Deploying the Mutation Impact mining pipeline with SADI: an exploratory case study

Alexandre Riazanov, Jonas Bergman Laurila and Christopher J O Baker*

Department of Computer Science & Applied Statistics, University of New Brunswick, Saint John, New Brunswick, E2L 4L5, Canada.

Email: Alexandre Riazanov - alexr@unb.ca; Jonas Bergman Laurila - j02h9@unb.ca; Christopher J O Baker* - bakerc@unb.ca;

*Corresponding author

Abstract

Our previous work on text mining for mutation impacts resulted in (i) the development of a GATE-based pipeline that mines texts for information about impacts of mutations on proteins, (ii) the population of this information into our OWL DL mutation impact ontology, and (iii) establishing an experimental OWL database for storing the results of text mining. The current focus of the project is to look for ways of deploying our software and data to facilitate the integration of our mutation impact data in a broader biological context.

This paper explores the possibility of using the SADI framework as a medium for publishing our mutation impact software and data. SADI is a set of conventions for creating web services with semantic descriptions that facilitate automatic discovery and orchestration. Here we describe a case study exploring and demonstrating the utility of the SADI approach in our context. We describe several SADI services we created based on our text mining API and data, and demonstrate how they can be used in a number of biologicaly meaningful scenarios through a SPARQL interface (SHARE) to SADI services. In all cases we pay special attention to the integration of mutation impact services with external SADI services providing information about related biological entities, such as proteins, pathways, and drugs.

Introduction.

The annotation of mutants with their consequences is central task for researchers investigating the role of genetic changes on biological systems and organisms. These annotations facilitate the reuse and re-interpretation of mutations and are necessary for the establishment of a comprehensive understanding of genetic mechanisms, biological processes and the resulting mutant phenotypes. As a result there are numerous mutation databases, albeit perpetually out of date and often with a latency of many years. Automated mutation extraction systems based on text mining techniques can identify and deliver mutation annotations for database curators to review. In this paper we outline the publication of a mutation impact extraction system in the form of semantic web services, and their integration with other semantically described bioinformatics services, based on the SADI framework.

In our previous work we developed the Mutation Impact pipeline [1] – a program, based on a GATE [2] pipeline, that makes it possible to extract mutation impacts on protein properties from texts, categorising the directionality of impacts as positive, negative or neutral. Moreover, the system *grounds mentions of proteins and mutations* to their respective UniProt identifiers and protein properties described in the Gene Ontology.

For example, consider these two excerpts from [3]: "The *haloalkane dehalogenase* from the nitrogen-fixing hydrogen bacterium *Xanthobacter autotrophicus* GJ10 (DhlA) prefers 1,2-dichloroethane (DCE) as substrate and converts it to 2-chloroethanol and chloride" and "DhlA shows only a small *decrease in activity* when *Trp-125 is replaced with phenylalanine*". Our pipeline (i) identified "*haloalkane dehalogenase*" as a protein, (ii) mapped it to the UniProt ID P22643 by grounding it to the identified organism "*Xanthobacter autotrophicus*", (iii) identified "*Trp-125 is replaced with phenylalanine*" as the point mutation W125F, (iv) identified "*activity*" as a protein property (GO_00188786 in the Gene Ontology, and (v) identified "*decrease*" as the direction of the impact of the mutation on the protein property.

Until now the Mutation Impact pipeline has been deployed as a simple Java API and could only be used programmatically. When the pipeline is executed on a document, it computes a sequence of Java objects representing mutation specifications. Every such object contains information about a series of elementary mutations, the corresponding wildtype and mutant proteins, and the discovered impacts of the mutations. The Java class representing impacts contains the direction of an impact and the type of the protein property being affected.

The practical use of the system and its results in this form is limited, because it requires programming. Recently in [1] we also explored the possibility of using semantic technologies for exporting the system outputs according to a domain specific knowledge representation. Now our system, like [4], delivers its results in the form of an OWL ABox, i.e., as a collection of logical statements characterising the extracted mutations, proteins and impacts. The classes and property predicates in these statements are defined in our Mutation Impact ontology [5] in OWL, based

on the earlier mutation ontology from [6]. Figure 1 (borrowed from [1]) shows the top level concepts of the ontology with some relations between them.

This semantic representation of text mining results already provides a great deal of flexibility – the results can be used with any toolsets that work with OWL. The most straightforward way of using semantically described data is by querying it directly, so we established an OWL database, using Sesame [7], that stores the results of mining different documents. Our users can query the populated OWL database via a SPARQL [8] endpoint [9]. Figure 2 shows an example of a query for proteins mutated with a specified impact.

As we anticipate a multitude of data reuse cases, the provision of a SPARQL endpoint as the sole data access form may not be sufficient. Consequently, we are looking for additional ways to provide access to the data. Our primary requirement is that the framework should support integration with other software and data for proteins, mutations, impacts and related biological entities, such as pathways, and drugs. This criterion is critical because the mutation impact mining results alone have limited reusability.

In this paper we review the SADI framework [10] as a candidate platform for providing interoperable access to our semantically exposed Mutation Impact data. The choice is based on the robust integrative features displayed by SADI services, discussed in the next section. This paper describes an exploratory case study using four biologically meaningful queries that require (i) some data from our Mutation Impact DB, as well as (ii) some biological knowledge from external sources. Furthermore, we test the queries using the SHARE client [11] which is designed to automatically discover and combine the required SADI services.

What is SADI?

The SADI framework [10] is a set of conventions for creating Semantic Web Services that can be *automatically discovered and orchestrated*. A SADI-compliant service consumes a whole RDF model as input and produces an RDF model as output. An input RDF model has some URI node designated as the central input node, and the whole input model is considered a description of the central node. Exactly the same node is always present in the output model as the central output node. The sole function of a SADI service is to annotate this node with new properties and assert these properties in the output model, in contrast with more conventional Web services that usually compute output without an explicit connection to the input.

The most important feature of SADI is that the predicates for these property assertions are fixed for each service. A declaration of these predicates, available online, constitutes a semantic description of the service. For example, if a service is declared with the predicate $myontology{:}isTargetOfDrug$ described in an ontology as a property linking proteins to drugs, the user knows that he can use the service to search for drugs targeting a given protein.

The declaration of the service predicates is done by specifying an OWL class for the output nodes. If this output class entails an existential restriction for some property, it means that the property is declared as produced by the service and the corresponding output data may be available from the service.

Another part of a service declaration is the input (OWL) class that imposes restrictions on the kind of input nodes the service can process. In particular, if this class subsumes an intersection of property restrictions, a well-behaved service will look for the corresponding properties attached to an input node, and use the values as parts of the input. As an example, consider the SADI service [12] computing the Body Mass Index. Its $InputClass$ is defined as the intersection of $\exists\ mged{:}has\_height.mged{:}Measurement$ ($mged$ abbreviates [13]) and $\exists\ mged{:}has\_mass.mged{:}Measurement$, so the service expects these two properties attached to an input node. The service's $OutputClass$ is a subclass of $\exists\ bmi{:}BMI.xsd{:}int$, so the service provides the predicate $bmi{:}BMI$ (bmi corresponds to the service's own ontology that describes the input and output classes). Given the following RDF as input

```
<http://www.freewebs.com/riazanov>
     mged:has_height <http://www.freewebs.com/riazanov/height> ;
     mged:has_mass <http://www.freewebs.com/riazanov/mass> .
<http://www.freewebs.com/riazanov/height>
     mged:has_value "1.7"^^xsd:float ;
     mged:has_units mged:m.
<http://www.freewebs.com/riazanov/mass>
     mged:has_value "85"^^xsd:float ;
     mged:has_units mged:kg .
```

the service generates this RDF as output:

```
<http://www.freewebs.com/riazanov>
     bmi:BMI "29.4"^^xsd:float .
```

The declaration of the input and output classes of a SADI service constitutes a *semantic description* of the service. Importantly, such semantic descriptions allow completely automatic discovery and composition of SADI services (see, e.g., [10, 11]). In practice, using SADI services to provide access to the Mutation Pipeline and DB will allow automatic integration with hundreds of external resources dealing with mutations, proteins and related biomedical entities, e.g., pathways and drugs, so long as they are registered with a SADI registry. These are desirable features of SADI motivating us to deploy our mutation impact services with this framework.

SADI services for Mutation Impact pipeline and data.

As an initial implementation with SADI, we created a service that takes a reference to a text, and outputs the property assertions derived from the input text, such as links from the text URI to the identified grounded mutations. Note that those grounded mutations also have links to ungrounded mutations, proteins and impacts. This SADI service can be

mostly useful in combination with services that find documents, as well as for users just wishing to use our pipeline remotely (with no installation effort). In fact, we currently use this service ourselves to populate the Mutation Impact DB with OWL ABox assertions, because it has the capability of converting the raw results of the Mutation Impact pipeline to OWL.

All other our SADI services essentially wrap some ad hoc queries to our Mutation Impact DB. For example, one of the most intensively used services – $getMutationByWildtypeProtein$ – finds all instances of the Mutation Impact ontology class $MutationSpecification$, given the UniProt ID of a protein that acts as the wildtype protein in those mutations. The class $MutationSpecification$ is central to the ontology and the DB: its instances represent grounded mentions of mutations and are linked to the corresponding wildtype and mutant proteins, the mutation impacts, and also the texts from which the mutation mentions were extracted. So, two other services also find $MutationSpecification$ instances by their mutant proteins and mutation impacts.

Two other services retrieve instances of biological entities of specified types, present in our DB. The service $getMIDBBioEntityByType$ does this for the top level biological entity classes in our ontology, such as $Protein$ or $PointMutation$. The service $getProteinPropertyByType$ specialises in protein property types, most of which are currently inherited from the Gene Ontology. Given a subclass of $ProteinProperty$, e.g., $GO\_0018786$ ('haloalkane dehalogenase activity') from the Gene Ontology, it finds all known properties of this type, grounded to specific proteins.

There are currently two auxilliary services: $getMutationImpactByProteinProperty$ finds mutation impact instances linked to a specified protein property grounded to a specific protein, and $getMutationSubseries$ finds series of elementary mutations identified in a text, that are subsets of a specified set of elementary mutations.

The list of all SADI services based on the Mutation Impact pipeline and DB, can be found in [14] and is also summarised in the following table:

service	operation
$mineTextForMutationImpacts$	extracts mutation specifications from a document
$getMutationByWildtypeProtein$	finds specifications of mutations grounded to a given protein
$getMutationByMutantProtein$	finds specifications of mutations resulting in a specified protein
$getMutationImpactByProteinProperty$	finds mutation impact instances affecting a specified grounded protein property
$getMutationByImpact$	finds mutation specifications corresponding to an impact on a specified grounded protein property
$getMutationSubseries$	finds mutation series instances that are subseries of a given mutation series
$getMIDBBioEntityByType$	finds biological entities by their type URIs
$getProteinPropertyByType$	finds protein properties grounded to specific proteins by their type URIs

All the services are also registered with the central SADI registry [15].

Use cases.

Here we introduce the use cases we have adopted to test the suitability of SADI as a medium for providing access to out Mutation Impact data. All our use cases are in the form of queries, i.e., the user is seeking some information from our Mutation Impact DB in combination with external resources.

Use case 1: Find all mutations and the structure images of wild type proteins that were mutated, where the impact of the mutation is an enhanced haloalkane dehalogenase activity. In this use case we aim to address the needs of a protein engineer who is seeking to understand what mutational changes can enhance the catalytic activity of an industrial enzyme, which is haloalkane dehalogenase in this scenario. The medium for reviewing the causal relationship of mutations on protein activity is a protein structure image which can be annotated with mutations and their impacts retrieved from a database/triplestore [16] or extracted automatically from documents using text mining techniques [4, 17]. In our use case, we perform retrieval of the specific protein structures where there are published reports of mutations having a positive impact on catalytic activity. The user would wish to retrieve and review these strucutres along with mutation locations and impact annotations. The expected output of the integrated SADI services is the selected protein structure files and the corresponding mutations.

Use case 2: Find all pathways, together with the corresponding pathway images, that might have been altered by a mutation of the protein Fibroblast growth factor receptor 3. In this scenario we address the needs of a systems biologist who is seeking to understand the likely impact of reported mutations on signalling or metabolic pathways [18] in which the mutated protein participates. This entails the retrieval of pathway information for the mutated proteins, which can be provided as a pathway diagram also. In the current use case we deal with mutations to the protein *Fibroblast growth factor receptor 3* reported in scientific papers which impact the protein either positively or negatively.

Use case 3: Find all drugs related to mutated proteins, together with their interaction partners, where the mutation impact is a decreased carbonic anhydrase activity. In this use case we address a query that a researcher in drug discovery would make when looking for existing drugs targeting a new disease condition. In the case of Carbonic anhydrase, an enzyme involved in the acid-base balance of blood (via the interconverion of carbon dioxide and bicarbonate), enzyme inhibitors such as acetazolamide cause mild metabolic acidosis. This can be beneficial to patients with severe chronic obstructive pulmonary disease (COPD) with chronic hypercapnic ventilatory failure who need a reduction in arterial carbon dioxide and a rise in arterial oxygen and the transport carbon dioxide out of tissues. The query will help us to identify the names of known drugs targeting the enzyme and what experimental modifications on the protein have resulted in lowering its activity *in situ*. Moreover, the query will also retrieve the names of proteins that interact with the enzyme directly through protein-protein interactions.

Use case 4: From the literature, find all reported mutations of the protein with the nsSNP rs2305178. In this use case, a researcher in genomics asks for all known mutations reported in the literature for a protein containing a non-synonymous SNP. Here the researcher is primarily looking for any literature describing impacts of a nsSNP on a protein. By retrieving all known mutations for the protein in which the nsSNP is reported, the researcher can find out if any of these reported mutations corresponds to the location of the SNP in question. Minimally the researcher can retrieve the full set of mutations to the protein based on reported experimental analysis and their impacts, together with references to the supporting literature.

Experiments with SHARE.

SHARE [11] is an experimental client featuring automatic discovery and orchestration of SADI services. From the user point of view, SHARE is a SPARQL engine that computes queries by picking and calling suitable SADI services from some registry. In a typical scenario, the user first looks up property predicates he needs for his query, in the list of predicates declared as provided by SADI services in a registry, and also related classes and properties in the referenced ontology. Then he uses the available concepts to form a regular SPARQL query, and sends it to a SHARE endpoint. Importantly, the SHARE engine decides itself which services have to be invoked and in what order, to execute the query. Note that the user deals only with an almost declarative query, i.e., he only needs to understand the semantics of the URIs being used in the query, although knowing the services providing the predicates can be beneficial. This situation suits our purposes well, so, for our experiments with SADI services for Mutation Impact data we are using the Web interface for SHARE [19].

Note that in the query examples below we omit prefix declarations – the information can be found in the corresponding URL references [20–27]. We also omit FROM clauses instructing SHARE to use our ontology for processing the queries, as well as FROM clauses importing RDF file [28] qualifying some individuals, e.g., $go{:}GO\_0018786$, as instances of the corresponding classes, e.g., $mioe{:}ProteinPropertyType$. Full versions of the queries are available from [29].

Our **use case 1** ("*Find all mutations and the structure images of wild type proteins that were mutated, where the impact of the mutation is an enhanced haloalkane dehalogenase activity*") can be realised with the following SPARQL query:

```
1 SELECT DISTINCT ?StructImage ?NormalizedMutation
2 WHERE {
3   ?Property mioe:proteinPropertyHasType go:GO_0018786 .
4   ?Impact mio:affectProperty ?Property .
5   ?Impact mio:hasDirection mio:Positive .
6   ?MutationSpec mio:specifiesImpact ?Impact .
7   ?MutationSpec mio:groundMutationsTo ?Protein .
8   ?MutationSeries mio:mutationSeriesIsSpecifiedBy ?MutationSpec .
9   ?MutationSeries mio:containsElementaryMutation ?Mutation .
10  ?Mutation mio:hasNormalizedForm ?NormalizedMutation .
11  ?Protein pred:has3DStructure ?Struct .
12  ?Struct obj:hasJmol3DStructureVisualization ?StructImage . }
```

Let us analyse how we construct this query. The predicate $mioe{:}proteinPropertyHasType$ in our ontology links grounded protein properties with their types, so we can use it to enumerate known instances of $GO\_0018786$. In lines 7-8, $mio{:}affectProperty$ links the grounded protein properties to the corresponding instances of mutation impacts and $mio{:}hasDirection$ selects only positive impacts. Using $mio{:}specifiesImpact$, we can select instances of mutation specifications (line 9), which in turn link to the corresponding wildtype proteins (line 10) and series of elementary mutations (line 11). We would like to see readable IDs of elementary mutations in the output, like D124N or V226A, so we use $mio{:}containsElementaryMutation$ to retrieve the corresponding elementary mutations and $mio{:}hasNormalizedForm$ to map them to the corresponding IDs.

So far we have used only predicates from our Mutation Impact ontology. Since the essense of the use case 1 is visualisation, we look for predicates in SADI-related ontologies, that could link proteins to their images. There is no direct link, but we can use the composition of $pred{:}has3DStructure$ and $obj{:}hasJmol3DStructureVisualization$ to first retrieve the PDB ID of a protein, and then find the corresponding graphics file.

SHARE was able to compute our query using three of our SADI services and two third party SADI services from the registry, providing $pred{:}has3DStructure$ and $obj{:}hasJmol3DStructureVisualization$. However, this was completely transparent to us as the end users. We only dealt with an almost completely declarative query composed of predicates we were able to find in ontologies. The only thing we need to know beyond the semantics of a predicate is the direction in which available services compute it: e.g., we cannot use $pred{:}has3DStructure$ to get from a PDB ID to the corresponding protein because there is currently no service that would annotate a PDB ID with the inverse of $pred{:}has3DStructure$. Finding the services, their invocation and some deduction with the ontological definitions of predicates, was done by SHARE completely automatically. Note especially the ease with which integrating our mutation-related information with the external sources of data was achieved.

The work required by **use case 2** (“*Find all pathways, together with the corresponding pathway images, that might have been altered by a mutation of the protein Fibroblast growth factor receptor 3*“) can also be divided into two

parts: the first part can be done using the predicates from our ontology, and the second part has to be delegated to external resources, dealing with genes, pathways and pathway visualisation. Since we know that the wildtype protein is *Fibroblast growth factor receptor 3* (UniProt ID P22607), we can easily retrieve the mutation specifications linked to this protein with the property $mio{:}groundMutationsTo$. These instances will have impacts attached to them with $mio{:}specifiesImpact$, and we can specify the interesting impact directions with $mio{:}hasDirection$. Using $pred{:}isEncodedBy$ we also map the protein to the corresponding gene, and $ont{:}isParticipantIn$ allows to retrieve the pathways in which the protein participates, $pred{:}visualizedByPathwayDiagram$ will fetch the corresponding graphics file URL. The resulting query is as follows:

```
SELECT DISTINCT ?Pathway ?PathwayDiagram
WHERE {
   ?MutationSpecification mio:groundMutationsTo uniprot:P22607 .
   ?MutationSpecification mio:specifiesImpact ?Impact .
   {?Impact mio:hasDirection mio:Positive}
     UNION {?Impact mio:hasDirection mio:Negative} .
   uniprot:P22607 pred:isEncodedBy ?Gene .
   ?Gene dmt:isParticipantIn ?Pathway .
   ?Pathway pred:visualizedByPathwayDiagram ?PathwayDiagram }
```

SHARE evaluated the query using our service that links proteins to mutations specifications, and two external SADI services providing $ont{:}isParticipantIn$ and $pred{:}visualizedByPathwayDiagram$, and found five pathways with diagrams.

Use case 3 ("*Find all drugs related to mutated proteins, together with their interaction partners, where the mutation impact is a decreased carbonic anhydrase activity*") is somewhat similar to use case 1: given the protein property type, we retrieve the grounded properties, positive impacts and the wildtype proteins with the help of some predicates from our ontology. The conection from the proteins to drug names is realised with the predicates $obj{:}isTargetOfDrug$ and $obj{:}hasDrugGenericName$. Separately, we find the interacting proteins with $pred{:}hasMolecularInteractionWith$. The resulting query is

```
SELECT ?DrugName ?InteractingProtein
WHERE {
   ?Property mioe:proteinPropertyHasType go:GO_0008270 .
   ?Impact mio:affectProperty ?Property .
   ?Impact mio:hasDirection mio:Negative .
   ?MutationSpecification mio:specifiesImpact ?Impact .
   ?MutationSpecification mio:groundMutationsTo ?Protein .
   ?Protein obj:isTargetOfDrug ?Drug .
   ?Drug obj:hasDrugGenericName ?DrugName .
   ?Protein pred:hasMolecularInteractionWith ?InteractingProtein }
```

Finally, the most difficult **use case 4** ("*From the literature find all reported mutations of the protein with the nsSNP rs2305178*") was implemented with the following query:

```
1  SELECT DISTINCT ?NormalizedMutation ?DocumentURL
2  WHERE {
3    dbsnp:rs2305178 obj:correspondsToEntrezGene ?EzGene .
4    ?Protein mioe:biologicalEntityHasType mio:Protein .
5    ?Protein pred:isEncodedBy ?KeggGene .
6    ?KeggGene obj:hasRefSeqTranscript ?RefSeq .
7    ?RefSeq obj:correspondsToEntrezGene ?EzGene .
8    ?MutationSpecification mio:groundMutationsTo ?Protein .
9    ?MutationSeries mio:mutationSeriesIsSpecifiedBy ?MutationSpecification .
10   ?MutationSeries mio:containsElementaryMutation ?Mutation .
11   ?Mutation mio:hasNormalizedForm ?NormalizedMutation .
12   ?Document foaf:topic ?MutationSpecification .
13   ?Document rss:link ?DocumentURL }
```

The property $obj{:}correspondsToEntrezGene$ (line 3) maps the specified dbSNP ID to an Entrez gene ID. If we were dealing with completely declarative queries, it would be enough to use a composition of the predicates $obj{:}correspondsToEntrezGene$, $obj{:}hasRefSeqTranscript$ and $pred{:}isEncodedBy$, as in lines 5-7, to map the Entrez gene ID to a protein. However, no SADI service currently provides the inverses to the first two predicates, so the composition can only work in the direction from proteins to Entrez gene IDs. To do so, we had to implement a service that enumerates all proteins known in our DB. In fact, it is more general – it enumerates instances of several main biological entity classes from our ontology, such as $MutationImpact$ or $PointMutation$. The service provides the inverse of $mioe{:}biologicalEntityHasType$ whose use is demonstrated in line 4. Linking the protein to elementary mutations is done exactly the same way as in use case 1. Finally, the last two lines in the query serve to retrieve the URLs of the documents from which the corresponding mutation specifications were extracted.

Conclusions and future work.

The primary goal of our case study was to explore the suitability of the SADI framework as a medium to faciliate data sharing and integration across biological data types. We have identified that SADI provides an effective way of exposing our mutation impact data such that it can be leveraged by a variety of stakeholders in multiple use cases. Our experience in deploying and regisrering mutation services in accordance with SADI specifications was positive, albeit with some challenges. Specifically, we identified that advanced skills in knowledge engineering were required to build semantic representations of the services. In addition, we note that formulating the queries based on the SADI services still requires extensive search for predicates in the SADI-related ontologies. Clearly, the necessary infrastructure for such search is yet to be built. We also did not yet fully explore the current capabilites of other SADI clients, such as the plugins for Taverna [30] and Sentient Knowledge Explorer (see, e.g., [11]) for our use cases.
In future work we aim to extend the Mutation Impact DB with more data types related to mutation annotations extracted from the literature, and create the corresponding SADI services facilitating integration with other

Bioinformatics data.

Acknowledgements.

This research was funded in part by the New Brunswick Innovation Foundation, New Brunswick, Canada; NSERC, Discovery Grant Program, Canada; and the CANARIE NEP-2 Program (C-BRASS project). We also thank Luke McCarthy for helping us with various SADI-related technical issues.

References

1. Laurilla J, Naderi N, Witte R, Riazanov A, Kouznetsov A, Baker CJO: **Algorithms and semantic infrastructure for mutation impact extraction and grounding**. In *ICOB2010* 2010. [To appear.].
2. Cunningham H, Maynard D, Bontcheva K, Tablan V: **GATE: A Framework And Graphical Development Environment For Robust NLP Tools And Applications**. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)* 2002.
3. Lau EY, Kahn K, Bash P, Bruice T: **The importance of reactant positioning in enzyme catalysis: a hybrid quantum mechanics/molecular mechanics study of a haloalkane dehalogenase**. *Proc. Natl. Acad. Sci. USA* 2000, **97**(18):9937–42.
4. Rajaraman K, Choo KH, Ranganathan S, Baker CJO: **A Workflow for Mutation Extraction and Structure Annotation**. *J. Bioinformatics and Computational Biology* 2007, **5**(6):1319–1337.
5. **Mutation Impact Ontology.** [http://unbsj.biordf.net/ontologies/mutation-impact-ontology.owl].
6. Witte R, Kappler T, Baker CJO: **Enhanced semantic access to the protein engineering literature using ontologies populated by text mining**. *Int J Bioinform Res Appl* 2007, **3**(3).
7. Broekstra J, Kampman A, van Harmelen F: **Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema**. In *The Semantic Web ISWC 2002* 2002:54–68.
8. **SPARQL Query Language for RDF, W3C Recommendation 15 January 2008.** [http://www.w3.org/TR/rdf-sparql-query/].
9. **Mutation Impact RDF triplestore SPARQL endpoint.** [http://unbsj.biordf.net/openrdf-workbench/repositories/mutation-impact-db/query].
10. Wilkinson MD, Vandervalk BP, McCarthy EL: **SADI Semantic Web Services – 'cause you can't always GET what you want!** In *APSCC* 2009:13–18.
11. Vandervalk BP, McCarthy EL, Wilkinson M: **SHARE: A Semantic Web Query Engine for Bioinformatics.** In *The Semantic Web (ISWC 2009)* 2009:367–369.
12. **SADI service computing the body mass index**. *http://sadiframework.org/examples/calculateBMI*.
13. **MGED ontology prefix, abbreviated as** $mged$. *http://mged.sourceforge.net/ontologies/MGEDOntology.owl*.
14. **SADI services based on the Mutation Impact pipeline and DB**. *http://unbsj.biordf.net/mutation-impact*.
15. **Central SADI registry.** [http://sadiframework.org/registry/services/].
16. Gabdoulline RR, Ulbrich S, Richter S, Wade RC: **ProSAT2Protein Structure Annotation Server** 2006.
17. Baker CJO, Witte R: **Mutation Mining-A Prospector's Tale**. *Information Systems Frontiers* 2006, **8**:47–57.
18. Bauer-Mehren A, Furlong LI, Rautschka M, Sanz F: **From SNPs to pathways: integration of functional effect of sequence variations on models of cell signalling pathways**. *BMC Bioinformatics* 2009, **10**(S-8):6.
19. **Web interface for SHARE.** [http://biordf.net/cardioSHARE/].
20. **Our Mutation Impact ontology core URL prefix, abbreviated as** mio. *http://www.freewebs.com/riazanov/mutationOntology2010.owl\#*.
21. **Our Mutation Impact ontology extension URL prefix, abbreviated as** $mioe$. *http://www.freewebs.com/riazanov/mutationOntology2010_extras.owl\#*.

22. **Gene Ontology prefix, abbreviated as** go. *http://purl.org/obo/owl/GO\#*.
23. **SADI service object ontology prefix, abbreviated as** obj. *http://sadiframework.org/ontologies/service_objects.owl\#*.
24. **SADI predicates ontology prefix, abbreviated as** $pred$. *http://sadiframework.org/ontologies/predicates.owl\#*.
25. **Dumontier Lab ontology prefix, abbreviated as** dmt. *http://ontology.dumontierlab.com/*.
26. **BioRDF UniProt nomenclature prefix, abbreviated as** $uniprot$. *http://biordf.net/moby/UniProt/*.
27. **dbSNP nomenclature prefix, abbreviated as** $dbsnp$. *http://lsrn.org/dbSNP:*.
28. **RDF file containing descriptions for seed values in our queries**. *http://dl.dropbox.com/u/2483134/input.rdf*.
29. **Full versions of the SPARQL queries presented in this paper.** [http://unbsj.biordf.net/mutation-impact/aimm2010_queries.html].
30. Withers D, Kawas E, McCarthy L, Vandervalk B, Wilkinson M: **Workflow Construction in Taverna: The SADI and BioMoby Plug-ins**. In *ISoLA 2010 (to appear)*.

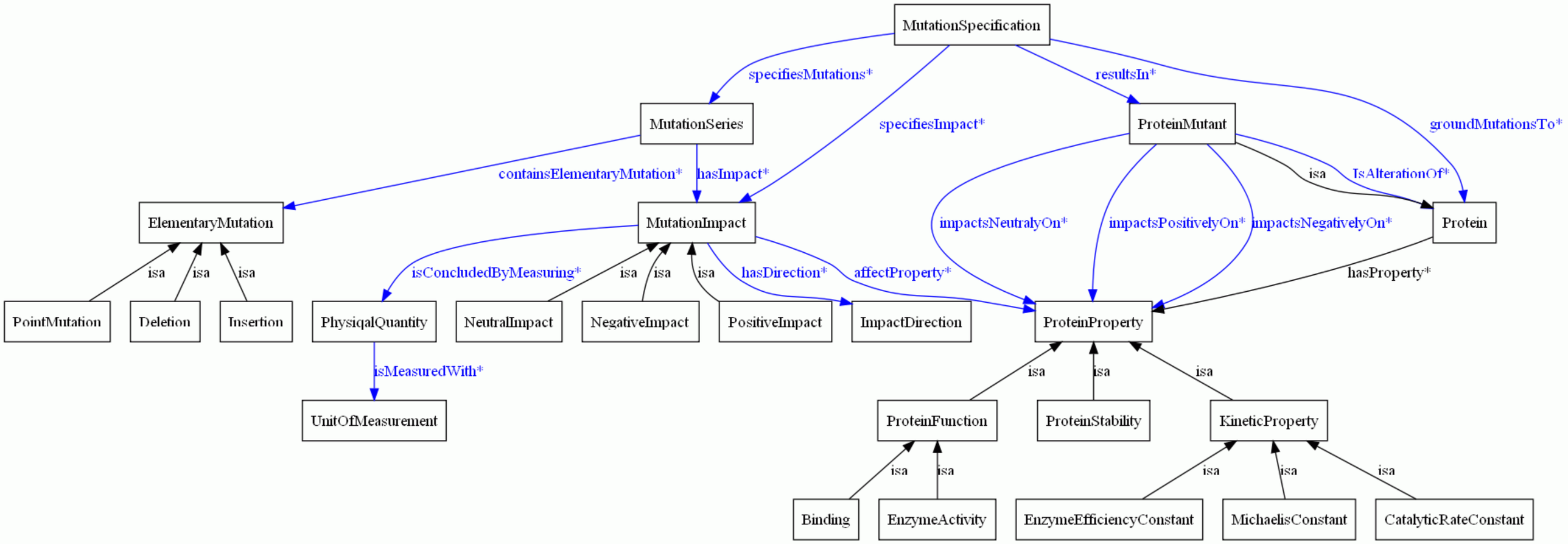
Figures

Figure 1 : Mutation impact ontology structure.

Visualization of top level concepts as *Mutation Specification*, *Protein*, *Mutation Impact* and *Protein Property* being connected through object properties.

Figure 2 : Sample SPARQL query to our Mutation Impact DB

A SPARQL query expressing the natural language question "*Which proteins have been mutated so that there is a negative impact on haloalkane dehalogenase activity and what is the sequences of the corresponding mutants?*" is shown to the left. The first four answers (result rows) are displayed to the right.


MutationSpecification
specifiesMutations*
specifiesImpact*
resultsIn*
groundMutationsTo*
MutationSeries
ProteinMutant
containsElementaryMutation*
hasImpact*
isa
IsAlterationOf*
ElementaryMutation
MutationImpact
Protein
impactsNeutralyOn*
impactsPositivelyOn*
impactsNegativelyOn*
isa
isa
isa
isConcludedByMeasuring*
isa
isa
isa
hasDirection*
affectProperty*
hasProperty*
PointMutation
Deletion
Insertion
PhysiqalQuantity
NeutralImpact
NegativeImpact
PositiveImpact
ImpactDirection
ProteinProperty
isMeasuredWith*
isa
isa
isa
UnitOfMeasurement
ProteinFunction
ProteinStability
KineticProperty
isa
isa
isa
isa
isa
isa
Binding
EnzymeActivity
EnzymeEfficiencyConstant
MichaelisConstant
CatalyticRateConstant

Query Language: SPARQL

Query:

```
PREFIX MIO:<http://www.unbsj.ca/sase/csas/ontologies/MIO.owl#>
PREFIX GO:<http://purl.org/obo/owl/GO#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?Protein ?MutantSequence
WHERE {
 ?MutationImpact MIO:hasDirection MIO:Negative .
 ?Property rdf:type GO:GO_0018786 .
 ?MutationSpecification MIO:resultsIn ?ProteinMutant .
 ?MutationSpecification MIO:specifiesImpact ?MutationImpact .
 ?MutationSpecification MIO:groundMutationsTo ?Protein .
 ?ProteinMutant MIO:hasSequence ?MutantSequence .
 ?MutationImpact MIO:affectProperty ?Property . }
```

Protein	MutantSequence
<http://biordf.net/moby/UniProt/P51698>	"MSLGAKPFGEKK
<http://biordf.net/moby/UniProt/P51698>	"MSLGAKPFGEKK
<http://biordf.net/moby/UniProt/P51698>	"MSLGAKPFGEKK
<http://biordf.net/moby/UniProt/P22643>	"MINAIRTPDQRF