SHARE: A Semantic Web Query Engine for Bioinformatics

Ben P Vandervalk, E Luke McCarthy, and Mark D Wilkinson

The Providence Heart + Lung Research Institute at St. Paul's Hospital,
University of British Columbia, Department of Medical Genetics,
Vancouver, BC, Canada
ben.vvalk@gmail.com

Abstract. Driven by the goal of automating data analyses in the field of bioinformatics, SHARE (Semantic Health and Research Environment) is a specialized SPARQL engine that resolves queries against Web Services and SPARQL endpoints. Developed in conjunction with SHARE, SADI (Semantic Automated Discovery and Integration) is a standard for native-RDF services that facilitates the automated assembly of services into workflows, thereby eliminating the need for ad hoc scripting in the construction of a bioinformatics analysis pipeline.

The SHARE Query System

The task of coordinating datasets and software packages from different research laboratories is a recurring practical problem in bioinformatics [1]. While the size, quantity, and diversity of bioinformatics resources continues to grow rapidly, there are still no widely adopted standards for encoding data or designing software interfaces. As a result, the orchestration of bioinformatics analyses typically requires a large time investment for studying database schemas and for implementing ad hoc scripts that connect incompatible data formats and software.

To address this problem, the authors have developed the SADI service standard and the SHARE query engine. SADI is a straightforward set of recommendations for exposing bioinformatics resources as RDF-native Web Services, while SHARE is a specialized SPARQL query engine that enables the flexible rewiring of SADI services into complex workflows. In more detail, the two data sources queried by SHARE are:

SADI services SADI [2] services are stateless, native-RDF Web Services that are invoked by means of an HTTP POST request. Both the request and the response are single RDF documents, and the input/output URIs within these documents are instances of a provider-specified input/output OWL class. The principal rule regarding input/output OWL instances is that they must be rooted at the same URI (i.e. the input URI). It follows from this rule that a service must report its results by attaching new predicates to the input URI, and thus *the precise relationship between the input data and the output data is explicitly stated.* The SADI service registry identifies the predicates that a service attaches to its

input URIs by comparison of its input and output OWL classes, and services may be retrieved from the registry using these predicates as a key.

SPARQL Endpoints Any standard SPARQL endpoint may be registered, indexed and queried by the SHARE query engine. As in the DARQ [3] system, SPARQL endpoints are indexed by the set of predicates that occur within their datasets.

The syntax of a SHARE query is identical to that of a standard SPARQL query. SHARE answers a SPARQL query by successively resolving each triple pattern in the query against a matching set of SADI services/SPARQL endpoints. A service/endpoint matches a triple pattern with predicate p if that service/endpoint attaches p (or the OWL inverse of p) to its input. The output triples generated by services/endpoints are accumulated in a local triple store, and the subject/object values of output triples are recorded as bindings (values) for any unbound variables in the current pattern. Bindings are carried forward into subsequent triple patterns, and thus the resolution of a particular pattern may require multiple invocations of the same service/endpoint. After all triple patterns in the query have been resolved, the solutions are displayed.

The SHARE demonstration site [4] allows users to issue SPARQL queries against a sample set of SADI services and SPARQL endpoints. By accessing several well known databases in bioinformatics, the provided services generate predicates that connect proteins (UniProt), genes (KEGG/Entrez), molecular structures (PDB), biological pathways (KEGG), and several other types of biological entities. The relationships between the service-generated predicates (and thus the services themselves) are depicted schematically at [5]. Fig. 1 shows an example query which asks: "What motifs are present in enzymes of the human caffeine metabolism pathway?". (Motifs are contiguous, evolutionarily-conserved subsequences of proteins that often have a known chemical function.)

```
PREFIX pred: <http://sadiframework.org/ontologies/predicates.owl#>
PREFIX kegg: <http://biordf.net/moby/KEGG_PATHWAY/>
SELECT ?gene ?protein ?motif
WHERE {
    kegg:hsa00232 pred:hasPathwayGene ?gene .
    ?gene pred:encodes ?protein .
    ?protein pred:hasMotif ?motif .
}
```

Fig. 1. A sample SHARE query which asks: "What motifs are present in enzymes of the human caffeine metabolism pathway?".

SHARE answers the example query in Figure 1 by resolving each of the triple patterns in the WHERE clause sequentially. In the first step, SHARE identifies the set of services/endpoints that attach the predicate **pred:hasPathwayGene** to their input URIs. In this case, the SADI registry contains one such service

called *getKEGGGenesByPathway*. The URI for the human caffeine metabolism pathway (**kegg:hsa00232**) is sent as input to *getKEGGGenesByPathway*, and in response, the service generates a set of triples with **kegg:hsa00232** as the subject, **pred:hasPathwayGene** as the predicate, and KEGG genes as the objects. The object URIs (genes) of these triples then become bindings for the variable **?gene**. In the second step, SHARE identifies services/endpoints that attach the predicate **pred:encodes** to their input URIs, and finds a service called *KeggToUniProt*. Each binding of **?gene** is sent as an input to this service, and in response, *KeggToUniProt* returns a set of triples that connect genes to proteins via the predicate **pred:encodes**. In the third and final step, the query engine resolves **pred:hasMotif** to a service called *FindMotifById*, in order to connect the bindings of **?protein** to the URIs of corresponding motifs.

While the first two services invoked by the sample query perform simple database retrievals, the third service (*FindMotifById*) invokes a search algorithm for discovering known patterns in sequence data. This demonstrates that SHARE is suitable for coordinating both data retrieval and arbitrary software analysis within a single query, which is a highly desirable charactertistic for our bioinformatics user base.

The main innovation of the SHARE system is its ability to resolve SPARQL queries across Web Services in a transparent manner. In this sense, it can be viewed as an extension of the ideas underlying the SemWIQ [6] and DARQ [3] projects that enable querying over SPARQL endpoints. The mapping from SPARQL queries to Web Services is facilitated by the SADI service standard, which allows services to be discovered via the relationships that connect their inputs and outputs. In addition, SADI offers a service model that is straightforward in comparison to existing Semantic Web Service standards such as OWL-S [7] and WSMO [8], provided only stateless and non-world-altering services are required.

References

1. Stein, L.: Creating a Bioinformatics Nation. Nature, 417:119-120 (2002)
2. The SADI Web Service Framework, http://sadiframework.org
3. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. The Semantic Web: Research and Applications, Springer, Berlin/Heidelberg, 524-538 (2008)
4. SHARE demonstration, http://biordf.net/cardioSHARE/query
5. SHARE predicates page, http://biordf.net/cardioSHARE/predicates.html
6. Langegger, A., Wöß, W., Blöchl, M.: A Semantic Web Middleware for Virtual Data Integration on the Web. Semantic Web: Research and Applications, Springer, Berlin/Heidelberg, 493-507 (2008)
7. Martin, D., Paolucci, M., McIlraith, S., et al.: Bringing Semantics to Web Services: The OWL-S Approach, Semantic Web Services and Web Process Composition, Springer, Berlin/Heidelberg, 26-42 (2004)
8. Roman, D., Keller, U., Lausen, H., et al.: Web Service Modeling Ontology, Applied Ontology, 1(1):77-106 (2005)