Gautama – Ontology Editor Based on Nyaya Logic

G.S. Mahalakshmi, T.V. Geetha, Arun Kumar, Dinesh Kumar, and S. Manikandan

Department of Computer Science and Engineering,
Anna University, Chennai, Tamil Nadu, India
{mahalakshmi,tvgeedir}@cs.annauniv.edu,
{arunk.frau2004,dd.jack,write2manikandan}@gmail.com

Abstract. Indian logic based approach of knowledge representation fundamentally classifies the world knowledge into concepts, and relations, both enriched with special qualities. To be more precise, Nyaya Sastra recommends a special categorization of world knowledge which is supposed to be elaborate in tapping the minute details in the defined knowledge units. *Nyaya logics* are a mechanism which defines the concept and relation elements of ontology based on the epistemology of Nyaya-Vaisheshika school of Indian logic. We have already proposed an ontology reference model based on *Nyaya logic*, known as NORM. To develop an ontology using *Nyaya logics*, one should be aware of the syntax and semantics of NORM *rdf*. To overcome the difficulty involved in creating NORM based ontology, in this paper, we propose *Gautama*, a tool for editing the ontology based on *Nyaya logics.* We also discuss the steps for building the ontology for a sample from 'Birds' domain.

Keywords: Indian logic, Nyaya Sastra, NORM, Ontology.

1 Introduction

The Nyaya-Vaisheshika is a self-contained system of philosophy. It proposes a unique categorisation of world knowledge elements [6,9]. Through the epistemological definitions of Nyaya-Vaisheshika system, the treatment of world knowledge elements was very special which contributed to the uniqueness of ontological categorization. The methodology of categorization was inaugurated by Gautama-Akshapada, which consists in enumeration and classification of world knowledge entities into specific categories which were recommended, argued and analysed by the followers of Gautama [6,9].

NORM is the Nyaya based Ontology Reference Model, which defines the standards for constructing ontology, based on the recommendations of the epistemology definitions of Nyaya-Vaisheshika school of Indian philosophy. NORM is organized as a two-layer ontology [8], where the upper layer represents the abstract fundamental knowledge and the lower layer represents the domain knowledge. According to NORM, a node in the ontology is composed of an enriched concept which is related implicitly to its member qualities and explicitly to other peer concepts, by means of relations [11].

A node of Nyaya-Vaisheshika [5,10] based ontology has the following structure (refer Fig. 1). Every concept of the world knowledge shall be thoroughly classified as per NORM structure. The abstract and domain concepts form a strict classification hierarchy. The traits of the top-level concepts are applicable down the hierarchy.

R. Ramanujam and S. Sarukkai (Eds.): ICLA 2009, LNAI 5378, pp. 232–242, 2009.

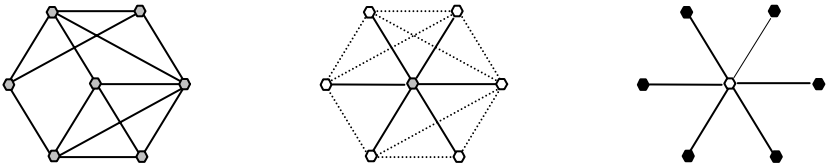

Fig. 1. NORM model for cognitive knowledge representation (a) ontology with concepts as nodes and external relations as edges (b) a concept with qualities as nodes, internal relations as thin edges, tangential relations as dotted edges (c) a quality with values as nodes, grouping relations as edges [8]

Every concept in the NORM model has links to other concepts by external relations (Fig. 1a). A concept is made of qualities or *gunas* [5,10]. In addition, the qualities are bounded to the concept by internal relations. The qualities may also be related to each other, which is depicted as dotted edges (refer Fig. 1.b). Every quality has a set of values. Every value is the substratum of the quality to which it is associated [5,10]. The values are bounded to the qualities by grouping relations (refer Fig. 1c). This model (Fig. 1) is inspired by the various recommendations of classifications of world knowledge according to Nyaya-Vaisheshika. The following section discusses the system of classification of Nyaya Sastra.

2 Nyaya-Vaisheshika System of Classification

According to Nyaya Sastra [4,5,10], every concept is classified into seven categories: substance, quality, action, generality, particularity, inherence and negation. Among these, the substance is of nine kinds: earth, water, light, air, ether, time, space, soul and mind. Every substance is threefold: body, organ and object. The object of light is fourfold: earthly, heavenly, gastric and mineral. Every substance is said to possess some quality. The quality is of twenty-four varieties which in turn possess values (refer Fig. 2).

The permissible action associated with the substance is of five types: upward motion, downward motion, contraction, expansion, and motion. Generality is either more comprehensive or less comprehensive. Particularities are innumerable [4,5,10]. Negation is of four varieties: antecedent negation (or prior negation, destructive negation (or posterior negation, [1]), absolute negation and mutual negation. Out of the nine substances, odour persists only in earth and is inherent. Earth exists in all the seven colors. Air has no color; water is pale-white in color and light is bright-white in color. Air has touch. Water has cold-touch and light has hot-touch. Dimension (or magnitude), distinctness, conjunction and disjunction are present in all the nine substances. Remoteness and Proximity is found in earth, water, light, air and mind. Heaviness or Weight is only in earth and water. Viscidity is present only in the substance, Water [4,5,10].

The detailed structure of a node in Nyaya-Vaisheshika ontology is shown in Fig. 3. The structure incorporates almost all the recommendations of Nyaya-Vaisheshika school along with the detailed definitions of relations at every level, between concepts, between concept and member qualities, between qualities, and between quality and member values. The following section describes the ontology editor, Gautama for editing the world knowledge in the form of Indian logic ontologies.

Color: white, blue, yellow, red, green, brown, varied
Taste: sweet, acid, saline, pungent, astringent, bitter
Odour: fragrant, foul
Touch: cool, hot, lukewarm
Number
Magnitude: atomic, large, long, short
Separateness
Conjunction
Disjunction
Remoteness: spatial, temporal
Proximity: spatial, temporal
Weight
Fluidity: natural, artificial
Viscidity
Sound: articulate, inarticulate
Intellect
Pleasure: ***Remembrance, apprehension***: *Erroneous apprehension, valid apprehension*:
Valid apprehension : perception, inference, analytical knowledge and verbal testimony
Pain
Desire
Aversion
Volition
Merit
Demerit
Tendency

Fig. 2. Ontological Classification of Nyaya-Vaisheshika Qualities

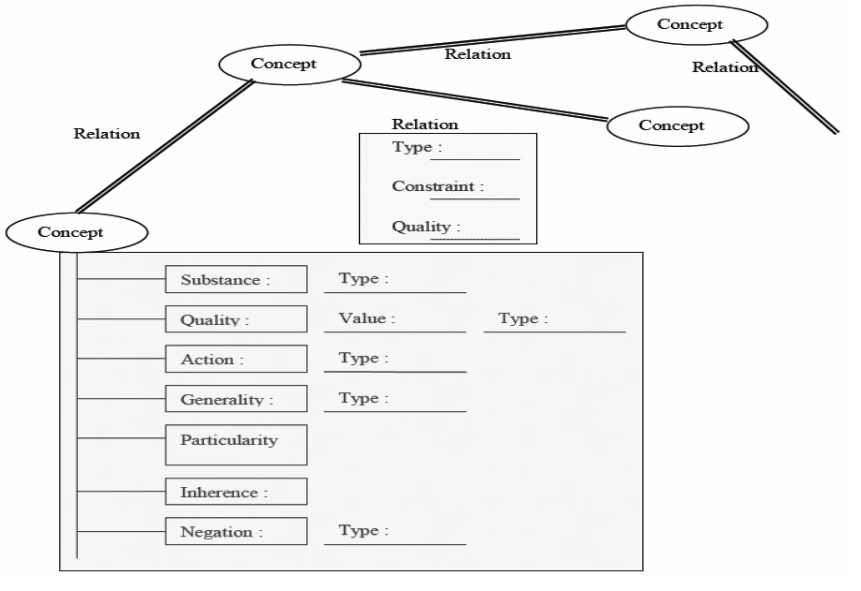

Fig. 3. The node ontology architecture of NORM

3 Gautama – Ontology Editor for Indian Logic

We have developed an ontology editor (refer Fig. 4) known as 'Gautama' for imparting the knowledge in the form of Indian logic. The editor has icons and toolboxes to create / edit the knowledgebase defined under the Nyaya-Vaisheshika system of classification [4,5,10].

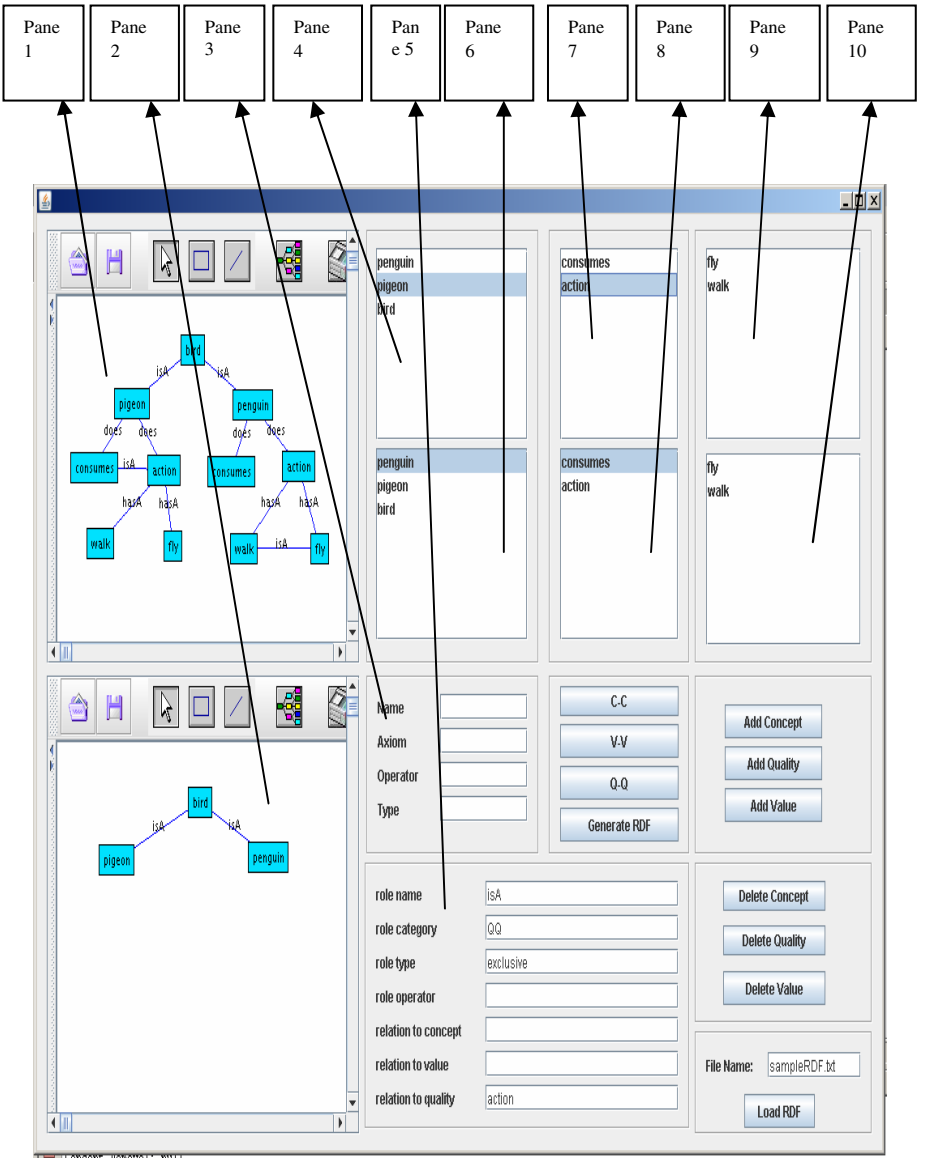

Fig. 4. Gautama – NORM based Ontology editor

The ontology editor has various panes for editing the concepts, qualities and relations, both graphically and through entry forms. Each one of those panes are described as follows:

ILO Visualisation Pane: This pane contains icons to save and print the ontology visualisation created in the top left pane of the editor. In addition, drawing icons have also been provided.

Concepts Visualisation Pane: This pane is similar to ILO Visualisation Pane, except that, here, only the concept hierarchy in the ontology is visualised.

Nodes Entry Pane: This pane provides controls for entering information about the nodes that are yet to be created to become part of the ontology. C-C denotes concept-concept; V-V denotes value-value and Q-Q denotes quality-quality. There are enough command buttons to add concepts, qualities and values. Using these, the concept definitions shall be created. The 'Generate RDF' button helps in generation of Resource description format of the underlying ontology.

Relations Entry Pane: The purpose of this pane is identical to Nodes entry Pane. Here, 'roles' shall be created as part of the ontology. NORM recommends various relations (refer Fig.1), therefore, this pane has provisions for creating relations at all levels. To the extreme right, is the command buttons for 'deletion' services. Using these buttons, concept / quality / value shall be deleted from the ontology. Alternatively, one can also load a pre-existing RDF through 'load rdf' button to have the ontology loaded into the memory at once.

Concepts list Pane: This pane lists all the concepts available in the ontology with specialised concepts first displayed, followed by the generalised concepts. (Please note from Fig. 4.4. that, 'penguin' and 'pigeon' are displayed before 'bird'. There are two concepts list pane, primary and secondary.

Quality List Pane: This pane lists all the qualities available for the selected concept in the adjacent left pane. This is divided into primary and secondary panes.

Value list pane: This pane lists all the values available for the selected quality in the adjacent left quality list pane. This is divided into Primary and secondary panes.

If two concepts are related to each other, one concept and its member qualities, member values shall be seen in the primary pane. Simultaneously, the other concept and its member qualities, member values shall be seen in the secondary pane. To facilitate the recording of knowledge in RDF (resource description format), appropriate tags have been defined, with a start tag and corresponding end tag with the item described in between. The following are the various tags defined for the RDF of Indian logic ontology generated by Gautama.

- <rdf:concept> - This tag is used to declare a concept prior and after its definition.
- <rdf:name> - This tag is used to declare the name of a concept / quality / relation.
- <rdf:desc> - This tag is used to create descriptions or definitions for a particular concept.
- <rdf:axiom> - This tag is used to create concept axioms.
- <rdf:quality> - This tag is used to create member qualities for a given concept.
- <rdf:type> - This tag is used to declare the type of a concept / quality / relation.
- <rdf:role> - This tag is used to declare the role of a concept / quality.
- <rdf:category> - This tag is used to declare the category of relation like external, internal, tangential or grouping.
- <rdf:operator> - This tag is used to declare the logical operators like and, or while creating the concept axioms of the ontology.

The sample RDF generated for a simple ontology for 'birds' domain is given in Fig. 5. The facilities for interacting with the knowledgebase are generally done through knowledge representation languages. *NORM model* for knowledge representation involves *Nyaya Description language (NDL)*, the set of commands used for defining the units of knowledge base.

```
<rdf:concept>
        <rdf:name>bird</rdf:name>
</rdf:concept>

<rdf:concept>
        <rdf:name>pigeon</rdf:name>
        <rdf:axiom>bird</rdf:axiom>
        <rdf:desc>
                <rdf:quality>
                        <rdf:name>action</rdf:name>
                        <rdf:type>exclusive</rdf:type>
                        <rdf:operator>and</rdf:operator>
                        <rdf:value>
                                <rdf:name>walk</rdf:name>
                                <rdf:operator>and</rdf:operator>
                        </rdf:value>
                        <rdf:value>
                                <rdf:name>fly</rdf:name>
                                <rdf:operator>inclusive</rdf:operator>
                                <rdf:role>
                                        <rdf:name>isa</rdf:name>
                                        <rdf:category>VVrelationship</rdf:cate
                                        <rdf:type>symmetric</rdf:type>
                                        <rdf:relToValue>walk</rdf:relToValue>
                                        <rdf:operator>and</rdf:operator>
                                </rdf:role>
                        </rdf:value>
                        <rdf:role>
                                <rdf:name>hasA</rdf:name>
                                <rdf:category>QVrelationship</rdf:category>
                                <rdf:type>transitive</rdf:type>
                                <rdf:relToValue>walk</rdf:relToValue>
                                <rdf:operator>or</rdf:operator>
                        </rdf:role>
                        <rdf:role>
                                <rdf:name>hasA</rdf:name>
                                <rdf:category>QVrelationship</rdf:category>
                                <rdf:type>transitive</rdf:type>
                                <rdf:relToValue>fly</rdf:relToValue>
                                <rdf:operator>and</rdf:operator>
                        </rdf:role>
                        <rdf:role>
                                <rdf:name>isA</rdf:name>
                                <rdf:category>QCrelationship</rdf:category>
                                <rdf:type>transitive</rdf:type>
                                <rdf:relToConcept>pigeon</rdf:relToConcept>
```

Fig. 5. NORM RDF– 'penguin' and 'pigeon' example

```
                    <rdf:operator>and</rdf:operator>
                </rdf:role>
            </rdf:quality>
            <rdf:quality>
                <rdf:name>consumes</rdf:name>
                <rdf:operator>and</rdf:operator>
                <rdf:type>exclusive</rdf:type>
                <rdf:role>
                    <rdf:name>isA</rdf:name>
                    <rdf:category>QQrelationship</rdf:category>
                    <rdf:type>transitive</rdf:type>
                    <rdf:relToQuality>action</rdf:relToQuality>
                    <rdf:operator>and</rdf:operator>
                </rdf:role>
                <rdf:role>
                    <rdf:name>isA</rdf:name>
                    <rdf:category>QCrelationship</rdf:category>
                    <rdf:type>transitive</rdf:type>
                    <rdf:relToConcept>pigeon</rdf:relToConcept>
                    <rdf:operator>and</rdf:operator>
                </rdf:role>
            </rdf:quality>
            <rdf:role>
                <rdf:name>isA</rdf:name>
                <rdf:category>CCrelationship</rdf:category>
                <rdf:type>transitive</rdf:type>
                <rdf:relToConcept>bird</rdf:relToConcept>
                <rdf:operator>and</rdf:operator>
            </rdf:role>
        </rdf:desc>
    </rdf:concept>
```

Fig. 5. (*continued*)

The knowledge representation language [1,7], is classified into concept/relationship definition language (CRDL), concept/relation manipulation language (CRML) and a set of editing commands and a query language. This knowledge representation language can be further used to define, manipulate and query the various levels of knowledge. CN refers to Concept name, QN refers to Quality Name, V – Quality value (Ex: color – Indigo: quality: color, value: Indigo) RN refers to Role name, I refer to Instance and Rdesc refers to Role descriptions. The CRDL constitutes the commands for defining the concepts, instances and relationships. Top and Bottom concepts are assumed by the system as default. The concept definitions have been recognized and the knowledge hierarchy is built. Therefore, using CRDL, the user can build the knowledge base right from scratch. Concepts can be linked to one another through relations where relations can be is-a, owns, part-of and uses. Relations and actions can also be defined between concept and quality. Instances of concepts can also be defined using CRDL. Following the above norms of definition of knowledge representation languages (as description logic commands), here, we define the sample Nyaya logic commands which are listed in Table 1.

Table 1. Commands for querying with Gautama

CRDL	**CRML**
define-concept<*CN, Level*> define-concept-axiom<*CN,Cdesc*> disjoint-concept<*C1,C2*> define-role-axiom<*RN, Rdesc*> disjoint-role<*R1,R2*> define-concept-role<*RN,C1,C2*> define-concept-qualities<*CN, (QM,Qman.List) / (QO,Qopt.List) / (QE,Qexceptional.List) / (QX,Qexclusive.List)*> define-quality-values<*CN,QN,V1….Vn*> define-role-quality <*RN,CN, Qreflexive.List / Qsymmetric.List / Qassymmetric.List / Qantisymmetric.List / Qtransitive.List / Qdirect.List / Qindirect.List / Qexclusive.List*> define-quality-role<*RNreflexive.List / RNasymmetric.List / RNsymmetric.List / RNantisymmetric.List / RNtransitive.List / RNdirect.List / RNindirect.List / RNexclusive.List, CN,QN*>	insert-quality<*QN*> delete-quality<*QN*> insert-values<*QN,V1….Vn*> delete-values<*QN,V1….Vn*> delete-concept<*CN*> delete-instance<*I*> update-instance<*I,Cnold,Cnnew*> delete-role-filler<*I1,I2,RN*> update-role-filler<*I1,I2,Rnold,Rnnew*> delete-role<*RN*> insert-role<*RN*> delete-concept-quality<*CN,QN*> delete-quality-value<*CN,QN,VInvariableConcomitance.List / VExclusive.List / VInvariableConcomitance.List / VDirect.List*> insert-quality-value<*CN,QN,VInvariableConcomitance .List / VExclusive.List / VInvariableConcomitance.List / VDirect.List*> update-quality-value<*CN,QN,Vold,Vnew*>
Query language	**Query language**
concept-satisfiable<*CN*> concept-subsumes<*C1,C2*> concept-disjoint<*C1,C2*> chk-concept<*CN*> concept-atomic<*CN*> concept-ancestors<*CN*> concept-descendants<*CN*> super-concept<*CN*> sub-concept<*CN*> chk-concept-related<*C1,C2*> chk-concept-related<*C1,C2,RN*> chk-concept-related<*C1,C2,RNreflexive.List / RNasymmetric.List / RNsymmetric.List / RNantisymmetric.List / RNtransitive.List / RNdirect.List, RNindirect.List / RNexclusive.List* > chk-quality<*QN*> chk-concept-quality<*CN,QN*> all-qualities	retrieve-direct-concepts<*I*> retrieve-indirect-concepts<*I*> retrieve-concept-fillers<*RN,C1*> all-concepts<*I*> retrieve-qualities<*CN*> retrieve-quality-value<*CN,QInvariableConcomitance / QExclusive / QExceptional*> retrieve-quality-value<*CN,QDirect*> chk-instance<*I*> chk-instance-type<*I,CN*> chk-instance-related<*I1,I2*> retrieve-direct-instances<*I*> retrieve-indirect-instances<*I*> retrieve-instance-fillers<*RN,I1*> all-instances<*CN*> retrieve-related-instances<*RN*> *retrieve-quality-value<*I,QN*> chk-role<*RN*> all-roles role-descendants<*CN*> role-ancestors<*CN*>

The CRML provides necessary commands for deleting and updating of concepts and associated relations in the knowledge hierarchy. The query language supports querying the classification hierarchy and to summarize the results of queries. The TAML commands have been utilized for the management of Tbox and Abox. The system shell is managed by create and use taxonomy which are used primarily for mounting and dismounting the Tbox and Aboxes. Upon commit, the information contained in the classification hierarchy is stored in a separate file, which also records every inferencing performed by the system. In addition the system provides concept and instance dictionary files, which summarises the total number of knowledge units present in the classification hierarchy. Using CRML, the ontology shall be updated or modified. The concepts and the relation between concepts can be manipulated using the commands of CRML.

The CRDL and CRML commands are used only during the creation of ontology by end users. To be more user-friendly, 'Gautama', the ontology editor provides built-in facilities for ontology creation and updation services. The query language shall be used with the RDF generated by Gautama, to query about various parts of the ontology. Here, we discuss few commands of the query services.

- Concept-satisfiable – This takes a concept name as the parameter and checks whether the addition of the concept will not violate the ontology definitions that exist prior to the execution of this command.
- Concept-subsumes – This takes two concepts as input, and checks whether the first concept subsumes the second concept. This is one of the famous reasoning service provided by any ontology-based reasoner.
- Concept ancestors and Concept-descendants – These commands list the ancestral / descending concepts in the ontology hierarchy. Role-ancestors and Role-descendants also have similar purpose.
- Sub-concept, Super-concept – These commands retrieve the child nodes or parent nodes of the parametric concept from the ontology hierarchy.
- Chk-concept-related – This command has three variations. It either checks whether a concept is related to another concept, through a particular relation name or through a particular set of relation categories.
- Chk-quality – This command checks the entire ontology hierarchy to check if the required quality is available in the ontology.
- Chk-concept-quality – This command checks the entire ontology hierarchy to check if the particular concept has the required quality.
- All-concepts, all-qualities, all-roles, all-instances – These commands just lists all the concepts, qualities, roles or instances available in the ontology.
- Retrieve-direct-concepts, retrieve-indirect-concepts – The first commands take an instance as input, and retrieve all the directly related concepts to those instances; The second command take the instance as input and retrieves all the second and higher degree concepts related to those instances. For example, if 'TooToo' is an instance of penguin, the first command may retrieve 'penguin' as the result; the second command will retrieve all the ancestors of penguin which are conceptually related to penguin. Retrieve-direct-instances, retrieve-indirect-instances also serve the same purpose.

4 Related Work

This paper proposed the ontology editor based on Indian logic based knowledge representation system. Using this editor, one can carefully handcraft the ontology based on Indian logic in the required domain. However, there are other noteworthy projects existing in the knowledge representation arena. Cyc, WordNet, Concept-Net and Mind-Net are to name a few.

Cyc is an artificial intelligence project [3] that attempts to assemble a comprehensive ontology and database of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning. The Cyc system is made up of three distinct components, all of which are crucial to the machine learning process: the knowledge base (KB), the inference engine, and the natural language system. The Cyc inference engine is responsible for using information in the KB to determine the truth of a sentence and, if necessary, find provably correct variable bindings. The natural language component of the system consists of a lexicon, and parsing and generation subsystems. The lexicon is a component of the knowledge base that maps words and phrases to Cyc concepts.

WordNet is a large lexical database [2] of English, where, Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet's structure makes it a useful tool for computational linguistics and natural language processing. Its design is inspired by current psycholinguistic and computational theories of human lexical memory.

ConceptNet is a freely available [13] commonsense knowledge base and natural-language-processing toolkit built at MIT. The ConceptNet knowledge base is a semantic network of commonsense knowledge encompassing the spatial, physical, social, temporal, and psychological aspects of everyday life. Whereas similar large-scale semantic knowledgebases like Cyc and WordNet are carefully handcrafted, ConceptNet is generated automatically from World Wide Web.

ConceptNet is a unique resource in that it captures a wide range of commonsense concepts and relations, yet this knowledge is structured as a simple, easy-to-use semantic network, like WordNet. While ConceptNet still supports query expansion and determining semantic similarity, its focus on concepts-rather-than-words, its more diverse relational ontology, and its emphasis on informal conceptual-connectedness over formal linguistic-rigor allow it to go beyond WordNet to make practical, context-oriented, commonsense inferences over real-world texts.

A MindNet is a collection of semantic relations that is automatically extracted from text data using a broad coverage parser [12]. MindNets are produced by a fully automatic process that takes the input text, sentence-breaks it, parses each sentence to build a semantic dependency graph (Logical Form), aggregates these individual graphs into a single large graph, and then assigns probabilistic weights to subgraphs based on their frequency in the corpus as a whole. The project also encompasses a number of mechanisms for searching, sorting, and measuring the similarity of paths in a MindNet.

'Gautama' proposed in this paper is not automatic, i.e. it does not harvest ontological entities automatically from the text corpora or web, instead, it is a first step in the

design of an ontology editor based on Indian logic, and therefore, presently it is only handcrafted to serve the purpose. In future, adapting more ideas of building the ontology from Indian philosophy would strengthen the outcome of the ontology editor.

5 Conclusion

This paper proposed the overview of Gautama, a tool for editing the world knowledge elements into ontology based on Indian logic. The ontology followed the guidelines of NORM (Nyaya Ontology reference model) based ontological standards which is built on the epistemological recommendations of Nyaya-Vaisheshika school of Indian philosophy, for defining the knowledge units of the ontology. We hope, this tool, facilitates easy creation of Indian logic based ontologies and thereby promotes the wide study of Indian logic in the ever green field of ontological and philosophical research.

References

1. Aghila, G., Mahalakshmi, G.S., Geetha, T.V.: KRIL – A Knowledge Representation System based on Nyaya Shastra using Extended Description Logics. VIVEK journal 15(3), 3–18 (2003)
2. Fellbaum, C.: WordNet - An Electronic Lexical Database, with a preface by George Miller (May 1998)
3. Matuszek, C., Witbrock, M., Kahlert, R.C., Cabral, J., Schneider, D., Shah, P., Lenat, D.: Searching for Common Sense: Populating Cyc from the Web. In: Proceedings of the Twentieth National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania (July 2005)
4. Vidyabhusana, S.C.: Gautama, The Nyaya Sutras. In: Sinha, N.L. (ed.) Sacred Book of the Hindus, Allahabad (1930); Reprinted in Motilal Banarsidass, Delhi (1990)
5. Ballantyne, J.R.: Lectures on Nyaya Philosophy-Embracing the text of Tarka Samgraha. Presbyterian Mission Press, Allahabad (1849), http://books.google.com
6. Ganeri, J.: Indian Logic: A Reader. Routledge (2001)
7. Mahalakshmi, G.S., Anupama, N., Chitra, R., Geetha, T.V.: Deepika – A Non-Monotonic Reasoning System Based On Indian Logic. In: International Conference on Soft Computing techniques in engineering, SOFTECH – 2007, Avinashilingam University for Women, Coimbatore, India, pp. 470–476 (January 2007)
8. Mahalakshmi, G.S., Geetha, T.V.: Reasoning and Evolution of consistent ontologies using NORM. In: IJAI, Indian Society for Development and Environment Research (ISDER), vol. 2(S09), pp. 77–94 (Spring, 2009); ISSN 0974-0635
9. Vidyabhusana, S.C.: A History of Indian Logic – Ancient, Medieaeval and Modern Schools, p. 84. Motilal Banarsidass Publishers Private Ltd., Delhi (1988); ISBN:81-208-0565-8
10. Swami Virupakshananda, Tarka Samgraha, Sri Ramakrishna Math, Madras (1994)
11. Wada, T.: Invariable Concomitance in Navya-Nyaya, Sri Garib Dass Oriental Series No. 101. Indological and Oriental Publishers, New Delhi (1990)
12. Vanderwende, L., Kacmarcik, G., Suzuki, H., Menezes, A.: MindNet: An Automatically-Created Lexical Resource. In: Proceedings of HLT/EMNLP 2005 Interactive Demonstrations, Vancouver, British Columbia, Canada (October 2005)
13. Concept Net, `http://www.conceptnet.org`