Semantic Web & Relative Analysis of Inference Engines

Ravi L Verma [1], Prof. Arjun Rajput [2]

[1]*Information Technology, Technocrats Institute of Technology*

[2]*Computer Engineering, Technocrats Institute of Technology, Excellence*

Bhopal, MP

Abstract **- Semantic web is an enhancement to the current web that will provide an easier path to find, combine, share & reuse the information. It brings an idea of having data to be defined & linked in a way that can be used for more efficient discovery, integration, automation & reuse across many applications.Today's web is purely based on documents that are written in HTML, used for coding a body of text with interactive forms. Semantic web provides a common language for understanding how data related to the real world objects, allowing a person or machine to start with in one database and then move through an unending set of databases that are not connected by wires but by being about the same thing. For understanding & using the information & knowledge encoded in semantic web documents requires an inference engine. Inference engines are software applications that derive new concepts from existing information. By creating a framework of information and relationship, we allow reasoners to make logical conclusion based on the framework. Use of inference engine in semantic web allows the applications to investigate why a particular solution has been reached, i.e. semantic applications can provide proof of the derived conclusion. This paper is a study work & survey, which demonstrates a comparison of different type of inference engines in context of semantic web. It will help to distinguish among different inference engines which may be helpful to judge the various proposed prototype systems with different views on what an inference engine for the semantic web should do.**

Key Words **- Inference Engine, Knowledge Representation,Logical Reasoning, Semantic Web.**

I. INTRODUCTION

Inference engine can be defined as combination of finite state machine consisting three main actions: match rules, select rules, execute rules. In match rule state, inference engine maintains all the rules that are satisfied by the present contents of the data store. A rule being in typical condition-action form implies testing of the conditions against working memory. Matching that is found are all considered candidates for execution, collectively known as conflict set. Same rule can be applied number of times in the conflict set if different subsets of data items are matched. The combination of rule & subset of matching of data items is called derivation of the rule. Now the inference engine passes the conflict set to the second state, the select rules. In second state the inference engine implements some selection procedure to determine which rules will be actually executed. Selection strategy can be a part of the engine or can be taken as part of model. At last the selected derivations are passed to the final state, execution rule. The inference engine executes the selected rules, with the derivation data items as parameters. Normallythe actions belonging to the right hand side of any rule change the data store, but it can also implement further processing outside of the inference engine (interacting with the user with graphical user interface).The data store is normallyupdated by firing rules, different set of rules matches during the next cycle after performing of these

actions. Inference engine now cycles back to the first state & is ready to start again. This mechanism for controlling is called recognize-act cycle.When no rules match the data the inference engine either stops on a given number of cycles, controlled by the operator or on a given state of the data store.

A semantic reasoner, rule engine, reasoning engine or simply a reasoner, is software that is able to infer logical patterns from a set facts or axioms. The work of a semantic matches that of an inference engine, by providing a richer set of mechanism to work with. Inference rules are commonly specified with the help of ontology language, an also a description language. In logic, a rule of inference is a function from sets of formulae to formulae. Argument is termed as the premise set and value as the conclusion. They can also be termed as relations holdings between premises and conclusions, whereby the conclusions are said to be derived from premises. If the premise set is empty, then the conclusion is said to be theorem of the logic. Many reasoners use first-order predicate logic to perform reasoning, inference is commonly maintained by forward chaining and backward chaining.

Inference engines are of two types: forward chaining and backward chaining. Forward chaining proceeds with the available data and uses inference rules to extract more data until a specified goal is achieved. An inference engine that is using forward chaining searched the inference rules until it finds one where antecedent (If clause) is found to be true. When found it can conclude, or infer, the consequent (Then clause), resulting in the addition of any new information to its data. Now as the data determines which rules are to be selected and used, thismethod is also called data-driven, and backward chaining is called goal-driven.

Backward chaining proceeds or starts with a list of goals (or a hypothesis) and it works from the (Then clause to If clause) to see if there is any data available that is going to support any of these consequents. An inference engine that is using backward chaining will be searching the inference rules till it finds one which is having a consequent (Then clause) that matches a specified goal. If the antecedent (If clause) of that particular rule is not known to be true, then it is to be added to the list of goals (in order to confirm your goal data to confirm this new rule should also be provided).As the list of goals determines which rules are to be selected and used, this method is also called goal-driven.

In next section we will carry out functional description of various inference engines selected for the comparative study. Section III gives comparative chart for the selected inference engine measured on different performance criteria. Section IV concludes the paper by giving the scope and limitation of the study that we have performed.

II. INFERENCE ENGINES

In our comparative analysis we have studied the following inference engines:-

1) Race Pro

Racer Pro [3] can be specifies as an OWL reasoner and inference server for semantic web. Racer means Renamed ABox and concept expression reasoner. RacerPro can be implemented in industrial projects that are based on W3Cs standards: RDF and OWL, and is an important tool for research and development.Racer Pro is a system based on knowledge representation that implements a very highly optimized tableau calculus used for a very expressive description logic.

2) Jena

Jena is an open source semantic web framework for Java. Jena is supportive for OWL (Web Ontology Language). It contains many internal reasoners and also provide

medium for eternal reasoners through DIG interface.

3) Fact

Fact (Fast Classification of Terminologies) is a description logic (DL) for modal logic satisfiability testing.It contains two reasoners, one for logic SHF (ALC combined with functional roles and role hierarchy) and other for logic SHIQ (SHF combined with inverse roles and qualified number restriction). Interesting features of Fact are one: its expressive logic; two: its support for reasoning with arbitrary reasoning logic;three: its CORBA based architecture.

4) Fact++

Fact++ is an extended version of Fact that implements tableaux algorithms for SROIQ description logic andIs implemented in C++ but contains very limited user interface and services as compared to other existing reasoners.Uses some strategies like model merging, told cycle elimination, absorption and synonym replacement.

5) Pellet

Pellet applies Reasoning on SHIN (D) with strategies like TBox partitioning, absorption, semantic branching and dependency directed back jumping [4]. It comprise various optimization techniques including optimization for Incremental reasoning.

6) Hoolet

Hoolet is an execution of a web ontology language-description logic (OWL-DL) reasoner that uses a first order logic. Ontology is translated to collection of axioms and then this collection of axioms is given to a first order logic for consistency checking. It is implemented using the Wonder Web OWL API for parsing and processing OWL, and vampire logic for reasoning purpose [5].Approach is not scalable.

7) F-OWL

F-OWL [6] is the OWL inference engine that uses a Frame-based System to reason with OWL ontologies. FOWLis accompanied by a simple OWL importer that reads an OWL ontology from a URI and extracts RDF triples out of the ontology. The extracted RDF triples are converted to format appropriate for F-OWL's frame style and fed into theF-OWL engine. It then uses flora rules defined in flora-2 language to check the consistency of the ontology and extract hidden knowledge via resolution.

8) KAON2

KAON2 [7] is a replacement to KAON project. The main difference to KAON is the supported ontology language. KAON used extensions of RDFS, whereas KAON2 is based on Frame logic. It is an architecture for frame logic ontologies.KAON2 supports answering conjunctive queries, although without true non-distinguished variables.This means that all variables in a query are bound to individuals explicitly occurring in the knowledge base, even if they are not returned as part of the query answer.

9) Jess

Jess [8]is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. Using Jess, one can build Java software that has the capacity to "reason" using knowledge you supply in the form of declarative rules. Jess is small, light, and one of the fastest rule engines available. Its powerful scripting language gives you access to all of Java's Application Programming Interfaces. The reference implementation of Java Specification Request 94 is a driver for Jess; with it, you can connect Jess to Java software using the vendor-independent JSR 94 API.

10) SHER

Scalable Highly Expressive Reasoner (SHER) [9] is a breakthrough technology that provides ontology analytics over highly expressive ontologies (OWL-DL without nominal). SHER does not do any inferencing on load; hence it deals better with quickly changing data (the downside is, of course, that reasoning is performed at query time). The tool can reason on approximately seven million triples in seconds, and it scales to data sets with 60 million triples, responding to queries in minutes. It has been used to semantically index 300 million triples from medical literature. SHER tolerates logical inconsistencies in the

data, and it can quickly point you to these inconsistencies in the data and help you clean up inconsistencies before issuing semantic queries. The tool explains (or justifies) why a particular result set is an answer to the query; this explanation is useful for validation by domain experts.

TABLE I

COMPARISION OF INFERENCE ENGINE

Inference Engines	Racer Pro	Jena	Fact	Fact++	Pellet	Hoolet	Jess	F-OWL	KAON2	Sher
Accessibility	Non Free/Closed source	Free/Open source	Free/Open source	Free/Open source	Free Open source & Non Free closed source.	Free/open source	Non free/closed source	Free Open source	Free Open source	Free Open source
Platform	Windows	Windows/Linux	Windows /Linux	Windows/Linux	Windows	Linux	Windows /Linux	Windows	Windows/Linux	Windows /Linux
System of Logic	SHIQ	Many Reasoners	SHIQ	SROIQ (D)	Combination of SHIF & SROIQ(D)	First order logic	Not clear	Frame Logic, Horn Logic	SHIQ	SHIN
Rule Support	Yes(SWRL-not fully support SWRL)	Yes(Own rule format)	No	No	Yes(SWRL-DL safe rules)	Yes(SWRL)	Yes(SWRL)	Yes(SWRL)	Yes(SWRL-DL safe rules	Yes(SWRL-DL safe rules
Reasoning Algorithm	Tableau	Rule based	Tableau	Tableau	Tableau	First order Prover	Rule Based	Tableau	Resolution & datalog	Rule based
Consistency Checking	Yes	Incomplete for OWL-DL	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Interface	DIG,Java ,GUI	DIG, GUI	DIG, Command Line	DIG, Command Line	DIG, Java	Java	Java,GUI, Command Line	Java ,GUI,Command Line	Java, GUI,Command Line	Java, Command Line.
DIG Support.	Yes	Yes	Yes	Yes	Yes	NO	No	Yes	Yes	Yes

III. CONCLUSION

Semantic web will provide us a world where web pages will be having meaningful content & software agents or understanding & using semantic data on the web, inference engine will be required. We have briefly described how various inference engines behave when compared on various measurement criteria. This analysis can be helpful while selecting various technologies while using ontology based inference engine for future research works.

IV. ACKNOWLEDGMENT

We would like to thank to our Director for providing the full cooperation, support and for helpful comments on an earlier draft of this paper. I would like to sincerely thank my research guide Prof. Arjun Rajput for his guidance, help & motivation. Apart from subject, I learnt a lot from him, which I am sure will be helpful in different stages of my life. Finally, this paperwould not have been possible without the confidence, endurance and support of our family. Our family has alwaysbeen a source of inspiration and encouragement.

References

[1]G. Antoniou, F.van Harmelen. A semantic web primer, the
MIT Press

[2] T.Berners-Lee e.a., Semantic Web Tutorial Using, May 20, 2003, www.w3.org/2000/10/swap/doc/

[3]Volker Haarslev and Ralf Moller, "RACER: A Core Inference Engine
For the Semantic Web".

[4] *Pellet: An owl dl reasoner*. B. Parsia, E. Sirin, In:

Proc. International Semantic Web Conference. 2005.

[5] http://owl.man.ac.uk/hoolet/

[6]Youyong Zou, Tim Finin and Harry Chen, "F-OWL: an Inference
Engine for the Semantic Web" supported by DARPA and NSF

[7]http://kaon2.semanticweb.org/

[8]http://www.jessrules.com/

[9] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Edith Schonberg,
Kavitha Srinivas, "Scalable highly expressive reasoner (SHER)"

[10] Jakub Moskal, Northeastern University, Chris Matheus, Vistology,
intelligent software's will take information from pages to pages for reaching a meaningful result. For Inc. Comparison of BaseVISor, Jena and Jess Rule Engines"

[11] Michael Kifer, Georg Lausen, James Wu. "Logical Foundations of Object_Oriented and Frame_Based Languages"http://jena.sourceforge.net/inference.

[12] Jakub Moskal, Northeastern University, Chris Matheus, Vistology, Inc. Comparison of BaseVISor, Jena and Jess Rule Engines"