Vivisecting LUBM

Paolo Castagna, Chris Dollin, Andy Seaborne

HP Laboratories
HPL-2009-348

Keyword(s):
lubm, inference, rdf, parallel processing

Abstract:

This report describes an implementation of a parallel inference engine for LUBM. Starting from the Univ-Bench ontology, we derive a specialized ruleset which captures its semantics. This ruleset can be simplified using simple transformations to remove data dependencies between rules. Finally, a parallel implementation of a custom LUBM reasoner, which performs inference in a streaming fashion using a small cluster, is described. Only preliminary results are presented, more experiments are necessary and an investigation on how to apply the same approach to OWL ter-Horst and OWL 2 RL profile is suggested.

External Posting Date: November 6, 2009 [Fulltext] Approved for External Publication
Internal Posting Date: November 6, 2009 [Fulltext]

Vivisecting LUBM

Paolo Castagna, Chris Dollin, Andy Seaborne

Hewlett-Packard Laboratories, Long Down Avenue, Stoke Gifford, Bristol BS34 8QZ, UK

Abstract

This report describes an implementation of a parallel inference engine for LUBM. Starting from the Univ-Bench ontology, we derive a specialized ruleset which captures its semantics. This ruleset can be simplified using simple transformations to remove data dependencies between rules. Finally, a parallel implementation of a custom LUBM reasoner, which performs inference in a streaming fashion using a small cluster, is described. Only preliminary results are presented, more experiments are necessary and an investigation on how to apply the same approach to OWL ter-Horst and OWL 2 RL profile is suggested.

Keywords: lubm, inference, rdf, parallel processing

1 Introduction

In terms of scalability, inference is one of the most challenging types of RDF processing. Issues arise from the size and the complexity of ontologies as well as from the amount of data to process.

Often inference, or some sort of data cleansing similar to inference, needs to be performed over large datasets at ingestion time when no indexes over the data are yet available. The complexity of inference varies with the expressivity of the language used to describe a vocabulary or ontology (RDF Schema, OWL Lite, DL or Full, OWL ter-Horst [12], OWL 2 RL[1], etc.) as well as with the specific ontology, which might use only a limited set of constructs from an ontology language.

This report focuses on a scenario which involves relatively simple ontologies; however, the amount of instance data to process is considerably large. The Lehigh University Benchmark (LUBM) [1][2][3], although limited [4][5], has been chosen because it is one of the oldest and, probably because of that, one of the most used benchmarks to compare inference engines. LUBM specifies a fairly simple OWL Lite ontology, called Univ-Bench[2], which describes the university domain. The datasets, which are random but repeatable, are synthetically generated and the benchmark describes fourteen test queries, most of which require inference, and their expected

[1] http://www.w3.org/TR/owl2-profiles/#OWL_2_RL
[2] http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl

results. Benchmarks[3] can be a valid tool to compare performance of RDF systems. However, a careful examination of a benchmark is necessary in order to understand its value, usefulness and limits.

Since LUBM is used by many open source projects as well as commercial products (such as: BigOWLIM[4], Oracle[5] or OpenLink Virtuoso[6]) to advertise and compare the performances of their RDF storage and inference systems, this report presents an in depth analysis of the characteristics of LUBM and it describes two scalable solutions (one serial and one parallel) to perform the necessary inference to answer all the fourteen queries correctly.

Parallel stream processing as well as advanced indexing techniques are two key factors to remove scalability issues related to inference with simple ontologies over large datasets. As shown by this report, it is relatively straightforward to implement a specialized and scalable reasoner for LUBM. While the implementation techniques described in this report are limited to the LUBM benchmark, some of the heuristics and aspects of the proposed approach might be extended to broader scenarios and, hopefully, to OWL ter-Horst or the newer OWL 2 RL profile.

The report is organized as follows. Section 2 describes related work focusing on the scalability issues related to inferencing. Section 3 illustrates the steps that have been followed to implement a scalable solution for doing inference over large LUBM datasets. Section 4 reports some of the measurements and evaluations of the solution proposed. Finally, section 5 provides our conclusions and future work directions relatively to scalable inferencing systems.

2 Related Work

The solution described in this report has been inspired by the work done by Soma and Prasanna [6][7] with the important difference that while they partition the data into groups and organize the processing into multiple rounds, we implemented our solution using a *streaming* technique.

Another approach has been proposed by Oren, Kotoulas et. al. [8]. Their solutions similarly partition the data to be processed and repartition the output between the processors. However, the processing does not exploit streaming techniques.

A completely different approach is described by Urbani [9][10] who proposes the use of Hadoop and the MapReduce model to perform RDF Schema inference and OWL ter-Horst. Hadoop has certainly better scalability and fault tolerance properties than the solution we propose in this report. However, multiple scans over the entire dataset are needed and faster implementations seem possible with fewer machines.

[3] http://esw.w3.org/topic/RdfStoreBenchmarking
[4] http://www.ontotext.com/owlim/benchmarking/lubm.html
[5] http://www.oracle.com/technology/tech/semantic_technologies/htdocs/performance.html
[6] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSArticleLUBMBenchmark

Finally, Kiryakov [11] suggests a target loading and inferencing speed of 100,000 triples/s for LUBM-8000 on a system which will cost less than 10,000 € by the end of 2010.

3 A Parallel Reasoner for LUBM

This section describes the Univ-Bench ontology and how to derive a specialized ruleset which captures the entire semantics of the Univ-Bench ontology. Once a ruleset is produced, simple techniques to transform a ruleset into an equivalent one are listed as well as motivations why it is convenient applying those transformations. A further optimization is possible, if the access patterns are known in advance, as it happens for LUBM and its fourteen queries.

At the end of the section, a serial and a parallel implementation of an inference engine for LUBM are described. Even if the proposed implementations are specific to LUBM, the approach and the overall architecture are applicable to broader scenarios and might be applicable to OWL 2 RL profiles; however, this is not the aim of this report.

3.1 Specialized Rulesets

The Univ-Bench ontology is a fairly simple OWL Lite ontology. It is composed by 43 OWL classes[7], 24 object properties[8] and 7 datatype properties[9]. Only a subset of the OWL Lite classes and properties has been actually used. Moreover, the ontology employs only a limited number of patterns.

The notions of a chair or a dean are defined as follows (throughout all this document, Turtle[10] syntax is used to represent serialized RDF):

```
:Chair
    A owl:Class ;
    rdfs:label "chair" ;
    rdfs:subClassOf :Professor ;
    owl:intersectionOf (
        :Person [
            a owl:Restriction ;
            owl:onProperty :headOf ;
            owl:someValuesFrom :Department
        ])

:Dean
    A owl:Class ;
    rdfs:label "dean" ;
    rdfs:subClassOf :Professor ;
    owl:intersectionOf (
```

[7] `grep "owl:Class rdf:ID" univ-bench.owl | wc -l`
[8] `grep "owl:ObjectProperty rdf:ID" univ-bench.owl | wc -l`
[9] `grep "owl:DatatypeProperty rdf:ID" univ-bench.owl | wc -l`
[10] http://www.w3.org/TeamSubmission/turtle/

```
        :Person [
            a owl:Restriction ;
            owl:onProperty :headOf ;
            owl:someValuesFrom :College
        ])
```

The notion of a graduate student is defined as:

```
:GraduateStudent
    a owl:Class ;
    rdfs:label "graduate student" ;
    rdfs:subClassOf :Person ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty :takesCourse ;
        owl:someValuesFrom :GraduateCourse
    ] .
```

Range and domain are specified for almost all of the object properties, for example:

```
:researchProject
    a owl:ObjectProperty ;
    rdfs:label "has as a research project" ;
    rdfs:domain :ResearchGroup ;
    rdfs:range :Research .
```

A few object properties are declared as inverse properties of another one, for example:

```
:degreeFrom
    a owl:ObjectProperty ;
    rdfs:label "has a degree from" ;
    rdfs:domain :Person ;
    rdfs:range :University ;
    owl:inverseOf :hasAlumnus .
```

Subclasses and subproperties are specified for most of the classes and properties, for example:

```
:VisitingProfessor
    a owl:Class ;
    rdfs:label "visiting professor" ;
    rdfs:subClassOf :Professor . :worksFor
    a owl:ObjectProperty ;
    rdfs:label "Works For" ;
    rdfs:subPropertyOf :memberOf .
```

Only one property is defined as a transitive property and that is:

```
:subOrganizationOf
    a owl:TransitiveProperty ;
    rdfs:domain :Organization ;
    rdfs:label "is part of" ;
    rdfs:range :Organization .
```

From these ontology fragments inference rules can be derived, as follows (these rules are written using the Jena2 syntax for rules).

From the fragments which include owl:intersectionOf and owl:Restriction used to define the notions of chair and dean, we derive respectively these rules:

```
(?x rdf:type ub:Chair) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:Person) (?x ub:headOf ?y) (?y rdf:type ub:Department)
-> (?x rdf:type ub:Chair) .
(?x rdf:type ub:Chair) -> exists ?y : (?x rdf:type ub:Person) (?x
ub:headOf ?y) (?y rdf:type ub:Department) .

(?x rdf:type ub:Dean) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:Person) (?x ub:headOf ?y) (?y rdf:type ub:College) ->
(?x rdf:type ub:Dean) .
(?x rdf:type ub:Dean) -> exists ?y : (?x rdf:type ub:Person) (?x
ub:headOf ?y) (?y rdf:type ub:College) .
```

The third rule on both examples is not actually a rule. It has no practical consequence for the inference required by LUBM and it represents the necessary condition that if a chair or a dean exists then it must be a person and it must be necessarily be a head of a department or a college.

A similar situation arises from the use of rdfs:subClassOf in conjunction with owl:Restriction. For example, for graduate students the rules are:

```
(?x rdf:type ub:GraduateStudent) -> (?x rdf:type ub:Person) .
(?x rdf:type ub:GraduateStudent) -> exists ?y : (?x ub:takesCourse
?y) (?y rdf:type ub:GraduateCourse) .
```

Only the first rule has an effect from a point of view of the implementation of a forward inference engine.

The rules derived from ranges, domains, inverse or transitive properties or subclasses/subproperties are trivial and, for the examples shown above, are:

```
(?x ub:researchProject ?y) -> (?x rdf:type ub:ResearchGroup) .
(?x ub:researchProject ?y) -> (?y rdf:type ub:Research) .

(?x ub:degreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:degreeFrom ?y) -> (?y rdf:type ub:University) .
(?x ub:degreeFrom ?y) -> (?y ub:hasAlumnus ?x) .
(?x ub:hasAlumnus ?y) -> (?y ub:degreeFrom ?x) .

(?x rdf:type ub:VisitingProfessor) -> (?x rdf:type ub:Professor) .

(?x ub:worksFor ?y) -> (?x ub:memberOf ?y) .

(?x ub:subOrganizationOf ?y) -> (?x rdf:type ub:Organization) .
(?x ub:subOrganizationOf ?y) -> (?y rdf:type ub:Organization) .
(?x ub:subOrganizationOf ?y) (?y ub:subOrganizationOf ?z) -> (?x
ub:subOrganizationOf ?z) .
```

Given the simplicity of the Univ-Bench ontology and the limited number of patterns used, it is possible to automatically generate a specialized ruleset which captures the semantics of the Univ-Bench ontology. The author successfully used SPARQL queries over the ontology and a template engine (Velocity), in order to automatically generate the ruleset included in Appendix A. An example of SPARQL query and corresponding Velocity template is also included in Appendix A.

3.2 Rulesets Transformations

Given a ruleset, there are *data dependencies* between rules since a rule might generate triples which trigger other rules. Some of these data dependencies can be removed applying simple transformations from a ruleset into an equivalent one.

For example, in the LUBM ruleset there are these rules:

```
R1:  (?x rdf:type ub:AdministrativeStaff) -> (?x rdf:type
ub:Employee) .
R2:  (?x rdf:type ub:SystemsStaff) -> (?x rdf:type
ub:AdministrativeStaff) .
R3:  (?x rdf:type ub:ClericalStaff) -> (?x rdf:type
ub:AdministrativeStaff) .
```

There is a data dependency between R2 and R1 and between R3 and R1, because the deductions of R2 or R3 always trigger the execution of R1. Therefore, the deductions of R1 can be added to the deductions of R2 and R3. This simple transformation removes any data dependency between R1, R2 and R3. The equivalent rulset is:

```
R1:  (?x rdf:type ub:AdministrativeStaff) -> (?x rdf:type
ub:Employee) .
R2:  (?x rdf:type ub:SystemsStaff) -> (?x rdf:type
ub:AdministrativeStaff) (?x rdf:type ub:Employee) .
R3:  (?x rdf:type ub:ClericalStaff) -> (?x rdf:type
ub:AdministrativeStaff) (?x rdf:type ub:Employee) .
```

Another type of transformation between rulesets can be described using the following rules as example:

```
R1:  (?x rdf:type ub:Person) (?x ub:headOf ?y) (?y rdf:type
ub:College) -> (?x rdf:type ub:Dean) .
R2:  (?x ub:headOf ?y) -> (?x ub:worksFor ?y) .
R3:  (?x ub:worksFor ?y) -> (?x ub:memberOf ?y) .
R4:  (?x ub:memberOf ?y) -> (?y ub:member ?x) .
R5:  (?x ub:member ?y) -> (?y rdf:type ub:Person) .
```

This chain of rules is sufficient to show that the triple pattern (?x ub:headOf ?y) implies the triple pattern (?x rdf:type ub:Person), therefore we can eliminate the first triple pattern from the premises of R1. The following rules are equivalent to the previous ones, but they have no data dependencies:

```
R1:  (?x ub:headOf ?y) (?y rdf:type ub:College) -> (?x rdf:type
ub:Dean) .
R2:  (?x ub:headOf ?y) -> (?x ub:worksFor ?y) (?x ub:memberOf ?y) (?y
ub:member ?x) (?x rdf:type ub:Person)  .
R3:  (?x ub:worksFor ?y) -> (?x ub:memberOf ?y) (?y ub:member ?x) (?x
rdf:type ub:Person) .
R4:  (?x ub:memberOf ?y) -> (?y ub:member ?x) (?x rdf:type ub:Person)
.
R5:  (?x ub:member ?y) -> (?y rdf:type ub:Person) .
```

Transformations which maintains equivalence between rulesets but reduce data dependencies between rules are useful because allow partitioning of a ruleset into multiple rulesets that can be executed independently, one after the other or in parallel.

Similarly, the absence of dependencies between rules allows exploiting data parallelism for the inferencing. The complete simplified and equivalent LUBM ruleset is included in Appendix B. Although this can be automated, a general solution to apply transformations to rulesets in order to reduce data dependencies between rules has not been implemented yet.

3.3 Minimal Ruleset

The LUBM benchmark specifies fourteen queries. From each triple pattern that appears in a query we can derive the closure of all the rules that might generate a matching triple. In other words, we can use the fourteen LUBM queries to derive a minimal subset of rules which contains only the necessary inference to correctly answer all the LUBM queries.

The complete list of triple patterns used by the fourteen LUBM queries is:

```
?x rdf:type ub:Professor
?x rdf:type ub:Person
?x rdf:type ub:Student
?x rdf:type ub:Course
?x rdf:type ub:Faculty
?x rdf:type ub:Chair
?x ub:memberOf <http://www.Department0.University0.edu>
?x ub:memberOf ?y
?x ub:subOrganizationOf ?y
?x ub:subOrganizationOf <http://www.University0.edu>
<http://www.University0.edu> ub:hasAlumnus ?x
```

Therefore, all the rules that might generate a triple that matches any of these triple patterns must be included in a minimal ruleset. This can be automated, however the solution would depends on the particular data structure used to represent the rules. The minimal LUBM ruleset is included in Appendix C.

We invite the reader to notice that the only data dependencies between the rules in the minimal LUBM ruleset are for the following rules:

```
R1:  (?x rdf:type ub:GraduateCourse) -> (?x rdf:type ub:Course) (?x
rdf:type ub:Work) .
R2:  (?x ub:takesCourse ?y) (?y rdf:type ub:Course) ->  (?x rdf:type
ub:Student) .
R3:  (?x ub:teachingAssistantOf ?y) (?y rdf:type ub:Course) ->  (?x
rdf:type ub:TeachingAssistant) .
R4:  (?x ub:subOrganizationOf ?y) (?y ub:subOrganizationOf ?z) -> (?x
ub:subOrganizationOf ?z) .
```

There is a data dependency between R1 and R2 and between R1 and R3 because R2 and R3 have a triple pattern in their premises that is included in the consequences of R1. These data dependencies cannot be removed. Finally, R4 depends on itself; this is the typical self dependency for transitive properties over instance data.

3.4 Serial Reasoner

Looking at the minimal LUBM ruleset in Appendix C, it is immediate to notice that the majority of rules have only one triple pattern in their premises: these rules can easily be implemented processing the dataset one triple at the time (i.e. in a streaming fashion). There are only 7 rules with two triple patterns in their premises:

```
(?x ub:headOf ?y) (?y rdf:type ub:Department) -> (?x rdf:type
ub:Chair) (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty) (?x
rdf:type ub:Employee) .
(?x ub:headOf ?y) (?y rdf:type ub:College) -> (?x rdf:type ub:Dean)
(?x rdf:type ub:Professor) (?x rdf:type ub:Faculty) (?x rdf:type
ub:Employee) .
(?x ub:headOf ?y) (?y rdf:type ub:Program) -> (?x rdf:type
ub:Director) (?x rdf:type ub:Person) .
(?x ub:worksFor ?y) (?y rdf:type ub:Organization) -> (?x rdf:type
ub:Employee) .
 (?x ub:takesCourse ?y) (?y rdf:type ub:Course) -> (?x rdf:type
ub:Student) .
(?x ub:teachingAssistantOf ?y) (?y rdf:type ub:Course) -> (?x
rdf:type ub:TeachingAssistant) .
(?x ub:subOrganizationOf ?y) (?y ub:subOrganizationOf ?z) -> (?x
ub:subOrganizationOf ?z) .
```

Each one of these rules requires a join. For the ub:subOrganizationOf property the results needs to be fed back to the rule itself.

A specialized LUBM reasoner that implements only the minimal ruleset and performs reasoning in a streaming fashion has been implemented and it can process LUBM-8000 dataset at a speed of about 45,000 triples/s on a single machine. TDB indexing subsystem has been used to implement symmetric hash joins [13]. TDB uses memory mapped files to manage its indexes; however, while the amount of memory is not a limiting factor, it is appropriate to avoid swapping.

This custom serial reasoner is not a general reasoner and it implements LUBM reasoning only. However, it can be envisaged a system which, given an ontology, for which it is possible an implementation using rules, derives a ruleset and generates custom code to perform the reasoning.

Probably, it is also possible to apply the same approach to RDF Schema reasoning, OWL ter-Horst and, perhaps, OWL 2 RL profile.

3.5 Parallel Reasoner

Although many optimizations and improvements to the trivial implementation briefly described in the previous section are possible, to achieve an order of magnitude improvement we looked at how to parallelize the processing using more than one machine in a cluster of industry standard machines configured according to a shared nothing architecture.

The cluster we are referring to is a small one, on the order of 10 machines rather than hundreds or thousands. In this situation, the probability of a failure during the processing is low. The processing also involves datasets on the order of billions of

triples rather than tens billion triples. No fault tolerance techniques or replication have been considered.

The authors are well aware of the, so called, *eight fallacies*[11] of distributed computing and the implementation described in this section has almost all of them. However, the aim has been to explore possibilities and the design space rather than provide a production ready solution. Correctness, soundness and processing speed have been the priorities during this investigation.

Data can be partitioned across multiple machines as well as rules. We decided to implement a solution which allows for random partitioning of the data and which does not partition the rules. All the machines in the cluster have the complete ruleset and they perform LUBM reasoning using the serial reasoner described in the previous section.

However, in front of and after each serial reasoner running on each machine there is a *partitioner* which decides for each triple if the triple should be processed locally or if it should be sent to a remote machine. The pseudo code of the partitioner is the following:

```
for each triple (s, p, o):

if (p is LUBM.headOf) or (p is LUBM.takesCourse)
    send triple to machine abs(hash(o)) % N
else if (p is LUBM.subOrganizationOf)
    send triple to machine abs(hash(s)) % N
    send triple to machine abs(hash(o)) % N
else if (p is RDF.type)
    if (o is LUBM.Department) or (o is LUBM.College) or (o is LUBM.Course)
        send triple to abs(hash(s)) % N
    else
        process triple locally
else
    process triple locally
```

This is nothing more than hash partitioning over the join variable for the rules with two triple patterns. The partitioning must be applied to all the inferred triples which may need to be forwarded to remote machines as well as stored locally as result.

This custom parallel reasoner is not a general reasoner and it implements LUBM reasoning only. However, once again, the partitioning function can be automatically generated from a ruleset and the same approach might be adopted for RDF Schema, OWL ter-Horst and, possibly, OWL 2 RL profile.

4 Preliminary Results

This section reports only preliminary results using a small cluster of 8 machines. Each machine is an HP bl460c running Linux RHEL 5 and it has 2 x Dual Core Intel Xeon 5160 @ 3GHz, 8 GB of RAM and about 130 GB hard drive. They are interconnected with 1 Gigabit Ethernet network ports.

[11] http://blogs.sun.com/jag/resource/Fallacies.html

One first set of experiments consisted in running the serial reasoner over different sizes of the LUBM datasets. This was done to check that the processing speed does not change significantly as the dataset size grows. The processing speed using one single machine was about 45,000 triples/s and it took about 24,000 seconds to infer about 869,000,000 triples over LUBM-8000 (i.e. about 1,000,000,000 triples). The TDB indexes were about 15 GB.

Another set of experiment consisted in splitting the LUBM-8000 dataset into multiple chunks and using the parallel reasoner to perform inference. The aggregate processing speed using 8 machines was about 330,000 triples/s (i.e. about 41,000 triples/s per node) and it took less than 3,600 seconds to process the entire LUBM-8000.

The code of an initial prototype used to gather these preliminary results has been published on a personal scratch area[12] of the Jena repository on SourceForge.

5 Conclusions

We described how, starting from the LUBM ontology, we derived a specialized ruleset and how we applied simple transformations to generate an equivalent ruleset but with less data dependencies between rules.

We presented a simple serial implementation of a reasoner which performs inference in a streaming fashion and we described how a serial reasoner can be used in a parallel implementation which simply uses hash partitioning over variables of triple patterns which involves joins. This approach to parallel processing does not depend on the particular serial reasoner running at each machine in the cluster.

The implementation has not been heavily optimized; for example, no compression has been used to stream triples between machines. Despite this, processing speeds in the order of hundred thousand triples per second are possible even with relatively small clusters.

Although LUBM requires only limited inference capabilities, it is used by others and it is has been helpful to compare different solutions and approaches.

Finally, more experiments will be necessary to better characterize the proposed approach in terms of scalability, network traffic and load balancing properties. It would be interesting to verify if the same approach can be used for RDF Schema, OWL ter-Horst and OWL 2 RL profile.

[12] https://jena.svn.sf.net/svnroot/jena/Scratch/PC/LUBM/

References

1. Guo, Y., Pan, Z., Helfin, J., *An Evaluation of Knowledge Base Systems for Large OWL Data Sets*. In Proc. of the 3rd International Semantic Web Conference, 2004
2. Guo, Y., Pan, Z., Helfin, J., *LUBM: A Benchmark for OWL Knowledge Base Systems*. Journal of Web Semantics, vol. 3, no. 2, pp. 158-182, 2005
3. http://swat.cse.lehigh.edu/projects/lubm/
4. Weithoner, T., Liebig, T., et. al., *What's Wrong with OWL Benchmarks?*. In Proc. of the Second Int. Workshop on Scalable Semantic Web Knowledge Base Systems, pp. 101-114, 2006
5. Ma, L., Yang, Y., et. al., *Towards a Complete OWL Ontology Benchmark*. In Proc. of the third European Semantic Web Conference, pp. 124-139, 2006
6. Soma, R., Prasanna, V., *Parallel inferencing for OWL knowledge bases*. In Proc. of the International Conference on Parallel Processing, pp. 75-82, 2008
7. Soma, R., Prasanna, V., *A Data Partitioning Approach for Parallelizing Rule Based Inferencing for Materialized OWL Knowledge Bases*. In Proc. of the 21st International Conference on Parallel and Distributed Computing and Communication Systems, pp. 19-25, 2008
8. Oren, E., Kotoulas, S., et. al., *Marvin: A platform for large-scale analysis of Semantic Web data*. In Proc. of the International Web Science conference, 2009
9. Urbani, J., *Scalable Distributed RDFS/OWL Reasoning using MapReduce*. Mather's thesis, Vrije Universiteit Amsterdam, 2009
10. Urbani, J., Kotoulas, S., et. al., *Scalable Distributed Reasoning using MapReduce*. To appear in Proc. of the International Semantic Web Conference, 2009
11. Kiryakov, A., *Measurable Targets for Scalable Reasoning*, D5.5.1 deliverable, LarKC project, 2008
12. ter Horst, H. J., *Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary*. Journal of Web Semantics, vol. 3(2-3), pp. 79-115, 2005
13. Wilschut, A.N. and Apers, P.M.G., *Dataflow query execution in a parallel main-memory environment*. Distributed and Parallel Databases, vol. 1, no. 1, pp. 103-128, 1993

Appendix A

The following is a Jena2 syntax ruleset which capture the semantic of Univ-Bench ontology. This ruleset has been automatically generated using SPARQL queries and Velocity templates.

```
@prefix ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#> .

# owl:intersectionOf

(?x rdf:type ub:Person) (?x ub:teachingAssistantOf ?y) (?y rdf:type ub:Course) -> (?x
rdf:type ub:TeachingAssistant) .
(?x rdf:type ub:Person) (?x ub:takesCourse ?y) (?y rdf:type ub:Course) -> (?x rdf:type
ub:Student) .
(?x rdf:type ub:Person) (?x ub:worksFor ?y) (?y rdf:type ub:Organization) -> (?x rdf:type
ub:Employee) .
(?x rdf:type ub:Person) (?x ub:headOf ?y) (?y rdf:type ub:Program) -> (?x rdf:type
ub:Director) .
(?x rdf:type ub:Person) (?x ub:headOf ?y) (?y rdf:type ub:College) -> (?x rdf:type ub:Dean)
.
(?x rdf:type ub:Person) (?x ub:headOf ?y) (?y rdf:type ub:Department) -> (?x rdf:type
ub:Chair) .

# Subproperties

(?x ub:worksFor ?y) -> (?x ub:memberOf ?y) .
(?x ub:undergraduateDegreeFrom ?y) -> (?x ub:degreeFrom ?y) .
(?x ub:mastersDegreeFrom ?y) -> (?x ub:degreeFrom ?y) .
(?x ub:headOf ?y) -> (?x ub:worksFor ?y) .
(?x ub:doctoralDegreeFrom ?y) -> (?x ub:degreeFrom ?y) .

# Inverse properties

(?x ub:memberOf ?y) -> (?y ub:member ?x) .
(?x ub:member ?y) -> (?y ub:memberOf ?x) .
(?x ub:hasAlumnus ?y) -> (?y ub:degreeFrom ?x) .
(?x ub:degreeFrom ?y) -> (?y ub:hasAlumnus ?x) .
(?x ub:degreeFrom ?y) -> (?y ub:hasAlumnus ?x) .
(?x ub:hasAlumnus ?y) -> (?y ub:degreeFrom ?x) .

# Subclasses

(?x rdf:type ub:TechnicalReport) -> (?x rdf:type ub:Article) .
(?x rdf:type ub:University) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:FullProfessor) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:AdministrativeStaff) -> (?x rdf:type ub:Employee) .
(?x rdf:type ub:Chair) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:Professor) -> (?x rdf:type ub:Faculty) .
(?x rdf:type ub:Faculty) -> (?x rdf:type ub:Employee) .
(?x rdf:type ub:Manual) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:JournalArticle) -> (?x rdf:type ub:Article) .
(?x rdf:type ub:Course) -> (?x rdf:type ub:Work) .
(?x rdf:type ub:UndergraduateStudent) -> (?x rdf:type ub:Student) .
(?x rdf:type ub:Program) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:ConferencePaper) -> (?x rdf:type ub:Article) .
(?x rdf:type ub:SystemsStaff) -> (?x rdf:type ub:AdministrativeStaff) .
(?x rdf:type ub:ResearchGroup) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:Book) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:Specification) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:Software) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:Department) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:Research) -> (?x rdf:type ub:Work) .
(?x rdf:type ub:Dean) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:AssociateProfessor) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:Lecturer) -> (?x rdf:type ub:Faculty) .
(?x rdf:type ub:ResearchAssistant) -> (?x rdf:type ub:Student) .
(?x rdf:type ub:College) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:PostDoc) -> (?x rdf:type ub:Faculty) .
(?x rdf:type ub:Institute) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:Article) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:UnofficialPublication) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:VisitingProfessor) -> (?x rdf:type ub:Professor) .
```

```
(?x rdf:type ub:GraduateCourse) -> (?x rdf:type ub:Course) .
(?x rdf:type ub:AssistantProfessor) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:GraduateStudent) -> (?x rdf:type ub:Person) .
(?x rdf:type ub:ClericalStaff) -> (?x rdf:type ub:AdministrativeStaff) .

# Ranges of properties

(?x ub:affiliatedOrganizationOf ?y) -> (?y rdf:type ub:Organization) .
(?x ub:teacherOf ?y) -> (?y rdf:type ub:Course) .
(?x ub:advisor ?y) -> (?y rdf:type ub:Professor) .
(?x ub:softwareDocumentation ?y) -> (?y rdf:type ub:Publication) .
(?x ub:teachingAssistantOf ?y) -> (?y rdf:type ub:Course) .
(?x ub:member ?y) -> (?y rdf:type ub:Person) .
(?x ub:researchProject ?y) -> (?y rdf:type ub:Research) .
(?x ub:affiliateOf ?y) -> (?y rdf:type ub:Person) .
(?x ub:orgPublication ?y) -> (?y rdf:type ub:Publication) .
(?x ub:mastersDegreeFrom ?y) -> (?y rdf:type ub:University) .
(?x ub:degreeFrom ?y) -> (?y rdf:type ub:University) .
(?x ub:hasAlumnus ?y) -> (?y rdf:type ub:Person) .
(?x ub:subOrganizationOf ?y) -> (?y rdf:type ub:Organization) .
(?x ub:publicationResearch ?y) -> (?y rdf:type ub:Research) .
(?x ub:publicationAuthor ?y) -> (?y rdf:type ub:Person) .
(?x ub:undergraduateDegreeFrom ?y) -> (?y rdf:type ub:University) .
(?x ub:doctoralDegreeFrom ?y) -> (?y rdf:type ub:University) .
(?x ub:listedCourse ?y) -> (?y rdf:type ub:Course) .

# Domains of properties

(?x ub:publicationAuthor ?y) -> (?x rdf:type ub:Publication) .
(?x ub:undergraduateDegreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:telephone ?y) -> (?x rdf:type ub:Person) .
(?x ub:doctoralDegreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:mastersDegreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:emailAddress ?y) -> (?x rdf:type ub:Person) .
(?x ub:advisor ?y) -> (?x rdf:type ub:Person) .
(?x ub:age ?y) -> (?x rdf:type ub:Person) .
(?x ub:softwareVersion ?y) -> (?x rdf:type ub:Software) .
(?x ub:softwareDocumentation ?y) -> (?x rdf:type ub:Software) .
(?x ub:publicationResearch ?y) -> (?x rdf:type ub:Publication) .
(?x ub:affiliateOf ?y) -> (?x rdf:type ub:Organization) .
(?x ub:title ?y) -> (?x rdf:type ub:Person) .
(?x ub:affiliatedOrganizationOf ?y) -> (?x rdf:type ub:Organization) .
(?x ub:orgPublication ?y) -> (?x rdf:type ub:Organization) .
(?x ub:teacherOf ?y) -> (?x rdf:type ub:Faculty) .
(?x ub:degreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:listedCourse ?y) -> (?x rdf:type ub:Schedule) .
(?x ub:member ?y) -> (?x rdf:type ub:Organization) .
(?x ub:hasAlumnus ?y) -> (?x rdf:type ub:University) .
(?x ub:tenured ?y) -> (?x rdf:type ub:Professor) .
(?x ub:researchProject ?y) -> (?x rdf:type ub:ResearchGroup) .
(?x ub:subOrganizationOf ?y) -> (?x rdf:type ub:Organization) .
(?x ub:teachingAssistantOf ?y) -> (?x rdf:type ub:TeachingAssistant) .
(?x ub:publicationDate ?y) -> (?x rdf:type ub:Publication) .

# Transitive properties

(?x ub:subOrganizationOf ?y) (?y ub:subOrganizationOf ?z) -> (?x ub:subOrganizationOf ?z) .

# Subclasses via owl:Restriction

# (?x rdf:type ub:ResearchAssistant) -> ThereExists ?y suchThat { (?x ub:worksFor ?y) (?y
rdf:type ub:ResearchGroup) } .
# (?x rdf:type ub:GraduateStudent) -> ThereExists ?y suchThat { (?x ub:takesCourse ?y) (?y
rdf:type ub:GraduateCourse) } .
```

An example of SPARQL query and corresponding Velocity template follows:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?t0 ?t1 ?t2 ?p1 {
    ?t0 owl:intersectionOf (?t1 ?b) .
    ?b rdf:type owl:Restriction .
    ?b owl:onProperty ?p1 .
```

```
    ?b owl:someValuesFrom ?t2 .
}
```

Corresponding Velocity template:

```
#foreach ($row in $result)
(?x rdf:type ub:$row.get("t1").getLocalName())
(?x ub:$row.get("p1").getLocalName() ?y)
(?y rdf:type ub:$row.get("t2").getLocalName()) ->
(?x rdf:type ub:$row.get("t0").getLocalName()) .
#end
```

Appendix B

```
@prefix ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#> .

(?x rdf:type ub:Course) -> (?x rdf:type ub:Work) .
(?x rdf:type ub:Research) -> (?x rdf:type ub:Work) .
(?x rdf:type ub:GraduateCourse) -> (?x rdf:type ub:Course) (?x rdf:type ub:Work) .
(?x rdf:type ub:UndergraduateStudent) -> (?x rdf:type ub:Student) .
(?x rdf:type ub:ResearchAssistant) -> (?x rdf:type ub:Student) .
(?x rdf:type ub:GraduateStudent) -> (?x rdf:type ub:Person) .
(?x rdf:type ub:Faculty) -> (?x rdf:type ub:Employee) .
(?x rdf:type ub:Professor) -> (?x rdf:type ub:Faculty) (?x rdf:type ub:Employee) .
(?x rdf:type ub:AssistantProfessor) -> (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty)
(?x rdf:type ub:Employee) .
(?x rdf:type ub:AssociateProfessor) -> (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty)
(?x rdf:type ub:Employee) .
(?x rdf:type ub:Dean) -> (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty) (?x rdf:type
ub:Employee) .
(?x rdf:type ub:FullProfessor) -> (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty) (?x
rdf:type ub:Employee) .
(?x rdf:type ub:Chair) -> (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty) (?x rdf:type
ub:Employee) .
(?x rdf:type ub:VisitingProfessor) -> (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty)
(?x rdf:type ub:Employee) .
(?x rdf:type ub:Lecturer) -> (?x rdf:type ub:Faculty) (?x rdf:type ub:Employee) .
(?x rdf:type ub:PostDoc) -> (?x rdf:type ub:Faculty) (?x rdf:type ub:Employee) .
(?x rdf:type ub:AdministrativeStaff) -> (?x rdf:type ub:Employee) .
(?x rdf:type ub:ConferencePaper) -> (?x rdf:type ub:Article) (?x rdf:type ub:Publication) .
(?x rdf:type ub:JournalArticle) -> (?x rdf:type ub:Article) (?x rdf:type ub:Publication) .
(?x rdf:type ub:TechnicalReport) -> (?x rdf:type ub:Article) (?x rdf:type ub:Publication) .
(?x rdf:type ub:Book) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:Software) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:Specification) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:UnofficialPublication) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:Manual) -> (?x rdf:type ub:Publication) .
(?x rdf:type ub:Department) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:Institute) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:College) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:ResearchGroup) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:University) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:Program) -> (?x rdf:type ub:Organization) .
(?x rdf:type ub:ClericalStaff) -> (?x rdf:type ub:AdministrativeStaff) .
(?x rdf:type ub:SystemsStaff) -> (?x rdf:type ub:AdministrativeStaff) .


(?x ub:advisor ?y) -> (?x rdf:type ub:Person) (?y rdf:type ub:Professor) (?y rdf:type
ub:Faculty) (?y rdf:type ub:Employee) .
(?x ub:affiliateOf ?y) -> (?x rdf:type ub:Organization) (?y rdf:type ub:Person) .
(?x ub:age ?y) -> (?x rdf:type ub:Person) .
(?x ub:degreeFrom ?y) -> (?x rdf:type ub:Person) (?y rdf:type ub:University) (?y rdf:type
ub:Organization) (?y ub:hasAlumnus ?x) .
(?x ub:doctoralDegreeFrom ?y) -> (?x rdf:type ub:Person) (?y rdf:type ub:University) (?y
rdf:type ub:Organization) (?x ub:degreeFrom ?y) (?y ub:hasAlumnus ?x) .
(?x ub:emailAddress ?y) -> (?x rdf:type ub:Person) .
(?x ub:mastersDegreeFrom ?y) -> (?x rdf:type ub:Person) (?y rdf:type ub:University) (?y
rdf:type ub:Organization) (?x ub:degreeFrom ?y) (?y ub:hasAlumnus ?x) .
```

```
(?x ub:hasAlumnus ?y) -> (?x rdf:type ub:University) (?x rdf:type ub:Organization) (?y
rdf:type ub:Person) (?y ub:degreeFrom ?x) .
(?x ub:member ?y) -> (?x rdf:type ub:Organization) (?y rdf:type ub:Person) (?y ub:memberOf
?x) .
(?x ub:memberOf ?y) -> (?y rdf:type ub:Organization) (?x rdf:type ub:Person) (?y ub:member
?x) .
(?x ub:worksFor ?y) -> (?x ub:memberOf ?y) (?y ub:member ?x) (?y rdf:type ub:Organization)
(?x rdf:type ub:Person) .
(?x ub:headOf ?y) -> (?x ub:worksFor ?y) (?x ub:memberOf ?y) (?y ub:member ?x) (?y rdf:type
ub:Organization) (?x rdf:type ub:Person) .
(?x ub:title ?y) -> (?x rdf:type ub:Person) .
(?x ub:undergraduateDegreeFrom ?y) -> (?x rdf:type ub:Person) (?y rdf:type ub:University)
(?y rdf:type ub:Organization) (?x ub:degreeFrom ?y) (?y ub:hasAlumnus ?x) .
(?x ub:telephone ?y) -> (?x rdf:type ub:Person) .
(?x ub:publicationAuthor ?y) -> (?x rdf:type ub:Publication) (?y rdf:type ub:Person) .
(?x ub:tenured ?y) -> (?x rdf:type ub:Professor) (?x rdf:type ub:Faculty) (?x rdf:type
ub:Employee) .
(?x ub:teacherOf ?y) -> (?x rdf:type ub:Faculty) (?x rdf:type ub:Employee) (?y rdf:type
ub:Course) (?y rdf:type ub:Work) .
(?x ub:teachingAssistantOf ?y) -> (?x rdf:type ub:TeachingAssistant) (?y rdf:type ub:Course)
(?y rdf:type ub:Work) .
(?x ub:affiliatedOrganizationOf ?y) -> (?x rdf:type ub:Organization) (?y rdf:type
ub:Organization) .
(?x ub:orgPublication ?y) -> (?x rdf:type ub:Organization) (?y rdf:type ub:Publication) .
(?x ub:subOrganizationOf ?y) -> (?x rdf:type ub:Organization) (?y rdf:type ub:Organization)
.
(?x ub:researchProject ?y) -> (?x rdf:type ub:ResearchGroup) (?x rdf:type ub:Organization) .
(?x ub:listedCourse ?y) -> (?x rdf:type ub:Schedule) (?y rdf:type ub:Course) (?y rdf:type
ub:Work) .
(?x ub:publicationResearch ?y) -> (?x rdf:type ub:Publication) (?y rdf:type ub:Research) (?y
rdf:type ub:Work) .
(?x ub:researchProject ?y) -> (?y rdf:type ub:Research) (?y rdf:type ub:Work) .
(?x ub:softwareDocumentation ?y) -> (?x rdf:type ub:Software) (?x rdf:type ub:Publication)
(?y rdf:type ub:Publication) .
(?x ub:softwareVersion ?y) -> (?x rdf:type ub:Software) (?x rdf:type ub:Publication) .
(?x ub:publicationDate ?y)  -> (?x rdf:type ub:Publication) .


(?x ub:headOf ?y) (?y rdf:type ub:Department) -> (?x rdf:type ub:Chair) (?x rdf:type
ub:Professor) (?x rdf:type ub:Faculty) (?x rdf:type ub:Employee) .
(?x ub:headOf ?y) (?y rdf:type ub:College) -> (?x rdf:type ub:Dean)  (?x rdf:type
ub:Professor) (?x rdf:type ub:Faculty) (?x rdf:type ub:Employee) .
(?x ub:headOf ?y) (?y rdf:type ub:Program) -> (?x rdf:type ub:Director) (?x rdf:type
ub:Person) .
(?x ub:worksFor ?y) (?y rdf:type ub:Organization) -> (?x rdf:type ub:Employee) .
(?x ub:takesCourse ?y) (?y rdf:type ub:Course) -> (?x rdf:type ub:Student) .
(?x ub:teachingAssistantOf ?y) (?y rdf:type ub:Course) -> (?x rdf:type ub:TeachingAssistant)
.


(?x ub:subOrganizationOf ?y) (?y ub:subOrganizationOf ?z) -> (?x ub:subOrganizationOf ?z) .
```

Appendix C

```
(?x rdf:type ub:FullProfessor) -> (?x rdf:type ub:Professor) .
(?x ub:headOf ?y) (?y rdf:type ub:Department) -> (?x rdf:type ub:Chair) .
(?x rdf:type ub:Chair) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:Person) (?x ub:headOf ?y) (?y rdf:type ub:College) -> (?x rdf:type ub:Dean)
.
(?x rdf:type ub:Dean) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:AssociateProfessor) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:VisitingProfessor) -> (?x rdf:type ub:Professor) .
(?x rdf:type ub:AssistantProfessor) -> (?x rdf:type ub:Professor) .
(?x ub:advisor ?y) -> (?y rdf:type ub:Professor) .
(?x ub:tenured ?y) -> (?x rdf:type ub:Professor) .

(?x rdf:type ub:GraduateCourse) -> (?x rdf:type ub:Course) .
(?x rdf:type ub:GraduateStudent) -> (?x rdf:type ub:Person) .
(?x ub:member ?y) -> (?y rdf:type ub:Person) .
(?x ub:affiliateOf ?y) -> (?y rdf:type ub:Person) .
(?x ub:hasAlumnus ?y) -> (?y rdf:type ub:Person) .
(?x ub:publicationAuthor ?y) -> (?y rdf:type ub:Person) .
```

```
(?x ub:undergraduateDegreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:telephone ?y) -> (?x rdf:type ub:Person) .
(?x ub:doctoralDegreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:mastersDegreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:emailAddress ?y) -> (?x rdf:type ub:Person) .
(?x ub:advisor ?y) -> (?x rdf:type ub:Person) .
(?x ub:age ?y) -> (?x rdf:type ub:Person) .
(?x ub:title ?y) -> (?x rdf:type ub:Person) .
(?x ub:degreeFrom ?y) -> (?x rdf:type ub:Person) .
(?x ub:takesCourse ?y) (?y rdf:type ub:Course) -> (?x rdf:type ub:Student) .
(?x rdf:type ub:UndergraduateStudent) -> (?x rdf:type ub:Student) .
(?x rdf:type ub:ResearchAssistant) -> (?x rdf:type ub:Student) .
(?x rdf:type ub:Professor) -> (?x rdf:type ub:Faculty) .
(?x rdf:type ub:Lecturer) -> (?x rdf:type ub:Faculty) .
(?x rdf:type ub:PostDoc) -> (?x rdf:type ub:Faculty) .
(?x ub:teacherOf ?y) -> (?x rdf:type ub:Faculty) .
(?x ub:headOf ?y) -> (?x ub:worksFor ?y) .
(?x ub:worksFor ?y) -> (?x ub:memberOf ?y) .
(?x ub:member ?y) -> (?y ub:memberOf ?x) .
(?x ub:subOrganizationOf ?y) (?y ub:subOrganizationOf ?z) -> (?x ub:subOrganizationOf ?z) .
(?x ub:mastersDegreeFrom ?y) -> (?x ub:degreeFrom ?y) .
(?x ub:doctoralDegreeFrom ?y) -> (?x ub:degreeFrom ?y) .
(?x ub:undergraduateDegreeFrom ?y) -> (?x ub:degreeFrom ?y) .
(?x ub:degreeFrom ?y) -> (?y ub:hasAlumnus ?x) .
```