OxO2 - A SSSOM Mapping Browser for Logically Sound Crosswalks

Henriette Harmse[1,*], Haider Iqbal[1], Helen Parkinson[1] and James McLaughlin[1]

[1]*Samples, Phenotypes and Ontologies Team (SPOT), EMBL-EBI, Wellcome Genome Campus, Hinxton, CB10 1SD, Cambridgeshire, United Kingdom*

Abstract

EMBL-EBI created OxO to enable users to map between datasets that are annotated with different ontologies. Mappings identified by the first version of OxO were not necessarily logically sound, lacked important provenance information such as author and reviewer, and could timeout or crash for certain requests. In this paper we introduce OxO2 to address these concerns. Provenance is addressed by implementing SSSOM, a mapping standard that defines provenance for mappings. SSSOM defines the conditions under which logical sound mappings can be derived and is implemented in OxO2 using Nemo, a Datalog rule engine. To ensure reasoning is performant and memory efficient, Nemo implements a number of strategies that ensures OxO2 will be stable for all requests. Due to these changes, OxO2 users will be able to integrate between disparate datasets with greater confidence.

Keywords

OxO, mapping, crosswalk, ontologies, FAIR Principles

1. Introduction

The FAIR principles are a set of guidelines designed to make datasets and their metadata Findable, Accessible, Interoperable and Reusable for both machines and humans [1], and require the use of FAIR vocabularies and/or ontologies [1, 2]. Since their publication, the FAIR principles have been widely adopted to support FAIR data implementations [3].

Specific domains may aim to standardize on the ontologies they use, but often multiple ontologies are employed within a given domain [4]. At EMBL-EBI, the Experimental Factor Ontology (EFO) [5] is used to annotate EMBL-EBI datasets. However, many other ontologies are also used within EMBL-EBI, such as the Mondo Disease Ontology (MONDO) [6], the Human Disease Ontology (DOID) [7] and the Human Phenotype Ontology (HP) [8].

When considering cross-domain integration, differences in frames of reference mean that different domains often standardize on different ontologies [9]. As an example, phenotypes (observable traits) are critical for diagnosing and treating diseases, and are studied in both model organisms (organisms that can easily be studied in a laboratory) such as zebrafish and mammals, as well as in humans. As such, zebrafish data are annotated with the Zebrafish Phenotype Ontology (ZP), mammalian data with the Mammalian Phenotype Ontology (MP) and human data with the Human Phenotype Ontology (HP), respectively. To study a specific phenotype, say increased heart size, researchers benefit from being able to combine data annotated with ZP:0000532 (increased size of heart), MP:0000274 (enlarged heart) and HP:0001640 (Cardiomegaly) [10].

In order to facilitate these kinds of integration across datasets annotated with different ontologies, there is the need to map the concepts from one ontology to the related concepts of another ontology. EMBL-EBI created the Ontology Xref (Cross-reference) Service - referred to as OxO - to address this need [4]. In this paper we will refer to this version of OxO as OxO1.

OxO1 identified mappings between concepts of different ontologies by crawling the Ontology Lookup Service (OLS) [11] and reading the **xref** (cross-reference relationships) annotations for each of the

Proceedings of FOIS 2025 Satellite events co-located with the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), September 10-12, 2025, Catania, Italy

*Corresponding author.

henriette007@ebi.ac.uk (H. Harmse)

0000-0001-7251-9504 (H. Harmse); 0000-0003-3035-4195 (H. Parkinson); 0000-0002-8361-2795 (J. McLaughlin)

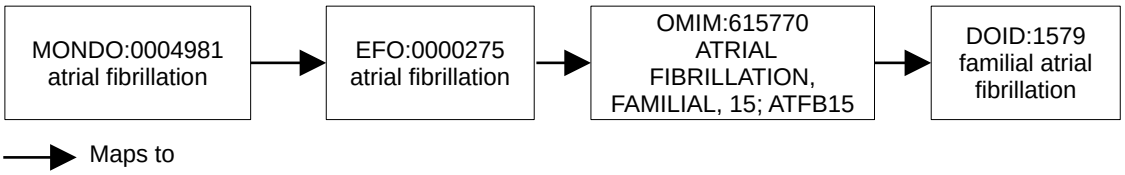

Figure 1: A crosswalk from `MONDO:0004981` to `DOID:1579` by mapping from `MONDO:0004981` to `EFO:0000275` to `OMIM:615770` to `DOID:1579`.

ontology concepts. The notion of an xref is intentionally loosely defined as part of the Open Biological and Biomedical Ontology (OBO) Flat File Format Specification, which has historically been used to define ontologies in the bioinformatics community [12, 13, 14]. An xref mapping may be used to indicate an exact match, but it could as well refer to a broader or narrower match. In fact xrefs may have legio meanings.

A key feature of OxO1 is that it enables walking across ontologies – from a concept in one ontology to a concept in the next ontology – referred to as a **crosswalk**. This is illustrated in Figure 1, where the crosswalk between `MONDO:0004981` and `DOID:1579` is realized by mapping from `MONDO:0004981` to `EFO:0000275`, to `OMIM:615770`, and finally to `DOID:1579`. This is referred to as a mapping at a **distance** of 3, since mappings between ontologies were performed 3 times. Distances greater than 3 are not supported in OxO1 [4]. Since xrefs are not well defined, the meaning of these kinds of crosswalks across multiple ontologies are also weakly defined.

Despite the loose definition of mappings in OxO1, it supports the following user groups.

Researchers, such as biologists seeking new treatments for heart disease, who may want to study the phenotype "enlarged heart".

Resource providers or data analysts integrating datasets annotated with different ontologies, such as ZP and MP.

At EMBL-EBI, OxO1 is used by resources such as European Variation Archive (EVA) [15], the Genome-Wide Association Study (GWAS) Catalog [16] and Europe PMC [17].

The Simple Standard for Sharing Ontological Mappings (SSSOM) provides precise definitions for mappings, along with related provenance information, and clearly defines the conditions under which new mappings can be derived from existing mappings [18]. In this paper we introduce OxO2, a new ontology mapping service that implements SSSOM and enables browsing of mappings between ontologies with logically sound crosswalks.

In the next section (Section 2), we give a brief overview of SSSOM and explain how it enables logically sound crosswalks. Section 3 describes the value proposition of OxO2. In Section 4, we motivate the design decisions of the OxO2 implementation, and in Section 5, we review related SSSOM implementations. Finally, in Section 6, we discuss the current limitations of our OxO2 implementation and future developments under consideration.

2. Background

In this section, we explain how OxO2 uses Datalog (Section 2.3) to derive logically sound and complete mappings based on SSSOM chain rules (Section 2.2), in a performant and memory efficient manner (Section 2.4).

2.1. An Overview of SSSOM

SSSOM [18, 19, 20, 21] is briefly summarised here, along with an example to illustrate the value proposition of OxO2.

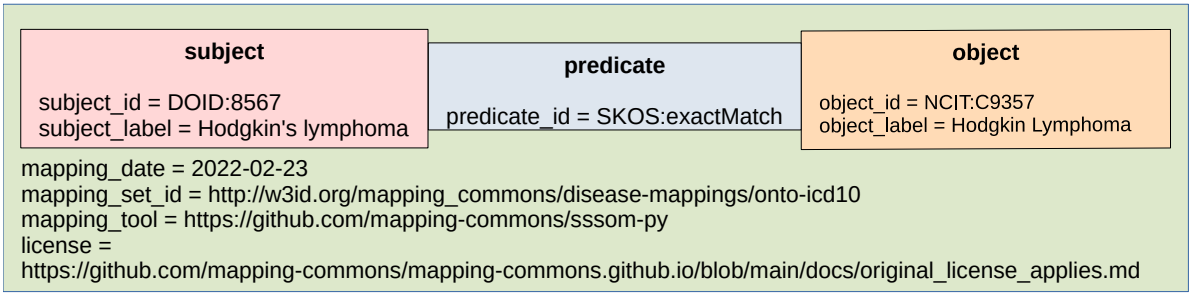

Figure 2: A SSSOM mapping that maps `DOID:8567` via `SKOS:exactMatch` to `NCIT:C9357`. The pink, blue and orange boxes represent subject-, predicate- and object metadata. The green box represents the mapping set metadata. The pink, blue and orange boxes are inside the green box to indicate that mappings belong to a mapping set.

Table 1
An abbreviated list of SSSOM standard metadata elements with example data, partially based on [23]. Author and reviewer information are fictional.

Name	Description	Example
author_id	Identifies the persons or groups responsible for asserting the mappings	Henriette Harmse
confidence	A score between 0 and 1 denotes the confidence of correctness of a mapping.	0.7
mapping_set_id	A globally unique identifier for the mapping set	http://w3id.org/mapping_commons/disease-mappings/onto-icd10
mapping_tool	A reference to the tool or algorithm that was used to generate the mapping	https://github.com/mapping-commons/sssom-py
object_id	The ID of the object	`NCIT:C9357`
object_label	The label of the object	Hodgkin Lymphoma
predicate_id	The ID that relates subject and object	`SKOS:exactMatch`
reviewer_id	Identifies the persons or groups that reviewed and confirmed the mapping	Helen Parkinson
subject_id	The ID of the subject of the mapping	`DOID:8567`
subject_label	The label of subject of the mapping	Hodgkin's lymphoma

Mappings are defined as a subject mapping via a predicate to an object. In SSSOM, this is expressed as a `subject_id` mapping via a `predicate_id` to an `object_id`, where the `subject_id`, `predicate_id` and `object_id` are each expressed as abbreviated Uniform Resource Identifiers (URIs), called CURIEs [22]. In general all identifiers in SSSOM are CURIEs, where CURIEs can refer to entities including ontology concepts, database references etc. To define the provenance of a mapping, a mapping is described using SSSOM standard metadata elements, such as `subject_label`, `author_id` (see Table 1).

For ease of use, SSSOM defines a tab-separated values (TSV) table format for representing mappings. Table 2 defines three mappings, firstly that `DOID:8567` maps via `SKOS:exactMatch` to `NCIT:C9357`, secondly that `HP:0012189` maps via `OWL:equivalentClass` to `DOID:8567`, and lastly that `MONDO:0009348` maps via `SKOS:closeMatch` to `HP:0012189`. To define the metadata of a set

Table 2
An example SSSOM mapping file.

subject_id	subject_label	predicate_id	object_id	object_label
DOID:8567	Hodgkin's lymphoma	SKOS:exactMatch	NCIT:C9357	Hodgkin Lymphoma
HP:0012189	Hodgkin lymphoma	OWL:equivalentClass	DOID:8567	Hodgkin's lymphoma
MONDO:0009348	classic Hodgkin lymphoma	SKOS:closeMatch	HP:0012189	Hodgkin lymphoma

of mappings, a mappings TSV file can be prepended with a header in YAML (a human-friendly data serialization) format [24]. The corresponding header for Table 1 is provided in Listing 1.

Listing 1: YAML Header that defines the mapping set for the mappings in Table 2

```
# mapping_set_id: https://fois2025.com/example.mappings.tsv
# curie_map:
#   DOID: http://purl.obolibrary.org/obo/DOID_
#   HP: http://purl.obolibrary.org/obo/HP_
#   MONDO: http://purl.obolibrary.org/obo/MONDO_
#   NCIT: http://purl.obolibrary.org/obo/NCIT_
#   OWL: http://www.w3.org/2002/07/owl#
#   SKOS: http://www.w3.org/2004/02/skos/core#
```

2.2. Chain Rules

With SSSOM it is possible to derive mappings from existing ones in a way that ensures derived mappings are logically sound. This is achieved by providing chain rules that define the circumstances under which mappings can be composed from existing mappings. A **rule** defines the conditions under which an action will occur and follows an if-then structure. **Rule chaining** (or rule chains) refers to the mechanism where the execution of one rule triggers the execution of subsequent rules. Execution stops when no more rules are applicable [25]. SSSOM chain rules are categorized as transitivity rules, rule chains over exact and equivalent matches, inverse rules and generalization rules. The approach that SSSOM takes is to provide reasonable defaults for mapping tools. Hence, SSSOM defines transitive rules for `skos:narrowMatch` and `skos:broadMatch` [26], despite them not being defined as transitive in the Simple Knowledge Organization System (SKOS) specification [27]. In total, the SSSOM chain rules define 22 rules that can be applied to SSSOM mappings, all of which are implemented and used in OxO2.

For illustration purposes, we will focus only on chain rules over exact and equivalent matches, labeled as RCE2 in the SSSOM specification [26]. RCE2 has two variants, which apply to mappings where the `predicate_id` is either `SKOS:exactMatch` or `OWL:equivalentClass`. These two variants are defined in Listings 2 and 3.

Listing 2: RCE2-1

$$A - [p] - B\,,$$
$$B - [\texttt{OWL:equivalentClass}] - C$$
$$\rightarrow A - [p] - C$$

Listing 3: RCE2-2

$$A - [p] - B\,,$$
$$B - [\texttt{SKOS:exactMatch}] - C$$
$$\rightarrow A - [p] - C$$

Listing 2 states that when we have CURIEs A, B and C, such that A is mapped to B via an arbitrary `predicate_id` p, and B is mapped to C via the `predicate_id` `OWL:equivalentClass`, then we

can derive that A maps to C via the same arbitrary `predicate_id` p. The meaning of Listing 3 follows in a similar fashion.

Based on the mappings in Table 2, we can derive the two new mappings shown in Listings 4 and 5, by applying the chain rules of Listing 2 and 3, respectively. In Listing 4 we derive that `MONDO:0009348` (classic Hodgkin lymphoma) has a `SKOS:closeMatch` with `DOID:8567` (Hodgkin's lymphoma), by applying the chain rule of Listing 2. By applying the chain rule of Listing 3, Listing 5 derives that `MONDO:0009348` has a `SKOS:closeMatch` with `NCIT:C9357` (Hodgkin Lymphoma), by using both an existing mapping from Table 2 and the derived mapping from Listing 4.

Listing 4: Apply RCE2-1

`MONDO:0009348` $-$ [`SKOS:closeMatch`] $-$ `HP:0012189`,
`HP:0012189` $-$ [`OWL:equivalentClass`] $-$ `DOID:8567`
$\rightarrow$ `MONDO:0009348` $-$ [`SKOS:closeMatch`] $-$ `DOID:8567`

Listing 5: Apply RCE2-2

`MONDO:0009348` $-$ [`SKOS:closeMatch`] $-$ `DOID:8567`,
`DOID:8567` $-$ [`SKOS:exactMatch`] $-$ `NCIT:C9357`
$\rightarrow$ `MONDO:0009348` $-$ [`SKOS:closeMatch`] $-$ `NCIT:C9357`

2.3. Datalog

In this section we show that SSSOM chain rules can be expressed as Datalog rules, and hence, all characteristics of Datalog rules also apply to SSSOM chain rules. OxO2 uses Datalog to derive logically sound and complete mappings in a performant and memory efficient way.

Datalog is a declarative logic programming language designed for logical inference. A **Datalog rule** defines the premises from which a conclusion follows and has an if-then structure similar to chain rules. A **Datalog program** consists of Datalog rules. Datalog rules are applied recursively which means that the result of a rule can be used by other rules to derive new information [25]. A Datalog rules engine is an application that, given a set of facts and a set of rules, it can infer new facts in a way that is logically sound. Since Datalog is declarative, it means that one can state the rules that are applicable, without having to state how the inferences are to be determined. The Datalog rules engine is responsible for implementing the method(s) of determining inferences [28].

To ensure that all facts that can be derived from a Datalog program are finite, 2 conditions must hold: firstly, all facts must be constants, and secondly, all variables in the conclusion of a rule, must appear in the body of a rule. We note that these conditions hold for SSSOM chain rules [29]:

1. All SSSOM mappings are without variables, as seen from the first mapping in Table 2, which states that `DOID:8567` maps via `SKOS:exactMatch` to `NCIT:C9357` where `DOID:8567`, `SKOS:exactMatch` and `NCIT:C9357` are all constants.
2. The second condition holds for the chain rule defined in Listing 2. The conclusion, given after the $\rightarrow$, is $A - [p] - C$ where A, p and C are variables. These variables appear in the premises $A - [p] - B$ and $B - [$`OWL:equivalentClass`$] - C$ where A and p appear in the first premise and C in the second premise. Doing a similar check, shows that condition 2 holds for Listing 3 as well. Note that this condition only applies to variables and not constants. Hence, any constant, such as `SKOS:exactMatch` can appear in the conclusion without appearing in any premises. We note that for all SSSOM chain rules [26] this condition holds.

Datalog has some characteristics [29] that are important for OxO2.

Termination For a finite set of facts, Datalog inferencing always terminates. Hence, the OxO2 dataloader can assume inferencing is complete when the inferencing algorithm terminates.

Soundness All Datalog inferences are guaranteed to follow logically from the assumed facts. For OxO2 this means inferred mappings can be trusted to be logically correct.

Completeness means that all inferences that can be derived from the facts, are indeed derived. Hence, OxO2 is guaranteed to include all mappings that follow logically from asserted mappings.

Favourable time and space computational complexity The output of the Datalog inference algorithm, and the time it takes to run till it terminates, is polynomial in the size of the input. This means the time and space needs of Datalog inferencing is limited by a polynomial, such as n^k, where n represents the input size and k is a constant. This is a positive result when compared to time and space needs that can be exponential, e.g., k^n. To illustrate the difference between polynomial time and exponential time, assume the input size is $n = 10$ and $k = 2$, then the polynomial time is $n^k = 10^2 = 100$, while exponential time is $k^n = 2^{10} = 1024$. If we now change the input size to $n = 100$, we see that the polynomial time is still reasonable, whereas the exponential time is not [1]. This result indicates that theoretically the Datalog inference algorithm is feasible.

2.4. Strategies for optimizing Datalog Reasoning

Datalog reasoning has some challenges when applied to large knowledge graphs and hence a number of strategies are employed to deal with these challenges. We mention the strategies here that are used in the implementation of OxO2.

The most basic approach to Datalog inferencing is the Naive Evaluation algorithm. It starts with the original facts, and in the first step it derives new facts by applying all the rules to the original facts. Subsequent steps apply the rules to both original facts and derived facts, repeatedly, until no new facts can be added. This can result in the same inferences being made repeatedly. To avoid this redundancy, an optimization is to apply subsequent rules only to facts derived in the previous iteration, which is called Seminaive Evaluation [25].

A related concern is the explanation of inferences, which is known to be exponential in the number of facts. This is because there can be multiple explanations for a single inference. To avoid this explosion of the number of explanations, a technique called tracing is used. Instead of recording all possible explanations, tracing aims to provide only 1 explanation for a given inference. This approach ensures that the explanation of all inferences is feasible [31].

To ensure fast and compact in-memory data storage, a column-based storage layout is used which enables efficient compression schemes. However, column-based storage increases the cost of updates (e.g. when new inferences are made). For this reason each rule application is written to a separate delta table [32]. To reduce the cost of combining delta tables, combined deltas are cached [33].

Even with the above optimizations, users can wait a long time when inferences are done in realtime. Therefore, it is recommended to precompute inferences to improve the user experience [32].

3. Results

3.1. OxO1 Challenges

The key value proposition of OxO1 is that it is able to identify potentially related mappings that could either map directly or indirectly up to a distance of 3 to other CURIEs. OxO1 has 3 main challenges.

1. The semantics of the mappings it returns are vague, which is exacerbated for mappings at a distance greater than 1. In OxO1 it is possible to derive mappings that have the form of Listing 6:

[1] There are many other computational complexity classes besides polynomial and exponential. See [30] for details.

Listing 6: Problematic inferred mapping of OxO1

$A - [\texttt{SKOS:broadMatch}] - B$,
$B - [\texttt{SKOS:narrowMatch}] - C$
$\rightarrow A - [\textbf{meaning?}] - C$

2. Returning mappings at a distance of 3 is only feasible when requesting mappings for a small set of CURIEs. This problem became more accentuated as the number of mappings and ontologies in OxO1 grew, with queries either timing out, or the OxO1 server crashing. To find mappings for a CURIE at distance 1, OxO1 queries all mappings in its Neo4J database. As these mappings are stored in the database, returning these results are highly efficient. For retrieving mappings at a distance of 2 or 3, OxO1 determines derived mapping at query time using the following algorithm. It first determines the cross product of all terms in the database. This result is used to define possible starting and ending nodes, for which all possible paths of length 2 or 3 are determined, depending on the distance specified. The computational complexity of the cross product is $n \times n$, where n is the number of terms. The computational complexity for determining all paths depends on the length of the path (or distance) d, and the average degree of the nodes of the graph under consideration, which is denoted by k. The degree of a node is determined by the number of inbound and outbound edges connected to it. The computational complexity for determining all paths is given by k^d. Hence, the total computational complexity for the OxO1 algorithm is $k^d \times n^2$. At the time of writing these values for OxO1 are as follows:
   - the number of terms $n = 698\,651$,
   - the average degree of nodes $k = 3$, and
   - the length of the paths to determine, which is also the mapping distance in OxO, where $d = 2$ or $d = 3$.

   Hence, the worst case complexity at distance 2 is $3^2 \times 698\,650^2 = 4.393006402 \times 10^{12}$ and for distance 3 it is $3^3 \times 698\,650^2 = 1.317901921 \times 10^{13}$. Looking at these calculations it becomes clear why OxO1 crashed or timeout for large requests.

3.2. How OxO2 address these Challenges

To address the concerns regarding semantics, OxO2 implements SSSOM fully, and hence all metadata associated with mappings and mapping sets, are stored in OxO2. Its Application Programming Interface (API) allows searching across all SSSOM metadata elements (see Table 1). This provides data integrators with multiple ways in which they can find mappings, thereby increasing their chances to find the data most suitable to their needs.

In order to ensure inferences are logically sound, OxO2 makes use of a Datalog rule engine to infer mappings. Hence, OxO2 users can trust that inferred mappings are logically correct. To enable fast searches across inferred mappings, OxO2 materializes all chain rule inferences as part of its data release (see Section 2.4). Moreover, there is no longer a restriction on the distance at which mappings can be returned, or the length of crosswalks that can be identified by OxO2. The derived mappings of Listings 4 and 5 were derived using this approach, and the mapping sets from [23]. Mappings that are derived by OxO2, are placed in a mapping set with value `https://www.ebi.ac.uk/spot/oxo/inferences`, `mapping_justification` value `SEMAPV:MappingChaining` and `mapping_tool` value as `OxO2`.

SSSOM's 22 chain rules are helpful to define the precise conditions under which new mappings can be inferred that are logically sound. But because chain rules can be recursively applied multiple times, how inferred mappings were derived may not be obvious to humans. To make this information accessible to users, one needs to identify all the chain rules that have been applied, and the facts that were used, to derive an inferred mapping. We are able to provide this information by using a Datalog rule engine, that records this information as part of OxO2's data release process.

3.3. Benefits to Users

OxO2 will help our users in the following ways:

1. A researcher searching for "enlarged heart" will be able to find the MP concept `MP:0000274`, but they may be missing `ZP:0000532` (increased size of heart) and `HP:0001640` (Cardiomegaly). OxO2 will help them to uncover `ZP:0000532` and `HP:0001640`.
2. Logical soundness does not necessarily mean biological correctness. Assume for the moment a mapping was mistakenly added that states that `MP:0000274` (enlarged heart) is a `SKOS:exactMatch` with `MP:0001095` (enlarged trigeminal ganglion). This could result in inferred mappings that are logically sound, but which are nonsensical from a biological perspective. A researcher can identify an error like this by verifying the premises of an inference. As an example, we can verify the inferences of Listings 4 and 5 by verifying the premises in Table 2.
3. Derived mappings in OxO1 were not axiomatised, and hence, the only way researchers could verify each of the mappings, was to manually validate them. In the case of OxO2, one can assume that the inferred mappings are logically correct. Any incorrect inferred mappings will only ever be due to incorrect explicitly stated mappings. Thus, to validate the correctness of derived mappings, a user can consider reviewing explicitly stated mappings.
4. Complete explanations of inferred mappings will help data integrators to identify the source of incorrect data, since explanations clearly state the facts that were used to derive inferred mappings.

4. Methods

For materializing inferred mappings, OxO2 uses Nemo, an in-memory rule engine [28, 33]. Nemo runs as a command line tool that loads facts and rules, processes them, and, once complete, writes out the relevant inferences; then the Nemo process terminates. From this perspective Nemo is well aligned with being run as part of a data release pipeline, as is the case for OxO2. To ensure that inferencing is fast and memory efficient, Nemo implements the various optimizations mentioned in Section 2.4.

Nemo implements Datalog with various extensions, that results in inferencing not necessarily terminating (Section 2.3). However, OxO2's use of Nemo is such that it only uses Datalog without any extensions and, hence for its use case, Nemo terminates [33].

Currently OxO2 imports 1 160 020 mappings, from which it is able to infer 49 536 mappings. Inferencing on an Intel Core Ultra 7 165U x 13 laptop, with 32GB RAM and SSD, completes in about 17 min, and uses about 380MB of memory. These results show that it is possible to run an OxO2 dataload on a personal computer, without necessarily having access to high-performance computing infrastructure.

For storing mappings, OxO2 uses Solr rather than Neo4J. Our motivation for this change in architecture is due to there being no need for graph queries in OxO2 – all derived mappings with their explanations are determined during the OxO2 dataload. All inferred mappings and their explanations are then stored in Solr. The OxO2 Solr schema design mimics the SSSOM metadata elements with a core for storing mapping information and core for storing mapping set information. Hence, all SSSOM metadata elements are searchable and can be accessed via API calls and its frontend. Explanations are stored along with the mappings in the mapping core.

The OxO2 dataload and backend are implemented using Java 17 and Spring Boot. The frontend is implemented in React using Typescript. Styling is implemented using TailWind CSS, which provides numerous utility classes that limits the need for custom CSS.

5. Related Tools

In support of SSSOM there are a number of tools as detailed in [20]:

- **sssom-py** is a Python library for reading, transforming and manipulating SSSOM mappings.
- The **Ontology Access Kit (OAK)** can provide mappings in the SSSOM format.
- The **sssom-java library** is a Java library for reading, transforming and manipulating SSSOM mappings.

- **Semantic Reasoning Assembler (SeMRA)** is a Python library that allows one to crawl mappings in various formats, not just SSSOM. It then de-duplicates mappings, infers mappings, derives associated confidence for inferred mappings, and filters false positive mappings [34].

These tools are used for the creation and manipulation of SSSOM files. The main purpose of OxO2 is to make existing mappings and their derived mappings available for users to search and browse.

The value of SSSOM as a standard for mappings depends on its community adoption. Currently most mappings in the bioinformatics domain are still embedded within ontologies, which are extracted by the OLS dataload into SSSOM files that are imported by OxO2. But many of these mappings are based on xrefs and hence still suffer from the poor semantics we discussed in the Introduction. For this reason, the tools listed above are complementary to OxO2, in that they enable users to create high quality mappings that are essential to the success of OxO2.

6. Discussion and Future

With the development of OxO2, we are able to address the key limitations of OxO1 in terms of the semantics of mappings, the meaning of inferred mappings and the length of crosswalks it can support. It is only through the availability of SSSOM that OxO2 is able to address these concerns.

Currently OxO2 is still in active development and we will prioritise as follows:

1. Currently, the API of OxO2 focuses on bringing new search capabilities to users, with its own new request and response structures. For our existing users there may be a need to create a backwards compatibility layer, so as to not disrupt their services when OxO2 is rolled out. Similar to OLS4 [11], this will be implemented using view classes that transforms the underlying OxO2 data model into the OxO1 data model view.
2. The value proposition of OxO2 is dependent on high quality SSSOM mappings. We need to identify mappings sets that can be included in OxO2 such as mappings from MONDO to HP found at [35] and SeMRA which provides an SSSOM export of their mappings [36].
3. The explanation implementation is currently a proof of concept and contains only information on the `subject_id`, `predicate_id`, `object_id` and chain rule that has been applied. To provide sufficient provenance, it also needs to include information such as `author_id`, `reviewer_id`, and the mapping set from which premises came. Currently, explanations are given in a text form. This needs to be augmented with a graph that can easily be navigated by users.
4. To ensure the high availability of OxO2, its data release pipeline will be rolled out on the EMBL-EBI high-performance computing infrastructure, and its frontend and backend to the Kubernetes infrastructure.
5. The EMBL-EBI instance of OxO is focussed on the biological domain. To enable other domains to install their own instances of OxO2, OxO2 will be dockerized.

Acknowledgments

J.A.M., H.I., H.P., and H.H. are supported in part by EMBL- EBI Core Funds. J.M., H.I. and H.H are supported in part by EVORA. The EVORA project has received funding from the European Union's HORIZON programme under grant agreement No 101131959.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The FAIR Guiding Principles for scientific data management and stewardship, Scientific data 3 (2016) 1–9.

[2] F. Xu, N. Juty, C. Goble, S. Jupp, H. Parkinson, M. Courtot, Features of a FAIR vocabulary, Journal of Biomedical Semantics 14 (2023) 6.

[3] M. van Reisen, M. Stokmans, M. Basajja, A. O. Ong'ayo, C. Kirkpatrick, B. Mons, Towards the Tipping Point for FAIR Implementation, Data Intelligence 2 (2020) 264–275.

[4] S. Jupp, T. Liener, S. Sarntivijai, O. Vrousgou, T. Burdett, H. E. Parkinson, OxO-A Gravy of Ontology Mapping Extracts., in: ICBO, 2017.

[5] J. Malone, E. Holloway, T. Adamusiak, M. Kapushesky, J. Zheng, N. Kolesnikov, A. Zhukova, A. Brazma, H. Parkinson, Modeling sample variables with an Experimental Factor Ontology, Bioinformatics 26 (2010) 1112–1118.

[6] N. Vasilevsky, S. Essaid, N. Matentzoglu, N. L. Harris, M. Haendel, P. Robinson, C. J. Mungall, Mondo Disease Ontology: harmonizing disease concepts across the world, volume 2807, eScholarship, University of California, 2020.

[7] L. M. Schriml, J. B. Munro, M. Schor, D. Olley, C. McCracken, V. Felix, J. A. Baron, R. Jackson, S. M. Bello, C. Bearer, et al., The human disease ontology 2022 update, Nucleic acids research 50 (2022) D1255–D1261.

[8] S. Köhler, M. Gargano, N. Matentzoglu, L. C. Carmody, D. Lewis-Smith, N. A. Vasilevsky, D. Danis, G. Balagura, G. Baynam, A. M. Brower, et al., The human phenotype ontology in 2021, Nucleic acids research 49 (2021) D1207–D1217.

[9] L. Vogt, P. Strömert, N. Matentzoglu, N. Karam, M. Konrad, M. Prinz, R. Baum, Suggestions for extending the FAIR Principles based on a linguistic perspective on semantic interoperability, Scientific Data 12 (2025) 688.

[10] N. Matentzoglu, S. M. Bello, R. Stefancsik, S. M. Alghamdi, A. V. Anagnostopoulos, J. P. Balhoff, M. A. Balk, Y. M. Bradford, Y. Bridges, T. J. Callahan, H. Caufield, A. Cuzick, L. C. Carmody, A. R. Caron, V. de Souza, S. R. Engel, P. Fey, M. Fisher, S. Gehrke, C. Grove, P. Hansen, N. L. Harris, M. A. Harris, L. Harris, A. Ibrahim, J. O. B. Jacobsen, S. Köhler, J. A. McMurry, V. Munoz-Fuentes, M. C. Munoz-Torres, H. Parkinson, Z. M. Pendlington, C. Pilgrim, S. M. C. Robb, P. N. Robinson, J. Seager, E. Segerdell, D. Smedley, E. Sollis, S. Toro, N. Vasilevsky, V. Wood, M. A. Haendel, C. J. Mungall, J. A. McLaughlin, D. Osumi-Sutherland, The Unified Phenotype Ontology : a framework for cross-species integrative phenomics, Genetics 229 (2025) iyaf027. doi:10.1093/genetics/iyaf027.

[11] J. McLaughlin, J. Lagrimas, H. Iqbal, H. Parkinson, H. Harmse, OLS4: A new Ontology Lookup Service for a growing interdisciplinary knowledge ecosystem, Bioinformatics (2025).

[12] J. Day-Richter, The OBO flat file format specification, version 1.2, The Gene Ontology (2006).

[13] A. Laadhar, E. Abrahão, C. Jonquet, Investigating One Million XRefs in Thirthy Ontologies from the OBO World., in: ICBO/ODLS, 2020, p. 1–12.

[14] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, et al., The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, Nature biotechnology 25 (2007) 1251–1255.

[15] T. Cezard, F. Cunningham, S. E. Hunt, B. Koylass, N. Kumar, G. Saunders, A. Shen, A. F. Silva, K. Tsukanov, S. Venkataraman, et al., The European Variation Archive: a FAIR resource of genomic variation for all species, Nucleic acids research 50 (2022) D1216–D1220.

[16] M. Cerezo, E. Sollis, Y. Ji, E. Lewis, A. Abid, K. O. Bircan, P. Hall, J. Hayhurst, S. John, A. Mosaku, et al., The NHGRI-EBI GWAS Catalog: standards for reusability, sustainability and diversity, Nucleic acids research 53 (2025) D998–D1005.

[17] E. P. Consortium, Europe PMC: a full-text literature database for the life sciences and platform for innovation, Nucleic acids research 43 (2015) D1042–D1048.

[18] N. Matentzoglu, J. P. Balhoff, S. M. Bello, C. Bizon, M. Brush, T. J. Callahan, C. G. Chute, W. D. Duncan, C. T. Evelo, D. Gabriel, J. Graybeal, A. Gray, B. M. Gyori, M. Haendel, H. Harmse, N. L.

Harris, I. Harrow, H. B. Hegde, A. L. Hoyt, C. T. Hoyt, D. Jiao, E. Jiménez-Ruiz, S. Jupp, H. Kim, S. Koehler, T. Liener, Q. Long, J. Malone, J. A. McLaughlin, J. A. McMurry, S. Moxon, M. C. Munoz-Torres, D. Osumi-Sutherland, J. A. Overton, B. Peters, T. Putman, N. Queralt-Rosinach, K. Shefchek, H. Solbrig, A. Thessen, T. Tudorache, N. Vasilevsky, A. H. Wagner, C. J. Mungall, A Simple Standard for Sharing Ontological Mappings (SSSOM), Database 2022 (2022).
[19] N. Matentzoglu, J. E. Flack, J. Graybeal, N. L. Harris, H. B. Hegde, C. T. Hoyt, H. Kim, S. Toro, N. A. Vasilevsky, C. J. Mungall, A Simple Standard for Ontological Mappings 2022: Updates of data model and outlook, in: OM@ ISWC, 2022, p. 61–66.
[20] N. Matentzoglu, I. Braun, A. R. Caron, D. Goutte-Gattat, B. M. Gyori, N. L. Harris, E. Hartley, H. B. Hegde, S. Hertling, C. T. Hoyt, et al., A simple standard for ontological mappings 2023: updates on data model, collaborations and tooling., in: OM@ ISWC, 2023, p. 73–78.
[21] Simple Standard for Sharing Ontological Mappings (SSSOM), 2025. URL: https://mapping-commons.github.io/sssom/.
[22] CURIE Syntax 1.0, 2010. URL: https://www.w3.org/TR/curie/.
[23] Disease Mappings, 2024. URL: https://github.com/mapping-commons/disease-mappings.
[24] YAML: YAML Ain't Markup Language™, 2021. URL: https://yaml.org/.
[25] S. Abiteboul, R. Hull, V. Vianu, Foundations of databases, volume 8, Addison-Wesley Reading, 1995.
[26] Simple Standard for Sharing Ontological Mappings (SSSOM) - Applying Chaining Rules, 2025. URL: https://mapping-commons.github.io/sssom/chaining-rules/.
[27] SKOS Simple Knowledge Organization System Namespace Document, 2009. URL: https://www.w3.org/2009/08/skos-reference/skos.html.
[28] A. Ivliev, S. Ellmauthaler, L. Gerlach, M. Marx, M. Meißner, S. Meusel, M. Krötzsch, Nemo: First glimpse of a new rule engine, arXiv preprint arXiv:2308.15897 (2023).
[29] S. Ceri, G. Gottlob, L. Tanca, et al., What you always wanted to know about Datalog(and never dared to ask), IEEE transactions on knowledge and data engineering 1 (1989) 146–166.
[30] M. Davis, R. Sigal, E. J. Weyuker, Computability, complexity, and languages: fundamentals of theoretical computer science, Elsevier, 1994.
[31] A. Elhalawati, M. Krötzsch, S. Mennicke, An Existential Rule Framework for Computing Why-Provenance On-Demand for Datalog, in: G. Governatori, A.-Y. Turhan (Eds.), Rules and Reasoning, Springer International Publishing, Cham, 2022, p. 146–163.
[32] J. Urbani, C. Jacobs, M. Krötzsch, Column-oriented datalog materialization for large knowledge graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.
[33] A. Ivliev, L. Gerlach, S. Meusel, J. Steinberg, M. Krötzsch, Nemo: Your friendly and versatile rule reasoning toolkit, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 21, 2024, p. 743–754.
[34] C. T. Hoyt, K. Karis, B. M. Gyori, Assembly and reasoning over semantic mappings at scale for biomedical data integration, bioRxiv (2025) 2025–04.
[35] MONDO to HP SSSOM mappings, 2024. URL: https://github.com/mapping-commons/disease-mappings/blob/main/mappings/mondo_hp_lexical.sssom.tsv.
[36] SeMRA Mappings, 2025. URL: https://zenodo.org/records/15208251.

A. Online Resources

- OxO2 GitHub repository,
- Simple Standard for Sharing Ontological Mappings (SSSOM).