Catching inconsistencies with the semantic web: a biocuration case study.

Jerven Bolleman*[1] , Sebastien Gehant[1] , the UniProt Consortium [1,2,3,4]

[1]SIB Swiss Institute of Bioinformatics, Centre Medical Universitaire, 1 rue Michel Servet, 1211 Geneva 4, Switzerland, [2]The EMBL Outstation, The European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK, [3]Protein Information Resource, Georgetown University Medical Center, 3300 Whitehaven St NW, Suite 1200, Washington, DC 20007, and [4]University of Delaware, 15 Innovation Way, Suite 205, Newark, DE 19711, USA

Email: Jerven Bolleman*- jerven.bolleman@isb-sib.ch; Sebastien Gehant - sebastien.gehant@isb-sib.ch; UniProt Consortium - help@uniprot.org;

*Corresponding author

Abstract

Background The UniProtKB/Swiss-Prot database is manually curated by a team of experienced biocurators with the aim to provide to the scientific community high-quality information on proteins. Ensuring a high-quality curation standard depends in part on effective tools that help curators to avoid trivial mistakes during data curation.

Description We describe here a system that is using SPARQL queries encoded in SPIN to identify UniProtKB database records that do not comply with manual curation rules. The system must generate specific and accurate warnings for curators by correctly defining known exceptions to general rules.

Conclusions Semantic web technologies such as SPARQL queries are a good way to encode quality control rules for manual curation efforts in the life sciences because they are simple and cheap to maintain. This is an important factor in the face of continuously growing and evolving knowledge about biology. The results of SPARQL queries can be presented in a user-friendly way to help curators with data correction.

1 Keywords

SPARQL, UniProt, RDF, quality assurance, biocuration, parallel, semantic web

2 Background

The mission of UniProt [16] is to provide the scientific community with a comprehensive, high-quality and freely accessible resource of protein sequence and functional information. The UniProt Knowledgebase (UniProtKB) is compiled by using a systematic approach for protein annotation that interprets and standardizes data that is integrated from a large number of different sources. It consists of two sections: UniProtKB/Swiss-Prot contains manually-curated records with information extracted from the literature and curator-evaluated computational analysis, while UniProtKB/TrEMBL records are generated fully automatically from data that is imported from other databases and computationally analyzed.

Maintaining the high quality of the manually-curated UniProtKB/Swiss-Prot section is expensive [15], since data about protein biology is being published at an ever increasing rate [17] and our understanding of biological processes is constantly evolving. There are a number of different factors that contribute to a high-quality curation standard. The expertise of the curators and manual quality assurance protocols are no doubt the most crucial factor, but human resources is at the same time the most costly factor. Good software for data curation is therefore an important tool for keeping costs at bay: It must free curators from repetitive tasks and help them to avoid common mistakes, thereby allowing them to focus on the biological questions while maintaining a high level of data consistency.

There are several sources of errors in annotating UniProtKB: There are errors in the scientific literature, which only scientific progress will correct (see [4]), and errors introduced by data submitters and curators. Errors in the data submission process affect UniProtKB/TrEMBL and must be corrected by the original submitters in the EMBL-Bank/Genbank/DDBJ database [21]. Manual curation errors are minimized by following a standard operating procedure [10] that includes the application of curation rules and manual quality control steps before integrating data into UniProtKB/Swiss-Prot. This was shown to yield a very low error rate [23].

3 Implementation

3.1 Database schema and curation rules

A database schema can define more or less complex data constraints, depending on the expressive power of the schema definition language that is used. Many databases need to enforce rules that cannot easily be expressed at the level of the schema and typically implement these rules in a layer of custom software.

UniProtKB is published in a number of data formats: flat text, XML, and RDF/XML. We use OWL [24] to describe the schema of the UniProtKB RDF representation [11]. OWL is a vocabulary for describing the

classes and properties of RDF resources and allows definition of generalization-hierarchies of properties and classes, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, characteristics of properties (e.g. symmetry), and enumerated classes. OWL is a very expressive language, but adopts the "open world assumption" [7], while classical database schema languages make a "closed world assumption" to validate data constraints. There exist, however, OWL reasoners with extensions for "closed world reasoning" that permit instance validation [9].

For curation rules that cannot be described with OWL, we build SPARQL1.1 [26] syntax CONSTRUCT queries using the UniProtKB OWL schema ontology and the SPIN [25] constraint vocabulary. SPARQL CONSTRUCT queries can "build" new information encoded in RDF triples. In our case the information we build is a warning message which is generated when the triples of a UniProtKB record match the graph pattern of the query's SELECT clause (see listing 1). The constructed RDF triples use the SPIN vocabulary, allowing generic tools that implement the SPIN API to show the warnings. The SPARQL 1.1. queries can be as specific as our RDF representation allows.

The rules are applied with a Java application that is using the open source SPIN API and the Jena [1] framework. The default implementation works with an in-memory Jena ARQ engine [2] or a SPARQL endpoint. The system currently contains 137 rules of which 110 are ready for production. We expect to develop another 100 rules in the coming year. New curation rules are easy to create. The quality assurance team typically requests a new rule when they notice the same type of error repeatedly. They hand over a set of records that are incorrect. The WHERE clause of the new rule can be composed by copying the triples that define the problem from the RDF representation of the incorrect records and translating them into a generic graph pattern. Once the basic query is written, it can be tested on the entire database and iteratively refined in consultation with the curators. The rules are currently all created by the software development team, but it is envisageable that some members of the quality assurance team could be trained to create rules. Learning the necessary SPARQL and SPIN syntax requires considerably less effort than learning a programming language in order to modify a custom rule system software.

3.2 Usability

An important usability aspect is that the warnings must be easy to act on. The message that is generated by a triggered rule should effectively tell the curator how to correct the database record. Furthermore, it is crucial to minimize false positive warnings in order to ensure that curators do not start to dismiss the warnings they receive. Listing 1 shows a simple rule: "Proteins with a homeobox are DNA binding". This

is nearly always true, but there are some UniProtKB records where this is known to be false. To avoid false positive matches, the rule must encode these exceptions by matching information in those records that indicates that the general rule should not apply.

The rule system must be seamlessly integrated with the UniProtKB/Swiss-Prot curation platform in a way that requires no knowledge of RDF by curators. When they curate a UniProtKB record, they want to click a "Validate" button and see the warnings of the rule system displayed next to the data that needs correction. The main difficulty is to associate the results of a triggered rule with the correct graphical user interface elements of the curation platform. This can be achieved in the following way: The curation platform serializes the records in N-Quads format [6], where each triple is associated with an International Resource Identifier (IRI) that defines the user interface element that displays the data corresponding to a triple to the curator. The rule system processes the quads and returns, in addition to the warning message, the IRI of each triple that violates a rule. This allows the curation platform to display the warning message next to the data that triggered the rule.

Listing 1: Warn about missing "DNA-binding" keyword.

```
PREFIX up:<http://purl.uniprot.org/core/>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX keyword:<http://purl.uniprot.org/keywords/>
PREFIX spin:<http://spinrdf.org/spin#>
CONSTRUCT {
    # Build a constraint violation
    [] a spin:ConstraintViolation ;
    # The resource to attach the warning to:
    # In this case a UniProtKB protein record.
       spin:violationRoot ?this ;
       spin:violationPath up:classifiedWith ;
    # Add a curator readable warning message.
       rdfs:label "URC90: If keyword:'Homeobox[KW-0371]' is present
                          we expect keyword:'DNA-binding[KW-0238]'" .
} WHERE {
    ?this a up:Protein .
    ?this up:classifiedWith keyword:371 . # Homeobox
    FILTER ( NOT EXISTS {
        ?this up:classifiedWith keyword:238 . # DNA-Binding
    }) .
    FILTER ( NOT EXISTS {
        # Find the known exceptions.
        ?this up:annotation ?annotation .
        ?annotation a up:Nucleotide_Binding_Annotation ;
            rdfs:comment ?comment .
        ?this up:annotation ?functionalAnnotation .
        ?functionalAnnotation a up:Function_Annotation ;
            rdfs:comment ?functionalComment .
        FILTER (contains(?comment, "Homeobox; atypical")
             && contains(?functionalComment, "does not bind DNA")).
    }) .
```

}

3.3 Automatic corrections

Some rules have no exceptions: for instance, when a protein was experimentally found in a specific subcellular location, the curator will annotate the subcellular location using a controlled vocabulary and must add the corresponding keyword to the record. Instead of issuing a warning when the curator forgets to add the keyword, a SPIN rule could automatically add the missing keyword (see listing 2).

Listing 2: Add keyword "Amyloplast" when the protein is annotated with the subcellular location "Amyloplast".

```
PREFIX up:<http://purl.uniprot.org/core/>
PREFIX sl:<http://purl.uniprot.org/locations/>
PREFIX keyword:<http://purl.uniprot.org/keywords/>
PREFIX rules:<http://swissprot.isb-sib.ch/rules/>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT {
    # Add the keyword "Amyloplast"
    ?protein up:classifiedWith keyword:35 .
} WHERE {
    ?protein a up:Protein ;
        up:annotation sl:12 .
    FILTER (NOT EXISTS {?protein up:classifiedWith keyword:35 .}) .
}
```

3.4 Maintenance

Rules need to be updated from time to time to match the evolution of biological knowledge. When curators find new exceptions to a rule, the rule must be updated accordingly. Each rule has a number that curators can use to send feedback to an issue tracking system. This permits the collection of all discussions about the rule as well as its implementation history.

Rules may become ineffective when the data that they rely on for the detection of a problem has been updated. A trivial real-world example is rules that need to be updated or deleted because they depend on a taxonomy identifier that became obsolete. It is important to detect such changes as soon as possible to avoid mistakes. Since SPIN rules are encoded in RDF, they can be checked by other SPIN rules. This permits daily checking with an automatic procedure. Listing 3 shows a rule that finds rules that mention an obsolete taxonomy database record. Besides checking the rules for the validity of the data they rely on, they are also checked for conformance to the UniProtKB OWL schema ontology, e.g. to detect predicates that have been obsoleted.

Each rule is tested by JUnit [5] tests. A JUnit test sets up test data and verifies that the rule is triggered when presented with incorrect data and is not triggered when presented with correct data. Both cases are required in a test to validate correct behavior.

Listing 3: Warn about SPIN rules with an obsolete taxon.

```
PREFIX spin:<http://spinrdf.org/spin#>
PREFIX sp:<http://spinrdf.org/sp#>
PREFIX up:<http://purl.uniprot.org/core/>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT {
    [] a spin:ConstraintViolation ;
       spin:violationRoot ?this ;
       spin:violationPath sp:Construct ;
       rdfs:label "Rule must not mention an obsolete taxon." .
} WHERE {
    ?this spin:constraint ?constraint .
    ?constraint a sp:Construct .
    # Find the WHERE−clause in another rule.
    ?constraint sp:where ?list .
    # Property expression to find all elements in a list.
    ?list rdf:first|((rdf:rest/rdf:first))* ?item ;
         sp:predicate up:organism ;
         sp:object ?object .
    ?object a up:Taxon ;
         up:obsolete true .
}
```

3.5 Sharing rule knowledge

Rules are presented on a simple website (see figure 1) and users can search for specific rules using SPARQL queries. Users can run these queries on their own data, as long as their triple patterns match the UniProtKB OWL schema ontology. This could be useful for genome annotation projects that would like to improve the annotation of their protein predictions. Since the rules are SPARQL queries, they are easy to adapt and extend to other groups' use cases and data.

The application of SPIN rules could be extended to other annotation sources, for example those providing annotations in the form of terms from the Gene Ontology (GO) [12]. One provider of such annotations is the InterPro resource of protein family and domain signatures [19]. It manually curates GO terms for InterPro records and then projects these GO terms in a fully automated fashion (labelled with the "IEA" evidence code [3]) to all UniProtKB records that match an InterPro signature with the InterPro2GO procedure [14]. While great care is taken to minimize false positive annotations, every automatic approach comes with a certain error rate. SPIN rules can help to further reduce these errors by providing additional

checks to detect GO terms that are inconsistent with the curated content of the UniProtKB record. For instance, the InterPro2GO procedure assigns the GO term GO:0043565, "sequence-specific DNA binding", to the proteins that the rule in listing 1 specifically excludes from annotation with the equivalent UniProtKB keyword "DNA-binding".

Figure 1: Rule URC100 is shown with options for downloads.

4 Results and discussion

4.1 Corrections to UniProtKB

At the time of writing all records pass the rules that have been validated by curators. 81 manually curated records have been updated. 2 UniRules [13] have been corrected due to feedback from this system and this affected 3,897 UniProtKB/TrEMBL and 852 UniProtKB/Swiss-Prot entries[1]. This means that the system improved the quality of UniProtKB before entering production.

4.2 Computational performance

While SPARQL and RDF are often considered to be slow and therefore impractical for big databases, our tests found that the performance was acceptable for our purposes, even on modest hardware. Using the in-memory Jena ARQ engine, and working with batches of 3000 entries at a time, a 4 CPU (Intel X7350 64GB of RAM, but only 4GB java heap) server run 137 rules against UniProtKB release 2011_12 (21 million records, 3 billion triples) in 23 hours and 15 minutes. The approximately 550,000 manually curated records can be checked within 3.5 hours on a small virtual server (2 core Intel E5540 assigned with 6GB of RAM but only 2GB java heap). On high-end hardware the same rules can be applied against 5.3 billion triples in one hour. This test was performed using the public beta.sparql.uniprot.org SPARQL endpoint, which is running OWLIM 5.2 [22] on 4*16 cores AMD opteron 6276 CPUs with 256GB of RAM of which 100GB java heap, and 2*7,200 RPM SAS (Seagate ST200NM0011) disks with XFS file system on CentOS 6.2 in Raid 1.

To our knowledge at the time of writing, no RDF triple store implements transactional consistency checks using SPIN rules. Therefore, checking all records in bulk is the only option today. But because all rules are independent of each other, and the same is true of UniProtKB records, this problem is embarrassingly

[1] Counts for UniProt release 2012_08 of September 5th 2012

parallel, and the run time is bound by the smallest piece of work.

4.3 Comparison to other approaches

We use OWL to describe the schema of the UniProtKB RDF representation and SPIN's OWL2-RL reasoner [8] to validate our RDF data with the schema ontology (see section 3.1). Using OWL as a schema language to validate RDF data works well for data that can be described with controlled vocabularies and ontologies. This approach has been successfully used to improve the quality of curated data [20] and of biological models [18]. But UniProtKB records contain also many literals with human readable text. Constraints that use the presence of information inside a literal cannot be expressed with OWL constructs (see listing 1). While it is desirable to replace these literals with formal vocabularies and ontologies, it would require significant curation resources to achieve this with a database like UniProtKB/Swiss-Prot, which has evolved over 25 years and whose primary target users are human readers. We therefore need an additional quality control system that can handle such constraints. While any Turing-complete language could be used instead of SPIN and SPARQL to encode and apply curation rules, the maintenance cost will generally be higher as the code and data are encoded in different and non-standard ways. There is therefore no easy method to check whether the code is still up-to-date with respect to the data that it is supposed to check. When using SPIN, the rule and the data are both expressed in RDF and both are accessible with SPARQL, which facilitates rule maintenance as shown in listing 3. Another major benefit of using a specialized rule language is that the rules can be shared by several applications that are written in different programming languages. At Swiss-Prot, for instance, the curation platform is written in C++, while the database production pipelines are mostly coded in Java and Perl. SPIN allows us to use the same rules in all systems. The only requirement is that the language has a SPARQL/RDF API.
Another issue with OWL reasoning is that it is difficult to validate that the model is up-to-date: Defining an axiom that detects other axioms that use obsolete concepts raises the complexity of the ontology into OWL2-full with its associated decidability issues. Such validations are trivial to implement with SPIN rules (see listing 3).

5 Conclusions

The use of SPIN rules to find data that is inconsistent or incorrect is a practical approach to assist curators to maintain a consistent annotation quality. SPIN rules are accurate and fast enough for daily use and have the benefit of low maintenance costs.

6 Availability and requirements

Project name UniProtKB/Swiss-Prot quality rules

Operating system Any

Programming language SPARQL 1.1 (draft of 5th of January 2012), Java

7 List of abbreviations

API Application Programming Interface

DNA Deoxyribonucleic acid

IRI International Resource Identifier

RDF Resource Description Framework

OWL Web Ontology Language

SPARQL SPARQL Protocol and RDF Query Language

SPIN SPARQL Inferencing Notation

UniProtKB Universal Protein Knowledgebase

8 Authors contributions

Jerven Bolleman wrote the SPIN rules, embedded these into a framework, executed the experiments and wrote the manuscript. Sebastien Gehant integrated the SPIN rules with the UniProtKB curation platform and reviewed the manuscript. Nicole Redaschi reviewed and revised the manuscript. The Swiss-Prot group members provided knowledge on what the rules should check, as well as feedback during the writing of the software and this article.

9 Acknowledgments

This activity at the SIB Swiss Institute of Bioinformatics is mainly supported by the Swiss Federal Government through the Federal Office of Education and Science, by the National Institutes of Health (NIH) grant 1U41HG006104-02, and from the European Commission contracts GEN2PHEN (200754),

MICROME (222886-2) and SLING (226073). The hardware platform is provided by the Vital-IT (http://www.vital-it.ch) Center for high-performance computing of the SIB Swiss Institute of Bioinformatics.

References

1. Apache jena, http://incubator.apache.org/jena
2. Arq - a sparql processor for jena, http://jena.apache.org/documentation/query/index.html
3. Guide to go evidence codes, http://www.geneontology.org/GO.evidence.shtml
4. The importance of being manual, http://www.uniprot.org/news/2012/03/21/release
5. junit, http://www.junit.org
6. N-quads specification, http://sw.deri.org/2008/07/n-quads/
7. Open world assumption, http://en.wikipedia.org/wiki/Open_world_assumption
8. Owl 2 rl in sparql using spinl, http://composing-the-semantic-web.blogspot.ch/2009/01/owl-2-rl-in-sparql-using-spin.html
9. Pellet integrity constraints: Validating rdf with owl, http://www.clarkparsia.com/pellet/icv
10. Standard operating procedure (sop) for uniprot manual curation, http://www.uniprot.org/docs/sop_manual_curation.pdf
11. Uniprot schema ontology, http://www.uniprot.org/core/
12. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene ontology: tool for the unification of biology. Nat Genet 25, 25–29 (2000), http://dx.doi.org/10.1038/75556
13. Bridge, A.J., Poggioli, D., O'Donovan, C.: Unirule - automatic annotation in uniprotkb. In: Biocuration 2010. Biocuration 2010 (2010), http://dx.doi.org/10.1038/npre.2010.5247.1
14. Burge, S., Kelly, E., Lonsdale, D., Mutowo-Muellenet, P., McAnulla, C., Mitchell, A., Sangrador-Vegas, A., Yong, S.Y., Mulder, N., Hunter, S.: Manual go annotation of predictive protein signatures: the interpro approach to go curation. Database 2012 (2012), http://database.oxfordjournals.org/content/2012/bar068.abstract
15. Chandras, C., Weaver, T., Zouberakis, M., Smedley, D., Schughart, K., Rosenthal, N., Hancock, J.M., Kollias, G., Schofield, P.N., Aidinis, V.: Models for financial sustainability of biological databases and resources. Database 2009 (2009), http://database.oxfordjournals.org/content/2009/bap017.abstract
16. Consortium, T.U.: The universal protein resource (uniprot) in 2010. Nucl. Acids Res. 38 (suppl 1), D142–D148 (2010), http://nar.oxfordjournals.org/content/early/2009/10/20/nar.gkp846
17. Friedberg, I.: Automated protein function prediction—the genomic challenge. Briefings in Bioinformatics 7(3), 225–242 (2006), http://bib.oxfordjournals.org/content/7/3/225.abstract
18. Hoehndorf, R., Dumontier, M., Gennari, J., Wimalaratne, S., de Bono, B., Cook, D., Gkoutos, G.: Integrating systems biology models and biomedical ontologies. BMC Systems Biology 5(1), 124 (2011), http://www.biomedcentral.com/1752-0509/5/124
19. Hunter, S., Jones, P., Mitchell, A., Apweiler, R., Attwood, T.K., Bateman, A., Bernard, T., Binns, D., Bork, P., Burge, S., de Castro, E., Coggill, P., Corbett, M., Das, U., Daugherty, L., Duquenne, L., Finn, R.D., Fraser, M., Gough, J., Haft, D., Hulo, N., Kahn, D., Kelly, E., Letunic, I., Lonsdale, D., Lopez, R., Madera, M., Maslen, J., McAnulla, C., McDowall, J., McMenamin, C., Mi, H., Mutowo-Muellenet, P., Mulder, N., Natale, D., Orengo, C., Pesseat, S., Punta, M., Quinn, A.F., Rivoire, C., Sangrador-Vegas, A., Selengut, J.D., Sigrist, C.J.A., Scheremetjew, M., Tate, J., Thimmajanarthanan, M., Thomas, P.D., Wu, C.H., Yeats, C., Yong, S.Y.: Interpro in 2011: new developments in the family and domain prediction database. Nucleic Acids Research 40(D1), D306–D312 (2012), http://nar.oxfordjournals.org/content/40/D1/D306.abstract
20. Jeong, E., Nagasaki, M., Ueno, K., Miyano, S.: Ontology-based instance data validation for high-quality curated biological pathways. BMC Bioinformatics 12(Suppl 1), S8 (2011), http://www.biomedcentral.com/1471-2105/12/S1/S8
21. Leinonen, R., Akhtar, R., Birney, E., Bower, L., Cerdeno-Tárraga, A., Cheng, Y., Cleland, I., Faruque, N., Goodgame, N., Gibson, R., Hoad, G., Jang, M., Pakseresht, N., Plaister, S., Radhakrishnan, R., Reddy, K., Sobhany, S., Ten Hoopen, P., Vaughan, R., Zalunin, V., Cochrane, G.: The european nucleotide archive. Nucleic Acids Research 39(suppl 1), D28–D31 (2011), http://nar.oxfordjournals.org/content/39/suppl_1/D28.abstract

22. Ontotext AD: OWLIM 5.2 Standard Edition (2012), http://owlim.ontotext.com/display/OWLIMv52/OWLIM-SE
23. Schnoes, A.M., Brown, S.D., Dodevski, I., Babbitt, P.C.: Annotation error in public databases: Misannotation of molecular function in enzyme superfamilies. PLoS Comput Biol 5(12), e1000605 (12 2009), http://dx.doi.org/10.1371%2Fjournal.pcbi.1000605
24. W3C: OWL Web Ontology Language Overview (2004), http://www.w3.org/TR/owl-features/
25. W3C: SPIN - Overview and Motivation (2011), http://www.w3c.org/Submission/2011/SUBM-spin-overview-20110222
26. W3C: SPARQL 1.1 Query Language (2012), http://www.w3.org/TR/sparql11-query