 94720, United States
[2]Center for Biomedical Informatics Research, Stanford University, 3180 Porter Dr., Palo Alto, CA 94304, United States
[3]Department of Physiology, Development and Neuroscience, University of Cambridge, Downing Street, Cambridge CB2 3DY, United Kingdom
[4]Institute for Applied Informatics (InfAI), Leipzig University, Goerdelerring 9, Leipzig 04109, Germany
[5]Semanticly, Spaces Ermou, Ermou 56, Athens 10563, Greece
[6]Knocean Inc., 2 - 107 Quebec Ave., Toronto, Ontario M6P 2T3, Canada

[*]Corresponding author. Environmental Genomics and Systems Biology, Lawrence Berkeley National Laboratory, One Cyclotron Rd., Berkeley, CA 94720, United States. E-mail: cjmungall@lbl.gov

Citation details: Hegde, H., Vendetti, J., Goutte-Gattat, D. *et al.* A change language for ontologies and knowledge graphs. *Database* (2025) Vol. 2025: article ID baae133; DOI: https://doi.org/10.1093/database/baae133

Abstract

Ontologies and knowledge graphs (KGs) are general-purpose computable representations of some domain, such as human anatomy, and are frequently a crucial part of modern information systems. Most of these structures change over time, incorporating new knowledge or information that was previously missing. Managing these changes is a challenge, both in terms of communicating changes to users and providing mechanisms to make it easier for multiple stakeholders to contribute. To fill that need, we have created KGCL, the Knowledge Graph Change Language (https://github.com/INCATools/kgcl), a standard data model for describing changes to KGs and ontologies at a high level, and an accompanying human-readable Controlled Natural Language (CNL). This language serves two purposes: a curator can use it to request desired changes, and it can also be used to describe changes that have already happened, corresponding to the concepts of "apply patch" and "diff" commonly used for managing changes in text documents and computer programs. Another key feature of KGCL is that descriptions are at a high enough level to be useful and understood by a variety of stakeholders—e.g. ontology edits can be specified by commands like "add synonym 'arm' to 'forelimb'" or "move 'Parkinson disease' under 'neurodegenerative disease'." We have also built a suite of tools for managing ontology changes. These include an automated agent that integrates with and monitors GitHub ontology repositories and applies any requested changes and a new component in the BioPortal ontology resource that allows users to make change requests directly from within the BioPortal user interface. Overall, the KGCL data model, its CNL, and associated tooling allow for easier management and processing of changes associated with the development of ontologies and KGs.

Database URL: https://github.com/INCATools/kgcl

Introduction

Ontologies are structures that encode concepts and entities in a particular domain in a way that facilitates data standardization as well as a wide variety of inferential tasks. They are crucial to many modern information systems. Ontologies, such as the Gene Ontology (GO), are used daily to interpret high-throughput experimental data. Anatomic and cell type ontologies, such as the Cell Ontology [1] and Uberon [2], are crucial for projects like HuBMAP [3] and the Human Cell Atlas [4] that aim to provide a molecular map of the bodies of humans and other organisms. In the clinical realm, ontologies, such as the Human Phenotype Ontology [5], are being used to standardize the representation of phenotypes in patients and for supporting applications such as phenotype-based variant prioritization. Ontologies can be distributed and accessed via portals such as BioPortal [6], OntoBee [7], and the Ontology Lookup Service [8].

Ontologies often have a graph-like structure and are frequently incorporated into knowledge graphs (KGs), which augment the textbook knowledge in an ontology with additional evidence-backed knowledge about individual entities. Examples of KGs in the biosciences include Hetionet [9], PheKnowLator [10], and KG-COVID-19 [11]. Ontologies and KGs are similar structures, with different emphases. Ontologies may emphasize expressive logical structures, and KGs typically emphasize simpler interconnections.

While ontologies and KGs vary tremendously in their applications, structure, and formalisms, a common underlying element is "change." They are not static. It is rare for a domain to be represented with perfect accuracy and completeness on the first attempt; instead, there is a constant process of refinement, as part of a complete lifecycle, involving many stakeholders. Figure 1, which shows the number of changes in GO terms between releases, illustrates this continual pro-

Received 20 September 2024; Revised 21 November 2024; Accepted 30 December 2024

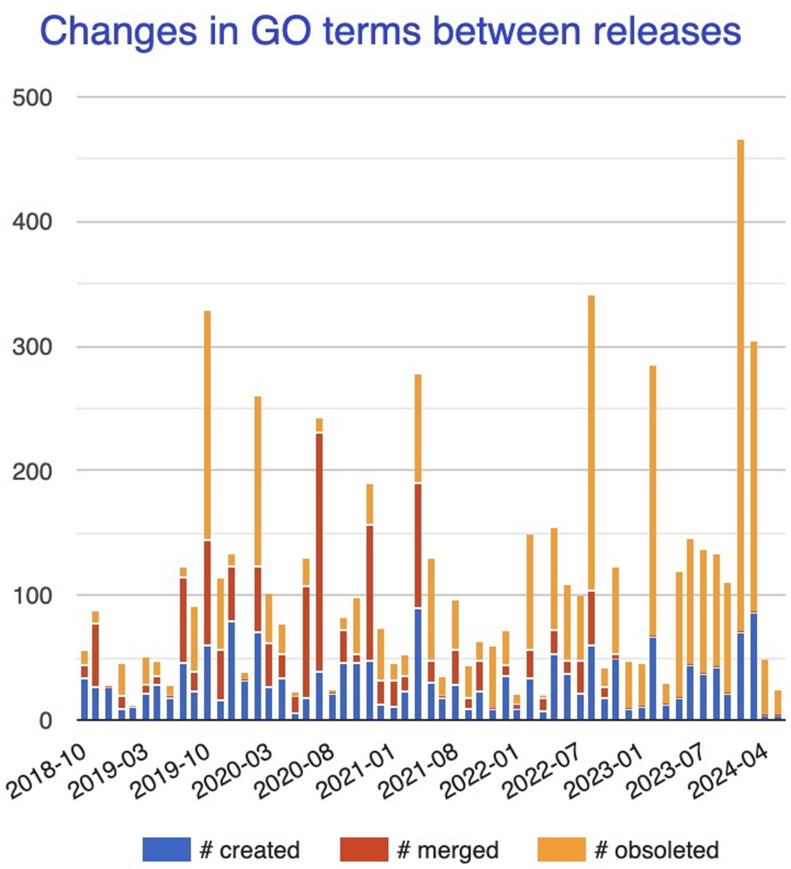

Figure 1. Changes in GO terms (number created, merged, or obsoleted) between major releases between late 2018 and early 2024.

cess of change in response to new knowledge. GO was first released over 20 years ago, yet its rate of change remains high and even increases, as the available pool of relevant knowledge continues to grow.

Changes to ontologies generally fall into several categories: adding a new element, obsoleting an existing element, merging two elements, and adding/deleting/modifying information associated with an element (such as definitions, synonyms, or taxon restrictions). Changes can occur in response to new knowledge, new terminology, improved modeling of knowledge, or simply to correct previous errors. Changes can be instigated by requests from the community, but they are typically applied by expert curators and ontology editors.

Despite the constant factor of change, there is surprisingly no single agreed-upon way to "represent or communicate" ontology changes. This is a major technical and communication obstacle for KGs and ontologies. This obstacle is presented in two ways (Fig. 2). The first way is how changes that have been made "retrospectively" are communicated to stakeholders (i.e. "diffs"). This is shown on the left side of Fig. 2: comparing two ontologies O1 and O2 via a "diff" operation yields a Change object representing the most parsimonious set of changes to go from O1 to O2; in this case, the change is moving Node E from under Node C to under Node B. The "diff" operation can be used to describe retrospective changes that have happened to an ontology over time.The second way is how contributors communicate "prospective" desired changes to the KGs (i.e. "patches"). An example of a prospective change is shown on the right side of Fig. 2: "applying" a Change object to O1 yields O2. T The "apply" operation can be used to describe intended changes to an ontology prospectively. A typical workflow is for a domain expert or curator to ask for a change in natural language, sometimes in the form of an issue/ticket in the GitHub project for an ontology. A specialized ontology editor then translates this request into a sequence of actions in an ontology development environment, such as Protégé [12, 13] or WebProtege [14]. This is a repetitive and time-consuming task with many inefficiencies, and it relies on the availability of ontology editors to process these change requests. Another challenge in ontology editing is that different ontologies implement different workflows. Sometimes these are documented, and sometimes one ontology will partially or fully adopt procedures from another ontology. There is an overall lack of commonality, which makes it harder to automate and, for curators, to transfer practice from one ontology to another. Standardizing these processes can reduce the number of mistakes and increase efficiency.

Community need for a change language and associated tools

To better understand what curators and ontology developers need from a change language, we hosted a virtual workshop in 2023 on change languages in ontologies. In order to scope the workshop, we focused on the kinds of biological ontologies found in ontology repositories such as BioPortal and, in

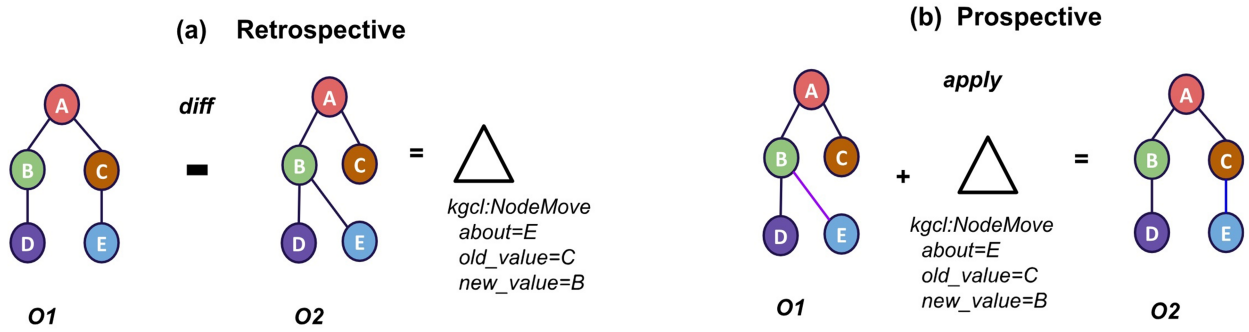

Figure 2. Logic of ontology changes.

particular, ontologies that are part of the OBO Foundry [15]. Before and during the workshop, we surveyed the community of biomedical ontology users and developers on a range of topics [16], including what tools they used to generate ontology diffs, how satisfied they were with those tools, and how they made changes to ontologies. We also demonstrated and solicited feedback on a proposed standard for representing ontology changes [which became the Knowledge Graph Change Language (KGCL)].

In our survey of ontology users, when asked how important it was for them to stay informed about changes to ontologies they use, 82% rated it as extremely or very important. The two tools that participants mentioned for generating diffs were the ROBOT ontology tool [17] and the Bubastis ontology diff tool [18]. ROBOT offers an ontology diff operation among its many different ontology processing operations and can operate over ontologies in OBO format or any OWL syntax. Bubastis is a dedicated ontology diffing tool that has been integrated into BioPortal and other OntoPortal end points. It is automatically executed for each new ontology release, allowing users to easily download and view changes. Both operate by performing setwise diff of OWL axioms, which gives a precise computable representation of changes, but at a lower level than how many ontology developers conceive of changes. For example, the NodeMode operation in Fig. 2 would be represented as two changes, a deletion and an insertion. In our surveys, users reported that they would be more satisfied with a higher-level representation and presentation of changes.

In addition to ROBOT and Bubastis, other ontology diffing frameworks include the QuickGO Change Log [19], GOtrack [20], and the COnto-diff framework [21]. Although these were not mentioned by survey participants in the list of tools they use, these three frameworks provide complementary and powerful ways of viewing and understanding changes in ontologies. Both the QuickGO Change Log and GOtrack are specific to the GO. The QuickGO Change Log is integrated into the QuickGO ontology browser and allows users to see changes to the GO in the context of other information and annotations about that term. GOtrack allows users to analyze the impact of changes on the GO. COnto-diff provides ontology diffing for any OWL ontology and pioneered the use of a taxonomy or classification of change types (see the "Aligning with related work" section).

Based on feedback from the workshop participants, there was a clear need for a standardized, high-level way of representing changes in ontologies. A total of 82% of survey respondents said that it was very important (4 or 5 on a scale of 1–5) for them to understand changes in the ontologies they used, while only 8% were satisfied with the current methods for viewing such changes. The survey results and direct feedback from workshop participants showed that there was a strong desire for improved methods for describing changes to automated agents that could apply these changes seamlessly to existing ontologies. We realized that a thoughtfully designed approach to encoding ontology changes could extend beyond ontologies and be applicable to knowledge bases and KGs more generally.

The change language workshop also covered the application of changes to ontologies. When we surveyed the community of people who use and/or build ontologies and asked how they request changes in an ontology, almost all of them were following the workflow just described, although some were technically skilled enough to make Pull Requests (PRs) in the ontology repository to accomplish the desired changes. Only 17% said that they were very or extremely satisfied with the turnaround time for the changes to happen. This dissatisfaction makes it clear that the current human-powered process of making ontology changes is not sufficient to keep up with demand and that mechanisms to speed up the process are needed.

Knowledge Graph Change Language provides a standard for describing ontology changes

To address the need for a standardized way to express changes in ontologies, we created KGCL. KGCL can represent common ontology editing operations (such as modifying a label or a definition, obsoleting a term, moving a term under another parent term, etc.). To represent these operations unambiguously and computably, we designed a Controlled Natural Language (CNL), which is a subset of a natural language (in this case, English) that restricts the grammar and vocabulary to reduce ambiguity and complexity. This CNL is designed to be as close to natural language (US English) as possible, yet to be unambiguously parseable by machines. As an example, the KGCL command to change the name of the ontology term with the ID ENVO:01000575 from "wax" to "oil" is "rename ENVO:01000575 from 'wax' to 'oil.'"

KGCL consists of three primary components (Fig. 3):

(i) A schema and taxonomy (classification) of change types
(ii) A CNL for specifying ontology changes
(iii) Multiple serialization formats such as JSON, YAML, and RDF, as well as tabular formats for use in spreadsheets.

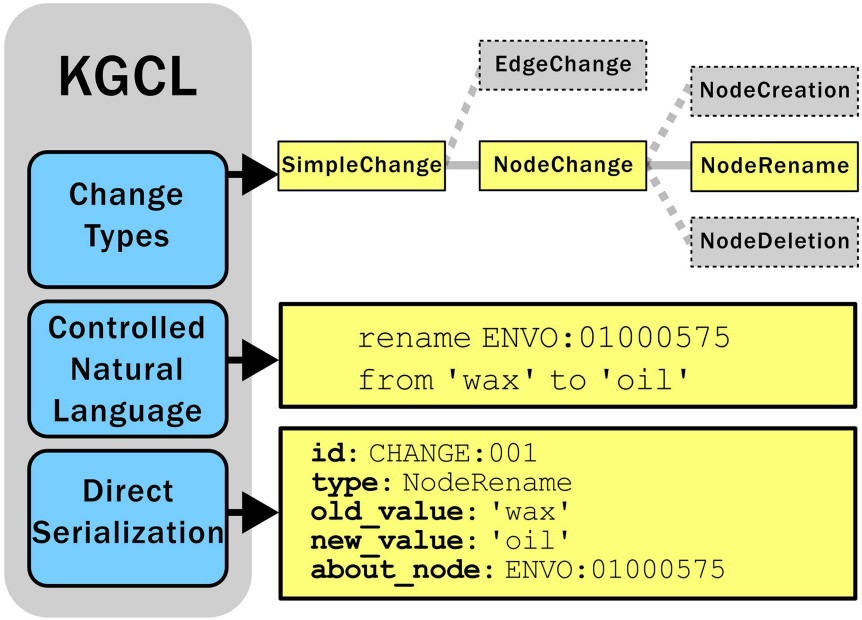

Figure 3. Overview of the three components of KGCL: (a) A classification of change types (here showing that NodeRename is a subtype of NodeChange and a sibling of NodeCreation and NodeDeletion); (b) A CNL for expressing changes in a simple human-readable yet computable syntax; (c) A data model that can be directly serialized in different syntaxes (here showing a NodeRename instance serialized as YAML).

A classification of types of ontology changes

KGCL organizes different kinds of changes into a classification hierarchy, such that all changes that affect terms (nodes) are in one branch and all changes that affect relationships (edges) are in another branch—see Table 1. The classification and terminology are also available to browse in BioPortal (https://bioportal.bioontology.org/ontologies/KGCL) and are available from the PURL https://w3id.org/kgcl/kgcl.owl.ttl.

Change data model

We expressed the aforementioned hierarchy in a semantic data model. The data model describes the attributes of each change type. Some attributes, such as the "id" attribute (a unique identifier for tracking each change), are shared across all change types. Other attributes are specific to individual types of change.

For example, as shown in Fig. 3, the NodeRename class has an attribute "about_node" (shared by all NodeChange objects, describing the node or term to be acted on), as well as "old_value" and "new_value," describing the name/label to be changed and the replacement name/label.

We use LinkML [22] to express the KGCL data model. LinkML is an open-source data modeling framework that provides flexible modeling features such as class hierarchies, mappings to other models, a rich semantic framework, and inline documentation for all model objects. LinkML also allows the data model to be expressed using other technologies, such as OWL or JSON-Schema, and LinkML tooling generates the KGCL website (https://w3id.org/kgcl), model diagrams, and documentation directly from the model itself.

Serialization formats and Controlled Natural Language expression

KGCL can be serialized and deserialized using different syntaxes. The canonical syntax for KGCL is the KGCL CNL, which is intended to be easily read and written by humans, but is also parseable by machines. The KGCL CNL is specified by a grammar using the Lark formalism [23]. KGCL can be serialized as JSON, YAML, or RDF, if there is no need for human readability. A tabular form is also available for use in spreadsheets, but this is less expressive than the other forms.

A tool suite for working with Knowledge Graph Change Language

We have developed tools aimed at ontology developers, curators, and software developers to help with common tasks related to change management. These tools include an automated agent (Ontobot) that waits for requests from curators on GitHub issue trackers and then enacts these changes on an ontology, a widget for the BioPortal ontology portal that allows users to make change requests in the BioPortal user interface, and software libraries and command-line tools in Java and Python that can be used by advanced users.

Ontobot: an automated agent for applying curator change requests

Many ontologies are managed in GitHub [24], with GitHub issues used to manage change requests from users and GitHub PRs to suggest these changes. This is currently a manual and time-intensive process, in which an ontology editor will read through ontology issues, carry out the requested changes using an ontology development tool such as Protégé, and then make a PR, which is later reviewed and merged. We created an automated agent called Ontobot that simplifies this workflow by automating the time-consuming intermediate steps in this workflow.

Ontobot is integrated with GitHub using the GitHub Actions mechanism [25]. GitHub Actions is a lightweight method for automating tasks in a GitHub repository and is

Table 1. Types of changes supported by KGCL, grouped into Node Changes and Edge Changes, with example KGCL command for each type, using the Uberon anatomy ontology

Change type	Description	Example command
NodeChange		
NodeRename	A node change where the name (aka rdfs:label) of the node changes	Rename UBERON:0002398 from "hand" to "manus"
NodeObsoletion	Deprecates usage of the node, but does not delete it	Obsolete "trachea" (alternatively: obsolete "UBERON:0003126")
NodeDeletion	Deletes a node from the graph	Delete node "heart"
ClassCreation	A node creation where the node is a class (as opposed to a relation)	Create "digestive system"
Synonym Replacement	A node synonym change where the text of a synonym is changed	Replace synonym "intestine" with "gut" for "alimentary canal"
NewTextDefinition	A node change where a *de novo* text definition is created	Add definition "A muscular organ that pumps blood through the body" to "heart"
Remove TextDefinition	A node change where a text definition is deleted	Remove definition for "liver"
NodeTextDefinitionChange	A node change where the text definition is changed	Change definition of "kidney" to "An organ that filters blood to produce urine"
NewSynonym	A node synonym change where a *de novo* synonym is created	Create exact synonym "thigh bone" for "femur"
RemoveSynonym	A node synonym change where a synonym is deleted	Remove synonym "arm bone" for "humerus"
EdgeChange		
EdgeCreation	An edge change in which a *de novo* edge is created	Create edge "hepatocyte" part_of "liver"
EdgeDeletion	An edge change in which an edge is removed	Delete edge "hepatocyte" part_of "lung"
NodeMove	A combination of deleting a parent edge and adding a parent edge	–
PredicateChange	An edge change where the predicate (relationship type) is modified	Change relationship between "stomach" and "digestive system" from "is_a" to "part_of"

the easiest way of bringing new features to GitHub users; they can be developed, tested, and deployed quickly and do not require additional infrastructure. The maintainers of an ontology repository can easily deploy Ontobot, after which it will monitor the GitHub repo, where it watches for issues with a specific text string: "Hey ontobot! apply:" followed by a bulleted list of ontology change requests, written in the KGCL CNL syntax. The agent will then carry out the request, generating a GitHub PR that will make the requested change(s) in the ontology source file. The PR can be quickly reviewed and merged by the maintainers of the ontology.

Figure 4 shows an example of this process in which a user requests the addition of a synonym to a term in the Mondo Disease Ontology (MONDO) and Ontobot creates a PR to make the change in the ontology file, which needs to be approved by a Mondo curator. In this example, Ontobot is invoked by a user via a GitHub issue to add a synonym to a MONDO term. Step 1: The user opens a GitHub issue in the ontology repository requesting an ontology change. The issue includes the special instruction "## Hey ontobot! apply:" followed by commands in KGCL CNL syntax describing the desired change(s) (here, adding an exact synonym to a term). Step 2: The Ontobot change agent, which watches the issue tracker for the "Hey ontobot" instruction, sees this issue and responds to it. Step 3: Ontobot creates a PR that will execute the requested change to the ontology. Curators are assigned to review the PR; it cannot be merged until at least one curator approves it. Step 4: Once the curator approves the proposed change, it is merged into the ontology and incorporated into the next release, where it is available for use by all.

User-friendly suggestion of ontology changes via BioPortal

The BioPortal ontology portal provides access to a wide selection of biomedical ontologies and tools for working with them. A new BioPortal widget, still in development, provides an easy way for biocurators to suggest changes to an ontology they work on. Under the hood, this widget uses KGCL CNL to express the desired ontology changes and invokes Ontobot to create the corresponding PR in GitHub. As we showed in the last section, biocurators can go directly to GitHub and open an issue to call Ontobot; the BioPortal widget provides an even simpler and more user-friendly interface that does not require any knowledge of KGCL.

When BioPortal users browse ontology classes, the new widget provides easy access to forms in the user interface for entering information about proposed changes (Fig. 5). Each form presented to users has the necessary fields to collect data that are specific to the various change types—e.g. for the addition of a synonym, a dropdown field allows the user to specify the type of synonym such as exact, narrow, broad, or related. In the example illustrated in Fig. 5, a BioPortal user opens the Mondo term "neurodevelopmental-craniofacial syndrome with variable renal and cardiac abnormalities" (Step 1). The user decides to request the addition of a synonym, so they click the "+" button at the right of the Synonyms row, which brings up a form that requests information about the change (Step 2). The user enters the desired new synonym (ZMYM2-related neurodevelopmental disorder with multiple anomalies) and selects the synonym type ("exact") from a pull-down menu. When the form is submitted, BioPortal creates a new GitHub issue in the Mondo repository invoking Ontobot to make the change and displays a message (Step 3) with a hyperlink to let the user go to the issue in GitHub.

When users submit change request proposals, BioPortal collects the data and generates issues that are sent to GitHub in the repository where the ontology source file is maintained. The issues have human-readable titles, and the body contents hold a machine-processable string that precisely describes the

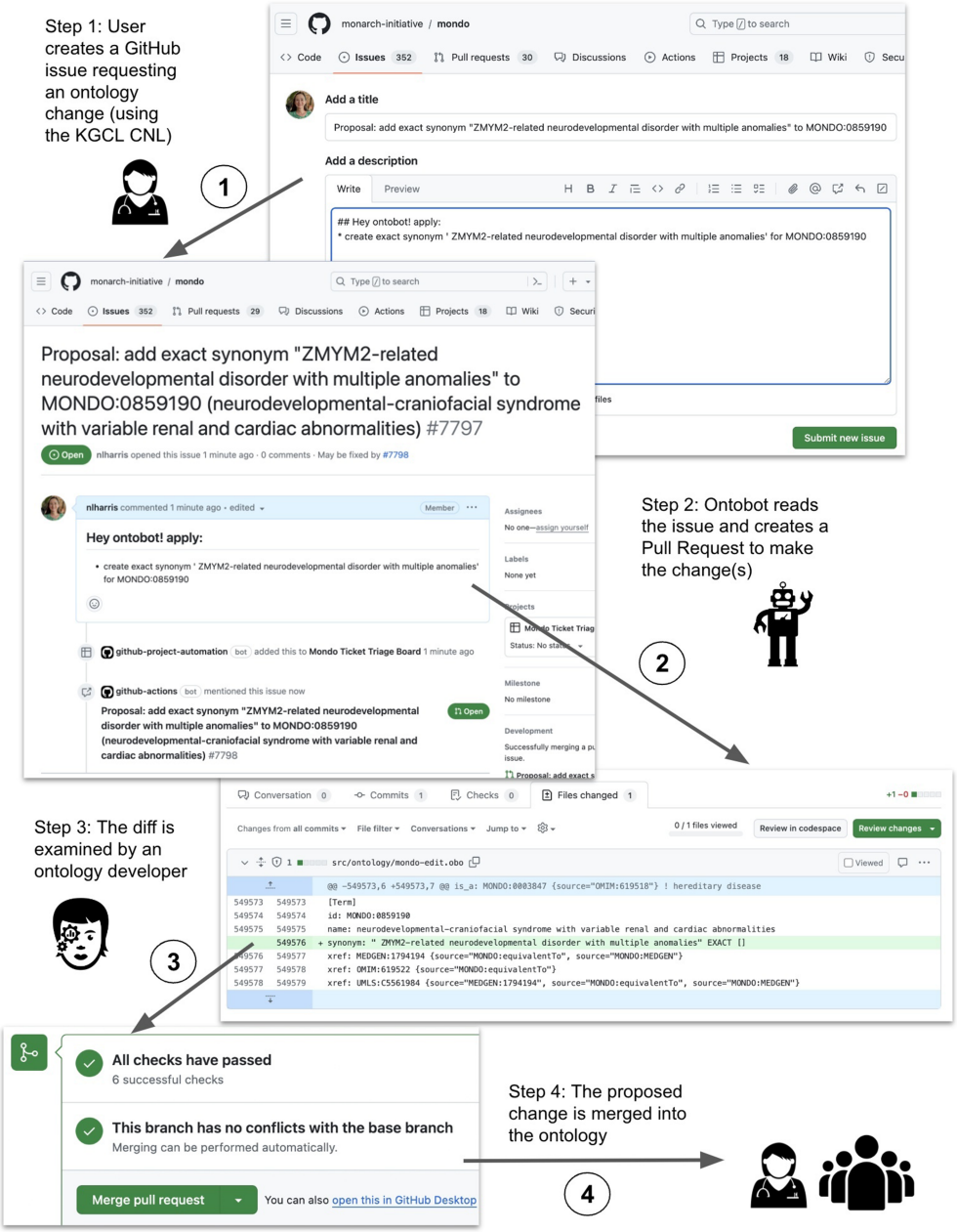

Figure 4. An example of a user-initiated change request handled by Ontobot.

requested change as a KGCL command, as described in the previous section. Figure 5 shows an example of how a user can request an ontology change, such as adding a synonym to a term, in BioPortal.

Since a good deal of text across these issues is common, we used Ruby's ERB templating system [26] to build templates for each change request type. For example, a GitHub issue title template for adding a synonym to a class appears in the BioPortal codebase as follows:

> Proposal: add synonym '**<%= synonym_label %>**' for **<%= concept_label %>**

After form submission, the template is evaluated and the appropriate fields are dynamically replaced to create the human-readable version—e.g.

> Proposal: add synonym 'cortical visual impairment' for cortical blindness

BioPortal currently supports four KGCL change request types: synonym creation, synonym removal, class obsoletion, and class renaming. We plan to continue adding support for additional change request types (Table 1). The change request functionality is also configurable on a per-ontology basis,

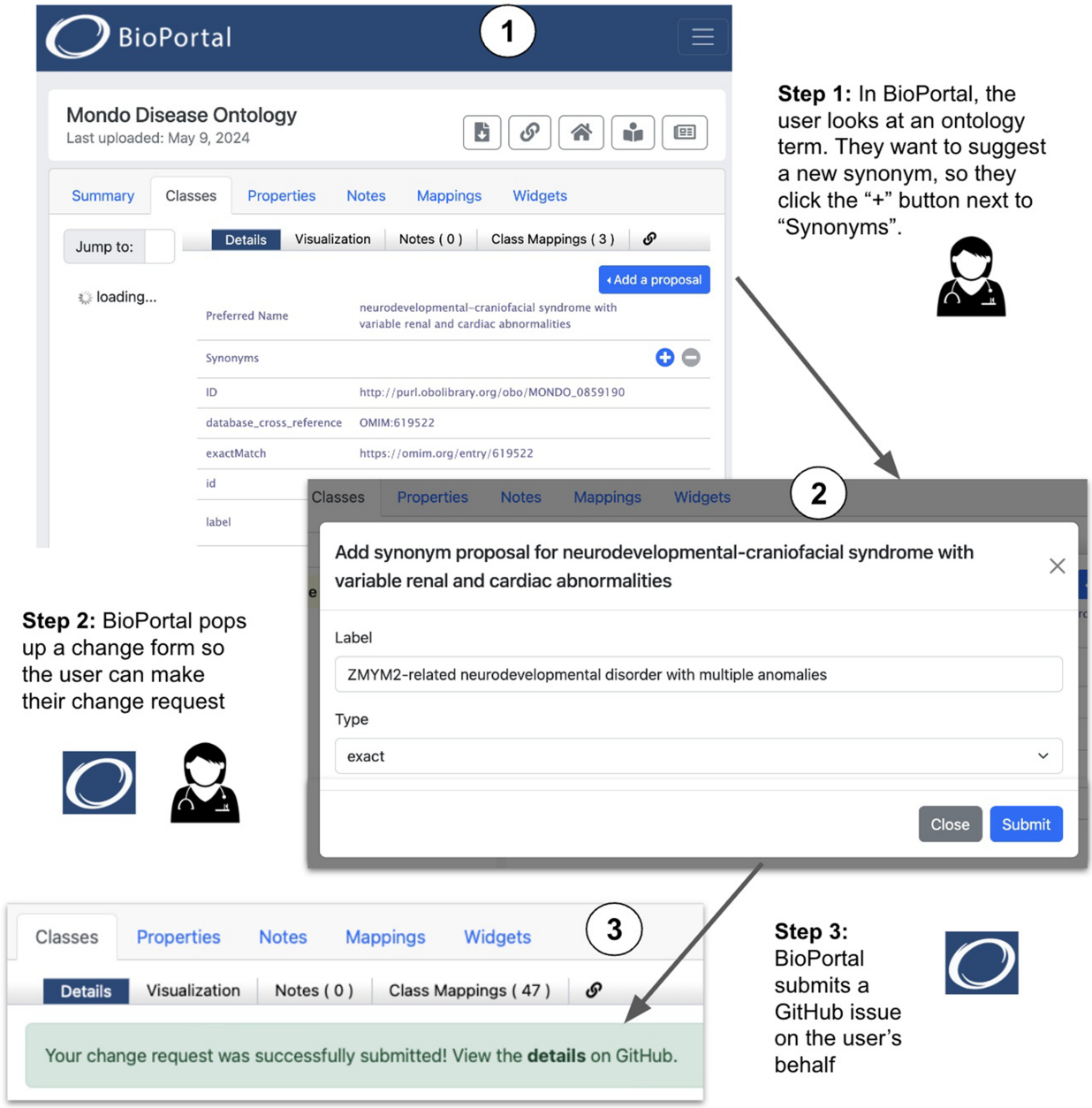

Figure 5. Requesting an ontology change in BioPortal.

since not all BioPortal users store their ontology source files in GitHub. We currently enable this functionality for four prominent ontologies in BioPortal including Mondo, GO, the Environment Ontology (ENVO) [27], and Uberon.

Developer tools for working with Knowledge Graph Change Language

Several tools that we developed or adapted to provide back-end support for Ontobot [including Ontology Access Kit (OAK) and KGCL-Java] can also be leveraged by advanced users and developers who want to add KGCL support to their applications. Currently, not all tools support all features of KGCL.

Python and Ontology Access Kit

The OAK [28] provides both a command-line interface and a Python API for two operations: (i) a "diff" operation, which takes as input two ontologies and provides a KGCL diff and (ii) an "apply" operation, which takes an ontology plus KGCL commands as input and generates a modified ontology. For both commands, the KGCL CNL and serialized data models in YAML and JSON are supported.

Java and ROBOT

In parallel with the Python implementation described earlier, we have also developed a support library for Java, KGCL-Java. The KGCL-Java library provides both a direct

translation of the KGCL data model into plain Java classes, which can be used for arbitrary manipulations of KGCL objects in a Java application, and a set of accompanying classes to facilitate working with KGCL. It includes a parser and serializer to convert KGCL objects to and from the KGCL CNL and classes to implement the changes represented by KGCL objects into OWL axioms so that the changes can be applied to an OWL ontology. Those classes are built on the OWL API library [29], which underpins several widely used ontology tools such as the Protégé ontology editor and the command-line ontology manipulation tool ROBOT (15); this opens the way for bringing KGCL support to such tools.

As a case in point, using the KGCL-Java library, we have developed a KGCL plugin for ROBOT, which adds a new "apply" command to the ROBOT toolkit. This command takes one or more KGCL changes expressed in the KGCL CNL, either directly from the command line or by reading from a file, and applies them to an ontology as part of a ROBOT pipeline, e.g.

```
robot apply -i input.owl -k "obsolete EX:1234 with
 replacement EX:5678" -o output.owl
```

Future work

Delayed applications of changes

We are considering the possibility of using KGCL to represent provisional changes: changes that are being proposed, but are not yet accepted into the target ontology or KG for some reason (e.g. because the editors need more time to assess whether they are correct). We envision a "provisional mode" for KGCL applications, where the changes that are being described using KGCL syntax are not directly applied to the target ontology, but instead "stored" as KGCL objects within the ontology until they are either approved (in which case they will then be effectively applied) or rejected. The main benefits of such an approach—compared to, e.g. keeping provisional changes as PRs waiting to be merged in a repository—is that the changes would be directly visible in the ontology itself and could be queried and manipulated like any other ontological entity using standard ontology tools and libraries.

How exactly KGCL objects should be stored in an ontology is still under consideration. Our current approach is to represent them as annotations on the very ontological entities that they are intended to modify.

A draft implementation of the "provisional mode" is available in the KGCL-Java library and ROBOT plugin, where a command like robot apply -k "obsolete EX:1234"– provisional will store a NodeObsoletion KGCL object into the ontology (instead of actually obsoleting the EX:1234 class), and conversely a command like robot apply –pending all will effectively apply all the provisional changes currently stored in the ontology.

Viewing diffs in BioPortal

KGCL makes it possible to decouple the representation of changes from the process of computing them. It provides a higher-level way to communicate changes that correspond to how ontologists and curators think of those changes, abstracting away from low-level RDF or OWL diffs. In the future, we plan to use KGCL to support BioPortal's change reporting to show the differences between two versions of an ontology.

Extension of the core model to support multiple flavors of KGs and ontologies

While our primary use case is changes in ontologies, we have aimed to keep the data model generic enough to use with generalized KGs. We aim to deploy KGCL within our Knowledge Graph Hub framework [30], showing at a high level how KGs change over time. For example, the KGCL "edge change" operations are intended to be used with any KG and are not limited to particular OWL axiom types such as subclass axioms. They are also intended to support property graph style, KGs, where individual edges can be annotated with additional contextual information, such as the kind described in the Biolink model [31]. However, some users have requested full support for more complex axiom types, including logical definition style equivalence axioms commonly found in OBO ontologies, so we plan to add these in the future.

Artificial Intelligence applications and evaluations

KGCL is fundamentally a tool to help humans communicate about changes in ontologies, either to communicate what changes have been made or to request desired changes. Generative Artificial Intelligence (AI) and Large Language Models (LLMs) can complement the use of KGCL for both these tasks. Our previous work has demonstrated that LLMs can be used under the supervision of expert users to assist with the generation of new terms [32]. However, as our community survey showed, many of the bottlenecks in ontology development are around making other kinds of changes to ontologies. To enable the use of LLMs for ontology changes, we developed a prototype ChatGPT plugin that can be used to generate KGCL CNL from free text descriptions of changes [33]. This plugin has the limitation that it has to be used within the ChatGPT UI, limiting broader uptake and making it hard to evaluate. To remedy this, we have started constructing an AI change evaluation set by mining GitHub issues and associated PRs across ontology GitHub repos. For each PR that can be associated with a change, we generate a KGCL CNL description of the changes, together with the issue history associated with that PR. Our intent is to use this in Retrieval Augmented Generation applications and incorporate this into the Ontobot change agent. Our goal is to allow Ontobot to read any issue in a GitHub repo (whether in CNL syntax or plain natural language) and generate a PR, using the previous history of changes, and information in the ontology as context. The evaluation set and an associated LangChain [34] agent are available from our GitHub repo [35].

Aligning with related work

Our work is influenced by existing OWL-level diffing tools such as ROBOT and Bubastis. The data model we have devised has similarities to the model used in the COnto-Diff framework, as well as work summarized in Groß *et al.* [36], the DIACHRON framework [37], and the more recent DynDiff framework [38]. The emphasis of our efforts has been on the creation of a human-readable CNL that can serve as a means of communication between humans and machines. We have commenced efforts to align and map these data models and provide bridges between these tools. Simple changes have largely been mapped to their counterparts in

COnto-Diff, DIACHRON, and DynDiff. For instance, NodeMove is mapped to move(c, C_To, C_From), Move_Class(a, B1, B2)/Move_Property(a, B1, B2), and moveC(c, B1, B2)/moveP(p, B1, B2) accordingly. Specific changes within the KGCL framework, such as NodeMappingChange and PredicateChange, do not have direct equivalents in these models. This is because they focus more on node-related changes (i.e. changes pertaining to classes, properties, and instances) rather than on edge changes or mappings. Instance-level (ABox) changes, like instance addition or deletion, and heuristic changes, such as concept merge and split, are yet to be considered, although as mentioned earlier, the model will be extended to accommodate these types of changes based on community-driven use cases.

Conclusions

Ontologies and KGs are highly dynamic in nature, undergoing frequent changes in the light of new knowledge or improved curation. However, change is rarely treated as a first-class object. By providing a standardized representation of changes in ontologies and KGs, KGCL and its associated tooling provide a mechanism to help communicate to curators and users the changes that have occurred in ontologies over time and a mechanism to communicate and enact desired changes in an ontology.

Acknowledgements

The authors thank all the participants of the inaugural change language workshop: Allen Baron, Jim Balhoff, Sierra Moxon, Bill Duncan, Sabrina Toro, Matthew Horridge, Nicole Vasilevsky, Naoual Smaili, Emily Hartley, Sue Bello, Yvonne Bradford, Ian Braun, Timothy Redmond, Chris Roeder, Leigh Carmody, Clement Jonquet, Claus Weiland, Jonas Grieb, Thomas Liener, Aleix Puig, Philip Strömert, Charles Tapley Hoyt, Paul Fabry, Rhiannon Cameron, Damion Dooley, and Daniel Olson.

They thank the BioPortal Scientific Advisory Board for their helpful feedback on our project: Yolanda Gil, Clement Jonquet, Andrew Phillips, and Andrew Su.

Author contributions

C.J.M. devised the overall framework and data model and drafted the manuscript. C.K. and J.A.O. devised and wrote the grammar and Python RDF implementation. H.H. wrote the GitHub adapter and integrated it into OAK. N.M. contributed to the overall framework and data model and helped draft the manuscript. N.L.H. contributed substantially to the manuscript. N.K. contributed to the data model and mappings. J.V. wrote the BioPortal layer. M.A.M. and J.B.G. contributed to the overall framework and schema. D.G.-G. wrote the Java library and ROBOT adapter. All authors made contributions to the manuscript.

Conflict of interest: None declared.

Funding

This work was supported by the National Institutes of Health [U24 GM143402, 5U01HG009453-03 (past support)]; the Director, Office of Science, Office of Basic Energy Sciences, of the US Department of Energy (Contract No. DE-AC0205CH11231 to H.H., J.H.C., N.L.H., and C.J.M.); and the National Science Foundation (BBSRC-NSF/BIO BB/T014008/1 to D.G.G.). In addition, a gift from Bosch Corporation helped support this work.

Data Availability

All software and schemas discussed in this paper are open access, and are available at https://github.com/INCATools/kgcl.

References

1. Diehl AD, Meehan TF, Bradford YM *et al*. The Cell Ontology 2016: enhanced content, modularization, and ontology interoperability. *J Biomed Semant* 2016;**7**:44. https://doi.org/10.1186/s13326-016-0088-7
2. Mungall CJ, Torniai C, Gkoutos GV *et al*. Uberon, an integrative multi-species anatomy ontology. *Genome Biol* 2012;**13**:R5. https://doi.org/10.1186/gb-2012-13-1-r5
3. Jain S, Pei L, Spraggins JM *et al*. Advances and prospects for the Human BioMolecular Atlas Program (HuBMAP). *Nat Cell Biol* 2023;**25**:1089–100. https://doi.org/10.1038/s41556-023-01194-w
4. Regev A, Teichmann SA, Lander ES *et al*. The Human Cell Atlas. *Elife* 2017;**6**:e27041. https://doi.org/10.7554/eLife.27041
5. Gargano MA, Matentzoglu N, Coleman B *et al*. The Human Phenotype Ontology in 2024: phenotypes around the world. *Nucleic Acids Res* 2024;**52**:D1333–46. https://doi.org/10.1093/nar/gkad1005
6. Noy NF, Shah NH, Whetzel PL *et al*. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Res* 2009;**37**:W170–3. https://doi.org/10.1093/nar/gkp440
7. Ong E, Xiang Z, Zhao B *et al*. Ontobee: a linked ontology data server to support ontology term dereferencing, linkage, query and integration. *Nucleic Acids Res* 2017;**45**:D347–52. https://doi.org/10.1093/nar/gkw918
8. Jupp S, Burdett T, Leroy C *et al*. A new ontology lookup service at EMBL-EBI. *SWAT4LS* 2015;**2**:118–9.
9. Himmelstein DS, Baranzini SE, Tang H. Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes. *PLoS Comput Biol* 2015;**11**:e1004259. https://doi.org/10.1371/journal.pcbi.1004259
10. Callahan TJ, Tripodi IJ, Stefanski AL *et al*. An open source knowledge graph ecosystem for the life sciences. *Sci Data* 2024;**11**:363. https://doi.org/10.1038/s41597-024-03171-w
11. Reese JT, Unni D, Callahan TJ *et al*. KG-COVID-19: a framework to produce customized knowledge graphs for COVID-19 response. *Patterns* 2021;**2**:100155. https://doi.org/10.1016/j.patter.2020.100155
12. Knublauch H, Horridge M, Musen MA *et al*. *The Protege OWL Experience*. OWLED, 2005. https://www.researchgate.net/profile/Mark-Musen/publication/221218459_The_Protege_OWL_Experience/links/09e415113c8d91ab91000000/The-Protege-OWL-Experience.pdf (1 September 2024, date last accessed).
13. Musen MA, Protégé Team. The Protégé project: a look back and a look forward. *AI Matters* 2015;**1**:4–12. https://doi.org/10.1145/2757001.2757003

14. Tudorache T, Nyulas C, Noy NF *et al*. WebProtégé: a collaborative ontology editor and knowledge acquisition tool for the web. *Semant Web* 2013;**4**:89–99. https://doi.org/10.3233/SW-2012-0057
15. Jackson R, Matentzoglu N, Overton JA *et al*. OBO Foundry in 2021: operationalizing open data principles to evaluate ontologies. *Database* 2021;**2021**:baab069. https://doi.org/10.1093/database/baab069
16. Harris N, Matentzoglu N, Mungall C. Data and slides from May 2023 virtual workshop on change languages in ontologies and knowledge graphs.https://doi.org/10.5281/ZENODO.14188134
17. Jackson RC, Balhoff JP, Douglass E *et al*. ROBOT: a tool for automating ontology workflows. *BMC Bioinf* 2019;**20**:407. https://doi.org/10.1186/s12859-019-3002-3
18. Malone J, Stevens R. Measuring the level of activity in community built bio-ontologies. *J Biomed Inform* 2013;**46**:5–14. https://doi.org/10.1016/j.jbi.2012.04.002
19. Binns D, Dimmer E, Huntley R *et al*. QuickGO: a web-based tool for Gene Ontology searching. *Bioinformatics* 2009;**25**:3045–6. https://doi.org/10.1093/bioinformatics/btp536
20. Jacobson M, Sedeño-Cortés AE, Pavlidis P. Monitoring changes in the Gene Ontology and their impact on genomic data analysis. *Gigascience* 2018;**7**:giy103. https://doi.org/10.1093/gigascience/giy103
21. Hartung M, Groß A, Rahm E. COnto-Diff: generation of complex evolution mappings for life science ontologies. *J Biomed Inform* 2013;**46**:15–32. https://doi.org/10.1016/j.jbi.2012.04.009
22. Moxon S, Solbrig H, Unni D *et al*. The linked data modeling language (LinkML): a general-purpose data modeling framework grounded in machine-readable semantics. In: *2021 International Conference on Biomedical Ontologies, ICBO 2021*. September 15–18, 2021. pp. 148–51. Bolzano, Italy:CEUR-WS, 2021.
23. INCATools/kgcl. *src/kgcl_schema/grammar/kgcl.lark at main*. Github. https://github.com/INCATools/kgcl/blob/main/src/kgcl_schema/grammar/kgcl.lark (1 September 2024, date last accessed).
24. Hoyt CT, Gyori BM. The O3 guidelines: open data, open code, and open infrastructure for sustainable curated scientific resources. *Sci Data* 2024;**11**:547. https://doi.org/10.1038/s41597-024-03406-w
25. Decan A, Mens T, Mazrae PR *et al*. On the use of github actions in software development repositories. In: *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. pp. 235–45. Limassol, Cyprus: IEEE, 2022.
26. Ruby/erb. *An Easy to Use but Powerful Templating System for Ruby*. Github. https://github.com/ruby/erb (1 September 2024, date last accessed).
27. Buttigieg PL, Pafilis E, Lewis SE *et al*. The environment ontology in 2016: bridging domains with increased scope, semantic density, and interoperation. *J Biomed Semant* 2016;**7**:57. https://doi.org/10.1186/s13326-016-0097-6
28. INCATools/ontology-access-kit. *Ontology Access Kit: Ontology Access Kit: A Python Library and Command Line Application for Working with Ontologies*. Github. https://github.com/INCATools/ontology-access-kit (1 September 2024, date last accessed).
29. Horridge M, Bechhofer S, and Noppens O. *Igniting the OWL 1.1 Touch Paper: The OWL API*. https://ceur-ws.org/Vol-258/paper19.pdf (1 September 2024, date last accessed).
30. Caufield JH, Putman T, Schaper K *et al*. KG-Hub-building and exchanging biological knowledge graphs. *Bioinformatics* 2023;**39**:btad418. https://doi.org/10.1093/bioinformatics/btad418
31. Unni DR, Moxon SAT, Bada M *et al*. Biolink Model: a universal schema for knowledge graphs in clinical, biomedical, and translational science. *Clin Transl Sci* 2022;**15**:1848–55. https://doi.org/10.1111/cts.13302
32. Toro S, Anagnostopoulos AV, Bello S *et al*. Dynamic retrieval augmented generation of ontologies using artificial intelligence (DRAGON-AI). *J Biomed Semantics* 2024;**15**:19. 10.1186/s13326-024-00320-3
33. Creators Mungall C. AI guided ontology curation workflows and the ROBOT template GPT helper. https://doi.org/10.5281/zenodo.10901704
34. Topsakal O, Akinci TC. Creating large language model applications utilizing LangChain: a primer on developing LLM apps fast. *ICAENS* 2023;**1**:1050–6. https://doi.org/10.59287/icaens.1127
35. Creators Hegde H. LLM change agent. https://doi.org/10.5281/zenodo.13712477
36. Groß A, Pruski C, Rahm E. Evolution of biomedical ontologies and mappings: overview of recent approaches. *Comput Struct Biotechnol J* 2016;**14**:333–40. https://doi.org/10.1016/j.csbj.2016.08.002
37. Papavasileiou V, Flouris G, Fundulaki I *et al*. High-level change detection in RDF(S) KBs. *ACM Trans Database Syst* 2013;**38**:1–42. https://doi.org/10.1145/2445583.2445584
38. Diaz Benavides S, Cardoso SD, Da Silveira M *et al*. Analysis and implementation of the DynDiff tool when comparing versions of ontology. *J Biomed Semant* 2023;**14**:15. https://doi.org/10.1186/s13326-023-00295-7

Database, 2025, 00, baae133, DOI: https://doi.org/10.1093/database/baae133, Original article