Example 1: Atypical homeoboxes don’t bind DNA.

Avoiding false positives
There are a few proteins with atypical homeobox patterns that do not bind DNA and we must not raise an error for these. Instead of hard coding such exceptions, we use whenever possible information in the record to avoid false positives.

Rules for rules
Maintaining rules is as important as implementing them. As both our insight into biology and our database change, we need to adapt our rules accordingly.

Since SPIN rules are encoded in RDF, like our data, we can easily make SPIN rules on SPIN rules. Example 2 shows a rule that checks that all rules contain only valid taxonomy identifiers. Comparable checks would be hard to express in languages where data is encoded differently from code.

Example 2: Rule checking for obsolete taxonomy identifiers in other rules.

Further work
The SPIN rules must be integrated into the C++ based curation platform. SPIN violations must be mapped to GUI elements to show the errors in the context in which they occur to make it easy for curators to fix them.

We also need to validate that the SPIN rules cover at least all rules that are implemented by the legacy Perl based system.

   http://dx.doi.org/doi:10.1093/nar/gkr981.

Performance
SPARQL is considered to be a slow technology, but SPIN rules are trivial to run in parallel due to rule independence. In this case study, it took 23 hours to check all 21 million UniProtKB/Swiss-Prot entries against 137 rules on a 4 CPU Intel X7350 64 GB RAM machine. The 0.5 million UniProtKB/Swiss-Prot entries can be processed within 3 hours on a 2 CPU virtual machine. It takes on average 1/4 of a millisecond to check one rule against one entry.

Violation triples
We build 4 triples whenever a constraint is violated to make the violation accessible as an object which describes the violating resource (via violationRoot and violationPath) and provides a human readable error message. The violating resource belongs to a (not shown) named graph which links the resource with the appropriate GUI element of the curation platform. This link is used to display the error message next to the data that triggered the error, in this example, the presence of the “Homeobox” keyword without its expected complement “DNA-binding”.

Finding “bad” data
SPARQL works by defining graph patterns to match. In this example, we are looking for a “Protein” (UniProtKB entry) that is classified with the keyword “Homeobox”, while excluding the entries which also have the keyword “DNA-binding” as those are correct.

Powered by Vital-IT

Jerven Bolleman1, Sebastien Gehant1 and the UniProt Consortium1,2,3
1Swiss-Prot Group, SIB Swiss Institute of Bioinformatics, rue Michel-Servet 1, 1211 Geneva 4, Switzerland
2European Bioinformatics Institute (EBI), Welcome Trust Genome Campus, Hinxton Cambridge CB101SD UK
3Protein Information Resource (PIR), Georgetown University Medical Center, 3300 Whitehaven Street, NW, Suite 1200, Washington, DC 20007, USA

UniProt is a “Knowledgebase”

The UniProt Knowledgebase (UniProtKB)1 consists of two sections:
• UniProtKB/Swiss-Prot contains manually reviewed records with annotation extracted from the literature and curators evaluated computational analysis.
• UniProtKB/TREMBL contains computationally generated records enhanced by automatic classification and annotation.

One of the central activities of the UniProt Consortium is the biocuration of UniProtKB/Swiss-Prot. Providing high quality and consistent annotation is a shared curation objective among curators who apply common curation standards. Customized curation tools help to apply standards and to avoid trivial mistakes, thereby allowing curators to focus on the scientific content.

We propose to use SPIN to encode and apply rules that ensure curation standards.

Example 1: Atypical homeoboxes don’t bind DNA.

Avoiding false positives
There are a few proteins with atypical homeobox patterns that do not bind DNA and we must not raise an error for these. Instead of hard coding such exceptions, we use whenever possible information in the record to avoid false positives.

Rules for rules
Maintaining rules is as important as implementing them. As both our insight into biology and our database change, we need to adapt our rules accordingly.

Since SPIN rules are encoded in RDF, like our data, we can easily make SPIN rules on SPIN rules. Example 2 shows a rule that checks that all rules contain only valid taxonomy identifiers. Comparable checks would be hard to express in languages where data is encoded differently from code.

Example 2: Rule checking for obsolete taxonomy identifiers in other rules.

Further work
The SPIN rules must be integrated into the C++ based curation platform. SPIN violations must be mapped to GUI elements to show the errors in the context in which they occur to make it easy for curators to fix them.

We also need to validate that the SPIN rules cover at least all rules that are implemented by the legacy Perl based system.

   http://dx.doi.org/doi:10.1093/nar/gkr981.