Diamond Debugger Demo: Rete-Based Processing of Linked Data

Daniel P. Miranker, Rodolfo K. Depena, Hyunjoon Jung, Juan F. Sequeda, and Carlos Reyna

Department of Computer Science
University of Texas at Austin
{miranker, jsequeda}@cs.utexas.edu
{rudy.depena,polaris79, creynam89}@gmail.com

Abstract. Diamond is a Rete match based system that evaluates SPARQL queries on Linked Data. The evaluation of SPARQL query predicates is a useful intermediate milestone for a system ultimately intended to support full rule-based inference on Linked Data. A byproduct is the integrated graphical rule debugging environment is a first of its kind debug environment for SPARQL queries.

Keywords: Rete, SPARQL, Linked Data, Semantic Web

1 Background

The Linked Data model is an emerging component of the Semantic Web. The base layer of the Semantic Web is a representation of a directed labeled graph, expressed using resource description framework (RDF). Each edge of such a graph is commonly known as a triple. A triple is composed of a subject, a predicate and an object. The predicate is the edge label. The subject and object are vertex labels. Each constituent may be a URI. The intention is that labels form global unique ids and overlay DNS services to identify a particular server that may provide additional details (semantics) for the URI in the form of additional triples. The object of a triple may contain a literal. Thus, an RDF graph can represent complex data, spanning an arbitrary set of Internet servers [6].

Formal semantics for the Linked Data model are still emerging [4, 5]. The base principles mimic the behavior of hyperlinks in html documents [6]. That is, like a URL, dereferencing a URI instigates a response from a particular server. However, in lieu of an HTML document that may contain both text and an embedded set of URL-based hyperlinks, the server simply returns a set of triples.

One can expect that Linked Data crawlers will be intelligent. To date, all Linked Data specifications and most related work is limited to RDF. There is no explicit connection to the schema, ontology and rule layers, (RDFS, OWL, RIF), of the Semantic Web technology stack. Thus, in Linked Data, any semantic entailment will necessarily be implemented by inference processes associated with the processes that initiate and control the Linked Data crawlers. For example, SPARQL 1.1 allows triples to be updated. SPARQL 1.1 also inlucdes entailment rules that define closure over subclass hierarchies. Unless a system admits to limiting query and inferance to a potentialy inconsistent cache, it is necessarily the case that inferance entail freshly collected data.

Architecturally, the intrinsic, incremental behavior of the Rete Match aligns well with the web crawling aspects of the Linked Data model. This is true if one is evaluating rule predicates, or just a single predicate. When a Linked Data URI is dereferenced it returns a set of triples. The values of the triples may include additional URIs that have not yet been dereferenced. This operational behavior is identical to the algorithmic behavior of the Rete match per its original context, the incremental evaluation of changes to working memory in forward-chaining rule systems. Since an RDF graph is arbitrarily large and dynamic, even if an implementation references a local cache of prefetched triples, one can anticipate that any formal semantics will have to be consistent with an evaluation method that, operationally, crawls the web of Linked Data and reports results prior to reaching all reachable vertices. I.e. crawling can paused at any time, and the system evaluated.

Motivated, in part, by the anticipation intelligent Linked Data agents, we have first built a SPARQL query engine based on the Rete match and architected the integration of a Rete-based system with link-crawling and caching components [3]. In Diamond, there is a Rete network object. Each rule predicate is compiled as an instance of the Rete network object[1].

Serendipitously our implementation of a graphical rule debugger is also a SPARQL query debugger. Those already familiar with graphical debugging environments for Rete-based inference engines will already be familiar with the operation and concomitant rendering of the Rete network and its content. This is not the case for most developers in the SPARQL community.

We anticipate the development of a SPARQL query debugger will, further, be welcomed by that community as SPARQL queries can be expansive, even larger than comparable SQL queries. To support this claim, a query from the Berlin SPARQL Benchmark Suite is reproduced in Figure 1. This benchmark is distinguished as it provides semantically equivalent queries in SQL, as shown in Figure 2. The definition of a set of Rete operators for SPARQL follows from long standing connections made between relational algebra and rule predicates, and results that prove an expressive equivalence between SPARQL, DatalogNeg without recursion and relational algebra. [1, 7]

2 The System

The Diamond architecture is illustrated in Figure 3. The Rete network is created, dynamically, from a runtime library of Rete netword object definitions [7]. The URI dereferancing object is static. A critical design component is the pair of

[1] Optimizations based on sharing Rete network subtrees is anticipated by more sophisticated compilation techniques and providing for the composition of Rete network object instances.

```
PREFIX bsbm-inst: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/>
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?label ?comment ?producer ?productFeature
    ?propertyTextual1 ?propertyTextual2 ?propertyTextual3
    ?propertyNumeric1 ?propertyNumeric2
    ?propertyTextual4 ?propertyTextual5 ?propertyNumeric4
WHERE
    bsbm-inst:ProductXYZ rdfs:label ?label .
    bsbm-inst:ProductXYZ rdfs:comment ?comment .
    bsbm-inst:ProductXYZ bsbm:producer ?p .
    ?p rdfs:label ?producer .
    bsbm-inst:ProductXYZ dc:publisher ?p .
    bsbm-inst:ProductXYZ bsbm:productFeature ?f .
    ?f rdfs:label ?productFeature .
    bsbm-inst:ProductXYZ bsbm:productPropertyTextual1 ?propertyTextual1 .
    bsbm-inst:ProductXYZ bsbm:productPropertyTextual2 ?propertyTextual2 .
    bsbm-inst:ProductXYZ bsbm:productPropertyTextual3 ?propertyTextual3 .
    bsbm-inst:ProductXYZ bsbm:productPropertyNumeric1 ?propertyNumeric1 .
    bsbm-inst:ProductXYZ bsbm:productPropertyNumeric2 ?propertyNumeric2 .
    OPTIONAL  bsbm-inst:ProductXYZ bsbm:productPropertyTextual4
                ?propertyTextual4
    OPTIONAL  bsbm-inst:ProductXYZ bsbm:productPropertyTextual5
                ?propertyTextual5
    OPTIONAL  bsbm-inst:ProductXYZ bsbm:productPropertyNumeric4
                ?propertyNumeric4
```

Fig. 1. BSBM SPARQL Query 2

```
SELECT pt.label, pt.comment, pt.producer, productFeature, propertyTex1,
    propertyTex2, propertyTex3, propertyNum1, propertyNum2,
    propertyTex4, propertyTex5, propertyNum4
FROM product pt, producer pr, productfeatureproduct pfp
WHERE pt.nr=XYZ AND pt.nr=pfp.product AND pt.producer=pr.nr
```

Fig. 2. BSBM SQL Query 2

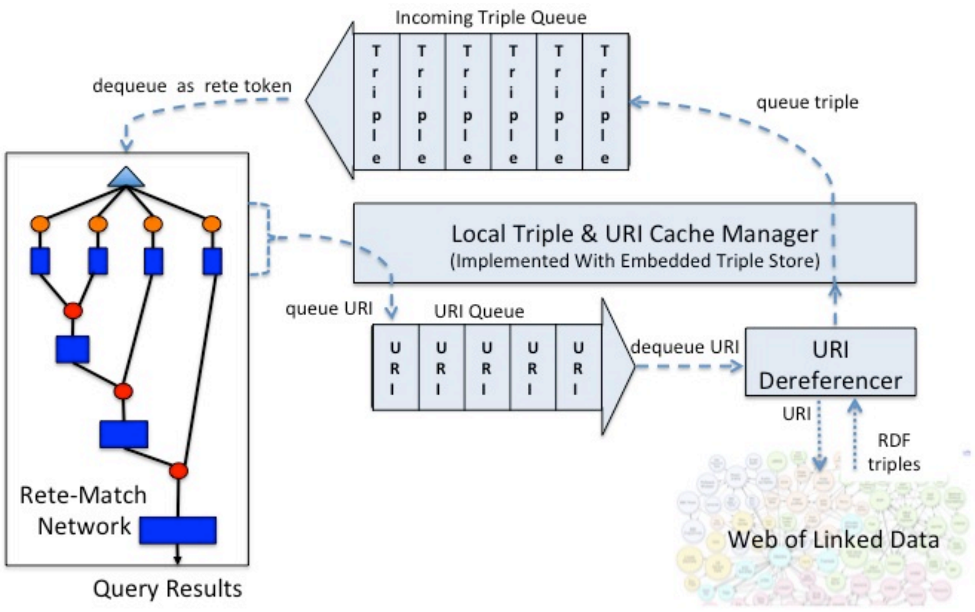


Fig. 3. Diamond Architecture Diagram

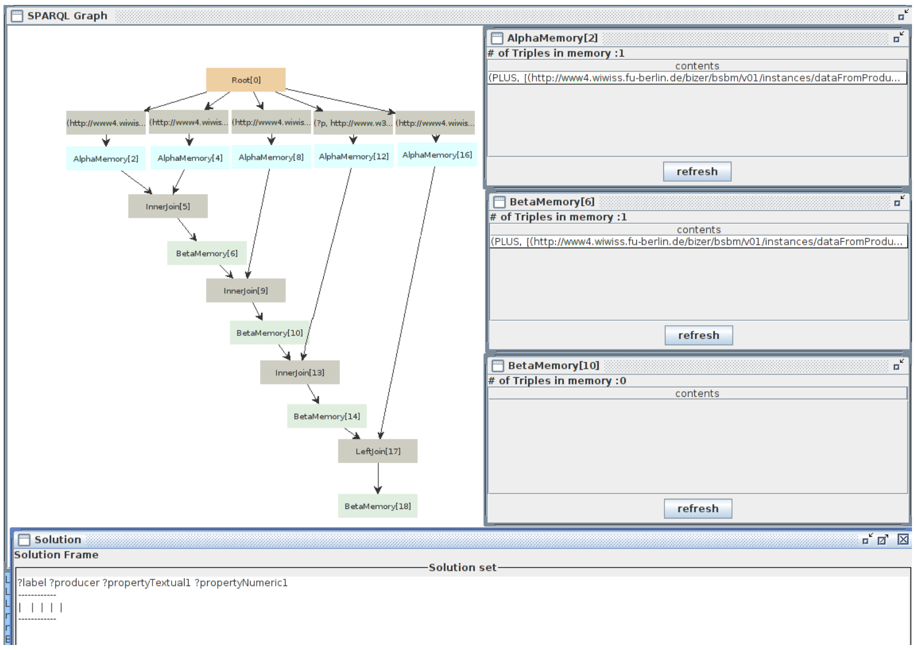
queues. The queues are intended to enable parallel, asynchronous execution of query evaluation and URI dereferencing. The queues are actively managed to avoid redundant dereferencing of URIs or redundant processing of triples by the Rete network. The queue manager is implemented by accumulating triple in an embedded copy of the Sesame [2] triplestore.

3 Demonstration

For demo purposes we use the benchmark query illustrated in Figure 1. For pedagogical purposes the screen shots are created by choosing an illustrative subset of 5 triple patterns. Video can be found at http://ribs.csres.utexas.edu/diamond/. Figure 4 is a screen shot of the debugger when there are triples that satisfy 4 of the 5 triple patterns. No triples that satisfy the third triple pattern.

Given the potential for a large number of both triples (data) and triple patterns (SPARQL clauses) the Rete network is rendered seperately from the data and the contents of the memory nodes. The contents of a memory node is viewed by clicking on the node in the Rete network, which opens a new window. Users may open, resize and move such windows anywhere on the screen. To save space, the figure shows three memory node windows overlayed on the window of the Rete network. We show, contents of one alpha-memory, that there is a repre-

[2] http://www.openrdf.org/



AlphaMemory[2]
# of Triples in memory :1
contents
(PLUS, [(http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProdu...

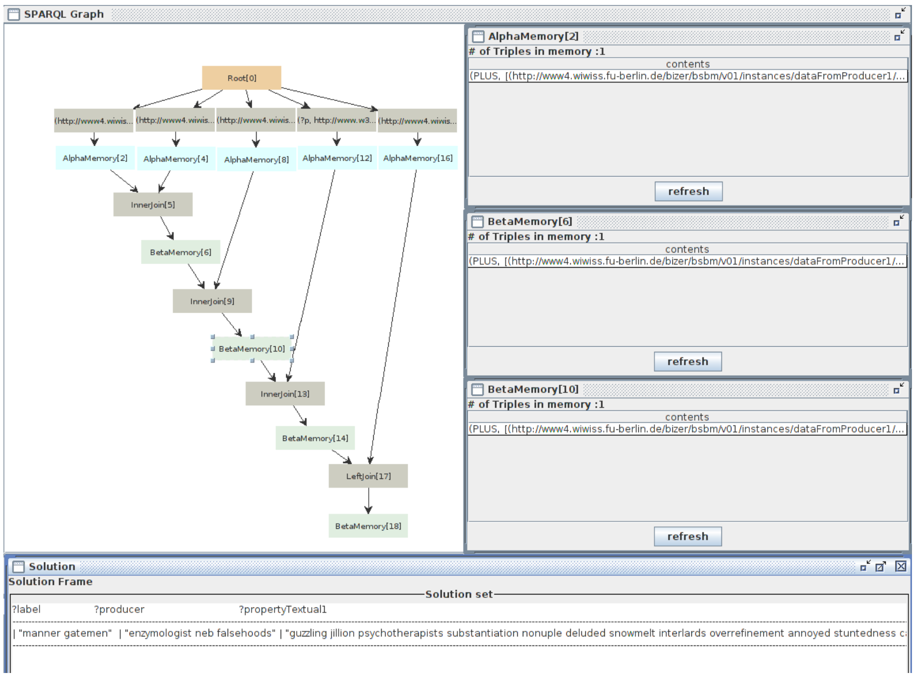
BetaMemory[6]
# of Triples in memory :1
contents
(PLUS, [(http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProdu...

BetaMemory[10]
# of Triples in memory :0
contents

Solution Frame
Solution set

```
?label ?producer ?propertyTextual1 ?propertyNumeric1
------------
| | | | |
------------
```

Fig. 4. Screenshot of before



AlphaMemory[2]
# of Triples in memory :1
contents
(PLUS, [(http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer1/...

BetaMemory[6]
# of Triples in memory :1
contents
(PLUS, [(http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer1/...

BetaMemory[10]
# of Triples in memory :1
contents
(PLUS, [(http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer1/...

Solution Frame
Solution set

```
?label             ?producer                      ?propertyTextual1
------------------------------------------------------------------------------------------------------------------------------------------------------
| "manner gatemen"  | "enzymologist neb falsehoods" | "guzzling jillion psychotherapists substantiation nonuple deluded snowmelt interlards overrefinement annoyed stuntedness c
------------------------------------------------------------------------------------------------------------------------------------------------------
```

Fig. 5. Screenshot of after

sentation of a successful join in the first beta-memory. The remainder of the network is empty.

Upon a the arrival of a triple that satisfies the third triple pattern the query becomes satisfied. Figure 5 shows the subsequent state of the Rete network. Each of the beta memories now has content.

4 Discussion

Although Diamond currently treats a SPARQL query as the predicate of a single rule, the system is easily extended to process a set of forward-chaining rules. The overlap of operator level equivalence between SPARQL query predicate evaluation and rule system evaluation is well established in the literature [1, 2, 8, 9]. The extensibility of the implementation is a byproduct of object-oriented design principles. Although we have no immediate plans per the investigation of parallel evaluation of rule systems, we note that the coordination of Rete network evaluation with the Linked Data crawlers is by means of asynchronous queues. Thus, the mechanisms for asynchronous concurrent rule evaluation are in place as well.

Acknowledgments. This research is supported by an NSF grant IIS-1018554. Juan F. Sequeda was supported by an NSF Graduate Research Fellowship.

References

1. Renzo Angles and Claudio Gutierrez, ‘The expressive power of sparql’, in *International Semantic Web Conference*, pp. 114–129, (2008).
2. François Bry, Tim Furche, Bruno Marnette, Clemens Ley, Benedikt Linse, and Olga Poppe, ‘Sparqlog: Sparql with rules and quantification’, in *Semantic Web Information Management*, 341–370, (2009).
3. Charles Forgy, ‘Rete: A fast algorithm for the many patterns/many objects match problem’, *Artif. Intell.*, **19**(1), 17–37, (1982).
4. Olaf Hartig, ‘Sparql for a web of linked data: Semantics and computability’, in *ESWC*, pp. 8–23, (2012).
5. Olaf Hartig, Christian Bizer, and Johann Christoph Freytag, ‘Executing sparql queries over the web of linked data’, in *Proceedings of the 8th International Semantic Web Conference*, pp. 293–309, (2009).
6. Tom Heath and Christian Bizer, *Linked Data: Evolving the Web into a Global Data Space*, Synthesis Lectures on the Semantic Web, Morgan & Claypool Pub., 2011.
7. Daniel P. Miranker, Rodolfo K. Depena, Hyunjoon Jung, Juan F. Sequeda, and Carlos Reyna, ‘Diamond: A sparql query engine, for linked data based on the rete match’, in *Proc. of the Artificial Intelligence meets the Web of Data Workshop at ECAI12*, (2012).
8. Axel Polleres, ‘From sparql to rules (and back)’, in *Proc. of the 16th int. conf. on World Wide Web*, WWW ’07, pp. 787–796, New York, NY, (2007). ACM.
9. Simon Schenk and Steffen Staab, ‘Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web’, in *Proc. of the 17th int. conf. on World Wide Web*, WWW ’08, pp. 585–594, New York, (2008). ACM.