GNOWSYS: A System for Semantic Computing

Nagarjuna G.

Homi Bhabha Centre for Science Education,
Tata Institute of Fundamental Research,
Mumbai India 400088,
nagarjun@gnowledge.org,
http://web.gnowledge.org

Abstract. The purpose of this essay is to introduce a heterogeneous computing environment with a possibility of semantic and formal annotations, covering the motivation, architecture, and functionality of the system called GNOWSYS (Gnowledge Networking and Organizing System). GNOWSYS is a system to specify, publish and query about multiple logics, ontologies, and epistemologies. Its kernel includes three semantic layers with increasing order of semantic specification, three groups of component classes for storing objects of various complexity in the knowledge base, and three levels of generality with objects belonging to tokens, types and metatypes. It has component classes for representing both declarative and procedural knowledge, and the latter specification is executable enabling semantic and visual computing. It is specially made for publishing vocabularies, propositions, ontologies, complex systems, web services, with or without formal annotation. Its architecture is not frozen, and is still actively taking shape. Hence a purpose of this communication is also a request for comments.

1 Introduction and Motivation

This essay is not about suggesting any specific original solutions to formal knowledge representation problems. The attempt is to draw from the current wisdom of logics and knowledge representation, to build a comprehensive, flexible, and extendable conceptual schemes of KR and develop a server that stores, modifies, and distributes across the Internet such schemes. Its main objective is to publish any knowledge representation scheme with or without constraints in an object oriented distributed database. It may be useful to inform the readers that the author (yours truly) of this essay, and the application GNOWSYS, is not a qualified computer scientist, but an epistemologist venturing to develop a computing solution to solve the problem of modeling conceptual dynamics in the context of learning and discovery. Also this paper does not serve the purpose of demonstrating that the current proposal is the better or distinctive or more functional solution to the problem of knowledge representation than the other methodologies. Such a task will be met at a later day. It is therefore thought that at least the motivation, dubbed as the rationale, for developing GNOWSYS

be made explicit at the outset. It is less than a concept paper, since the formal properties of the system are not specified rigorously in this composition.

The initial motivation for making GNOWSYS was to develop a community portal (gnowledge.org) that meets the following three objectives: (a) to draw concept graphs and inferences from a knowledge base; (b) the system represents the conceptual scheme of an expert's knowledge in a given area, and reports the matches and mismatches of a learner's conceptual schemes in the process of acquiring knowledge; (c) each node in the expert's conceptual scheme refers to one or more learnable resources like lessons, images, videos, figures, and other such digitally encoded and accessible resources available on the Internet. An obvious application of such a tool is a sophisticated knowledge base for elearning. It may be noted that objectives (a) and (c) are several times easier than the objective (b), since the problem is not merely about static representation of knowledge of an expert and a novice, but to model the dynamics of conceptual change in the process of learning and discovery, which are known to be complex. To fulfil the objective (b) it is required to have a framework to store expert's knowledge.

Some aspects of the problems were realized in the early days of AI by Herbert Simon in his often cited work *Sciences of the Artificial*[1]. Simon's pioneering contributions to the theory and applications of AI, and Marvin Minsky's idea of frames[2], continue to play a pivotal role in solving the knowledge representation problems even today. This work grows out of and draws from their wisdom. Recently, several researchers used concept maps and semantic networks to enhance conceptual learning[3–6] in the context of education. Most of these tools, suggested in the above citations, are essentially drawing tools, and the maps drawn by the students or experts could not be stored in an accessible knowledge base. Graphs were stored as separate files, making reusing a component of a graph difficult. Since the graphs were encoded in a format that is internal to the applications, it is difficult to compare two concept graphs, made by different applications, and remain unshareable. The objective of matching and mismatching of concept graphs of two or more agents could not be achieved without a sharable encoding. While designing GNOWSYS these problems were kept in mind, so the graphs generated by other applications could be shared and published by the system.

The epistemological presuppositions (the working hypotheses) of this undertaking are: (1) a cognitive agent understands a new concept when relations are established between the preexisting concepts with the new concept[7, 8, 3]; (2) to educate a person therefore is to facilitate the process of establishing the relevant relations between concepts so as to match that of an expert; (3) learning therefore involves restructuring of conceptual schemes; and (4) misunderstandings are due to mismatching between conceptual schemes between the agents. According to this approach no concept gains any meaning independent of its relations with other concepts. Thus, meaning of a concept is the network it forms with others.

The sense of understanding used here is stronger since we are seeking that the relations between the concepts be made explicit. For example, when we look at

a tree, and recognize that it is indeed a tree, is also understanding of a sort, but it is implicit. Also the term 'education' is used here in a strong sense. This does not cover the various forms of behavioral mastery, such as skills, that children learn and execute without any explicit understanding. One of the challenges in education, particularly of exact sciences, is to gradually train learners towards more and more explicit forms of representation. Formal sciences like theoretical physics, mathematics and logic, for example, are domains of discourse where procedural knowledge is declaratively stated and declarative knowledge is procedurally stated reaching a highest degree of explicit knowledge representation[9]. If a system could model the process of learning science beginning from folk-lore to formal knowledge, the system is required to capture implicitly several entities in the knowledge base with entities loosely and sometimes inconsistently held with other entities, and with several degrees of modalities of expressing a proposition.

Thomas Kuhn's *Structure of Scientific Revolutions*[10] had a major impact on researchers studying cognitive development, and several research programs are guided by this work for studying conceptual change during ontogeny[11]. Conceptual scheme, by which Kuhn also meant the taxonomy (ontology) of a scientific theory, became a very important mode of analysis for modeling conceptual change and inter-theoretic semantic relations (see e.g., [12]). These problems are not new, but recent rise of interest, among the computer science community, in semantic web[13, 14], brought computational, semantic and logic oriented problems of ontologies a new life.

The context of learning (cognitive development) and discovery (history of science), is possibly the most challenging problem for AI, because a learner or a scientist respectively, in the course of development, not only changes from one conceptual scheme to another, but also often harbors contradictory systems of beliefs, often due to implicit and unknown parameters of knowledge. Automated real cognitive agents are not like the artificial systems that maintain rigid consistency and require for their functionality explicit frameworks. Thus the problem is to model the transformation from a loosely and often inconsistently structured and implicit epistemologies and ontologies to tightly integrated, explicitly consistent forms.

Further, a model developed say for biology may not be appropriate for mathematics. A model chosen for representing common sense may not be good for science, and a model for science may not be good for formal sciences like logic and mathematics. And the problem becomes even more difficult when we think of mapping different folk-lore of divergent cultures across the globe. This suggests that we need a system that could express multiple epistemologies, ontologies and logics.

Even for exact sciences, it is not an easy task to get together all the pool of predicates required for even a single domain of knowledge. Added to the problem is that an ontology made by one school of thought will sure be contested by another school of thought. Controversies on issues related to epistemology, ontological commitments etc., may make none of the models successful or complete.

An awareness of this seemingly impossible and utterly difficult task of modeling the process of learning and using this to build elearning applications by employing the current wisdom of cognitive and computer science suggests that the project has all seeds of failure inbuilt at the very core, therefore doomed to fail.

Is there a way out? I think so, and it is the objective of this essay to suggest a proposal in the form of GNOWSYS. This proposal is not described in a formal language, essentially because its shape is not final, without a final shape it is difficult to perform a discourse in a formal language. I thought it is wise to seek comments from experts at this very stage before freezing the final architecture. It is likely that the model outlined here for GNOWSYS has inherent, known or unknown limitations.

A comment here on ACT-R system developed by J.R. Anderson and his colleagues is appropriate[15]. ACT-R is one striking comprehensive architecture of cognition with various levels of simulating the process of cognition, problem solving and a test bed to evaluate theories of cognitive development with a comprehensive theory of memory. GNOWSYS system, unlike ACT-R, and is not based on the current information processing model of cognition. I have argued elsewhere that semantic memory, as against episodic memory, is not stored in the body of a cognitive agent, and semantics are a function of inter-agent communication[9]. Approach of GNOWSYS is to create a collaborative and communicating knowledge base capable of storing and exchanging propositions, conceptual schemes, behavioral schemes and belief systems with other agents, both human and artificial.

The popular Cyc project, and the recently released OpenCyc (a free software version of Cyc technology), was built as a huge knowledge base containing 2.5 million facts and rules capturing common sense knowledge[16]. GNOWSYS supports such knowledge representation with the difference that the knowledge is stored in a hybrid form (as frames, objects, agents, structures and processes). GNOWSYS kernel does not contain any inference engine, because GNOWSYS by design has provision for expressing executable external libraries implemented in any computer language, thus can support, in principle, any inference implementation through its procedural objects and RPC interphase, leaving the focus on building a high performance knowledge base.

The following section is a non-formal description of the architecture of GNOWSYS. The latter sections explicate how the system can be used for semantic computing applications.

2 The Architecture

The kernel of GNOWSYS is structured to accommodate different dimensions of knowledge representation: semantic, complexity, and generality dimensions, in addition to a communicating interphase. Figure 1 is one way of representing the architecture.

2.1 Semantic Dimension

The system is intended to store propositions like, "All gods are present everywhere.", "All gods are present somewhere.", "Some gods are present everywhere.", "No gods are present somewhere.", "Some gods are present somewhere.", etc. These propositions surely cannot be held together in a single belief system. But they should be stored and talked about, just as such propositions exist in this composition. One clear way to achieve the possibility of storing multiple epistemologies and ontologies of this kind is to build a database system that has distinct layers in its kernel. It should be possible to have a layer where ground level propositions exist without 'interaction' between them, hence even contradictory or false propositions can be stored. This layer therefore will not have semantic constraints. Another layer that takes the propositional 'atoms' and combines them with semantic constraints to eliminate contradictions so as to build a system that is *implicitly* consistent. But even at this layer, it should be possible to combine propositions to form a system without checking for consistency. After all we harbor inconsistent belief systems and we need to engage in a discourse about them. This requires that a second layer of the kernel that builds the systems contains only constraints of logical connectives (like 'and', 'or', 'implies' etc.) without checking for consistency. The kernel therefore needs another layer where consistency is imposed.

Based on these requirements, I propose, we build a knowledge base with component classes that can instantiate objects into three different layers:

Layer 1 of Well Formed Formulae (WFF): A comprehensive, flexible, and extendable logical core layer to store well formed formulae that are completely neutral to epistemologies and ontologies.

Layer 2 of Implicitly Structured Systems (ISS): A mechanism to compose the well formed formulae expressed in the above layers using logical connectives, quantifiers, and modalities, propositional attitudes etc. Consistency is implicit, but not explicitly imposed by the system.

Layer 3 of Consistently Structured Systems (CSS): Similar to the second layer, but with explicitly imposed validity constraints resulting consistent systems. All the consistent systems are defined only in relation to an explicit set of semantic rules internal to the system.

By composing the elements from Layer 1 while expressing the elements of Layers 2 and 3, we can represent belief systems with varying degrees of consistency without any conflicting interaction between them, for each system can be instantiated as independent objects in the knowledge base. Several such systems with independent semantics can be stored in a single knowledge base or in a collection of distributed systems over the Internet to form a semantic grid, to express multiple epistemologies and ontologies. A meta-discourse about the semantic matching or mismatching between structures can be made possible by using them as the terms and the predicates from upper ontology (Metatype level).

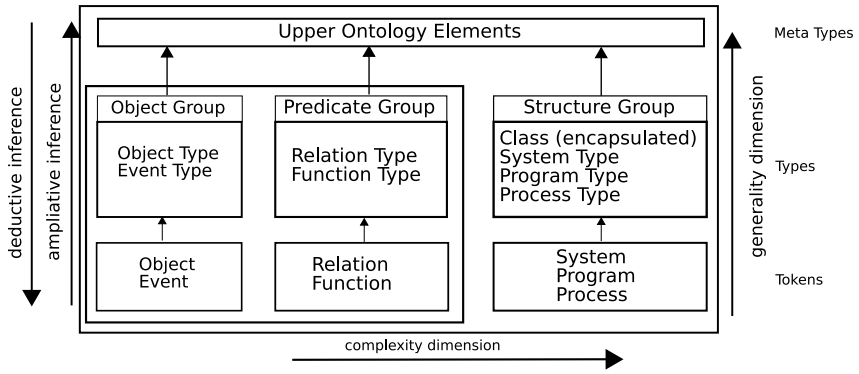

Fig. 1. A diagram representing the architecture of GNOWSYS kernel showing the component classes, and complexity, and generality dimensions.

2.2 Complexity Dimension

The kernel is designed to provide support to persistently store very simple atoms of knowledge representation like *terms*, *predicates* and very complex propositional systems like *arguments*, *rules*, *axiomatic systems*, loosely held *paragraphs*, and more *complex structured and consistent compositions*. All the component classes in GNOWSYS are classified according to complexity into three groups, where the first two groups are used to express all possible well formed formulae permissible in a first order logic.

Terms: 'Object', 'Object Type' for declarative knowledge, 'Event', 'Event Type', for temporal objects, and 'Meta Types' for expressing upper ontology. The objects in this group are essentially any thing about which the knowledge engineer intends to express and store in the knowledge base, i.e., they are the objects of discourse. The instances of these component classes can be stored with or without expressing 'instance of' or 'sub-class of' relations among them. They are also designed to borrow monadic relations (attributes) from the predicate group to characterize them.

Predicates: This group consists of 'Relation', and 'Relation Type' for expressing declarative knowledge, and 'Function' and 'Function Type' for expressing procedural (behavioral) knowledge. This group is to express qualitative and quantitative relations among the various instances stored in the knowledge base. While instantiating the predicates can be characterized by their logical properties of relations, quantifiers and cardinality as monadic predicates of these predicate objects. An additional comment of how functions can be represented in GNOWSYS is warranted. Though it is possible to explicitly define functions from very primitive 'atoms' (as done in ACT-R) GNOWSYS provides an implicit way of instantiating surrogates of functions available in the operating system or from the grid of GNOWSYS servers any where on the

Internet. More on this unique feature later. Another notable point regarding predicates is that the relations are not only expressed by specifying foreign keys on the argument objects, but also as independently reified objects.

The object and predicate group together provide the basic vocabulary and ground assertions. They constitute the set of well formed formulae in the knowledge base, with quantifiers, and other possible kinds of formal annotations.

Structures: 'System', 'Encapsulated Class', 'Program', and 'Process', are base classes for complex structures, which can be combined iteratively to produce more complex systems. The component class 'System' is to store in the knowledge base a set of propositions composed into ontologies, axiomatic systems, complex systems like say a human body, an artifact like a vehicle etc., with or without consistency check. An 'Encapsulated Class' is to compose declarative and behavioral objects in a flexible way to build classes. A 'Program' is not only to store the logic of any complete program or a component class, composed from the already available behavioral instances in the knowledge base with built-in connectives (conditions, and loops), but also execute them as web services. A 'Process' is to structure temporal objects with sequence, concurrency, synchronous or asynchronous specifications. Using these classes a wide variety of domain specific systems are specifiable either formally or informally. These classes can be used to flexibly design in more than one design architecture, like functional, structural, object or agent oriented ways. As mentioned earlier these structures are part of the higher layers, which can reuse the ground assertions as well as other simpler structures instantiated earlier, but they do not interact with each other unless explicitly specified. This is to ensure that inconsistencies in one structure do not propagate to other structures stored in the knowledge base. This is how GNOWSYS is intended to store multiple ontologies, epistemologies and logics.

Every object in the database keeps the *neighborhood* information, such as its super-class, sub-class, instance-of, and other relations, in which the object has a role, in the form of predicates. This feature makes computation of drawing graphs and inferences, on the one hand, and dependency and navigation paths on the other hand very easy. Since references about an object are present on other objects in the network, an accidentally lost object could be reconstructed. These points justify an element of redundancy in the database. All the data and metadata is indexed in a central catalogue making query and locating resources efficient.

2.3 Generality Dimension

Alternatively the GNOWSYS kernel can also be viewed in the form of three semantically significant levels: metatype, type and token levels. All the two type level constructors help a knowledge engineer construct (specify) a model (the structure of the cognitive system), The epistemic value of this modeling type

layer is *consistency*, since these are conceptual in nature. All the token level constructors help store the data about a given situation. The epistemic value of this token layer is *truth*. Though contradictory propositions can be stored in the system at the ground layer, while building the structures additionally and explicitly specified semantic constraints prevent contradiction. The connection between the levels is made by the inference layer, which does the validity checking, to deduce consequences that are not fed into the knowledge base explicitly, to hypothesize either abductively or by other ampliative means. This will add new inferred elements, both propositions and concepts, to the system. The inference layer is optionally kept out of the system, for it should be possible to conduct semantic discourse about an instantiated structure. The top metatype level can serve the purpose of expressing and storing what is now popularly known as upper ontologies and the relations among them[17, 18].

2.4 Flexible Knowledge Representation Model

As mentioned above, one of the guiding principle in making GNOWSYS is to make the kernel support flexible modeling. As a result the primitive constructors of the systems are sufficiently general and are less imposing. The number of constructors are kept as low as possible, but at the same time the approach is not minimalist in kind, since such an approach may make it very abstract, difficult and inconvenient to use. Keeping in mind the motivations mentioned in the beginning the system is suitable for building systems with high degree of expressivity than closed, strictly constrained formal systems, though the latter possibility is not ruled out. Thus from unstructured data (WFF) to semistructured (ISS) and structured data (CSS) can be constructed and stored. Also since multiple ontologies can be stored in the knowledge base, the relations between them can also be expressed using various semantic matching proposals (For example[19]). This is very important for meeting the objective (b). A teacher or an automatic evaluation engine could specify the semantic matching relations between different conceptual schemes and send the reports to the students. Since matching relation is a kind of analogy, more expressive analogical reasoning techniques can be employed.

The model draws from various well known models of knowledge representation and tries to support expressibility of the common wisdom of the area. In this regard, special mention be made on the models that are taken more seriously for the core structure: standard first order logic for declarative knowledge, UML for representing object oriented modeling[20], Petri Net Markup Language[21] for procedural representation and processes, and standard upper ontology for discourse of highly abstract meta level[17].

A special mention must be made in this regard on a resource that came in handy for grasping the wisdom: John Sowa's comprehensive work on *Knowledge Representation*[18]. The attempt is to meet Sowa's challenge of knowledge soup ("the fluid, loosely organized, dynamically changing contents of the human mind"), on the one hand and reach dialectically the procedurally represented

declarative knowledge, and declaratively represented procedural knowledge on the other.

This flexible architecture of GNOWSYS also includes data exchange modules that can support interchange of knowledge base into various standards used in the industry and academia. Notable among them are: Common Logic (CL)[22], Concept Graphs (CG)[23, 18], KIF (Knowledge Interchange Format)[24], Web Ontology Language (OWL)[25, 26], Topic Maps (XTM)[27], and PNML (Petri Net Markup Language)[21]. OWL and XTM support is already implemented and support for others is under development. UML, CG and Petri Nets are very convenient modes of graphical representation of knowledge, and so are of special significance to the pyGTK[28] based GUI for GNOWSYS called 'gnowser'.

The ability to store active classes and functions in the knowledge base can be actually used for *self-representation* of GNOWSYS. This however remains a possibility that we wish to explore and experiment in future.

A communication interphase with the knowledge base is an integral part of the GNOWSYS kernel. A database without RPC interphase makes not much sense particularly at this time when Internet is driving every application design. GNOWSYS contains an RPC called Gnowledge Query Library (GQL), implemented currently using XML-RPC. Other clients and servers, independent of their implementation mode, can communicate with GNOWSYS remotely. These calls can be embedded in any language that support XML-RPC library as an integral part of their logic, and hence very convenient to develop applications using GQL. This brings us to one of the most vital implementation strategies of this application.

3 Semantic Computing Vision of GNOWSYS

Not withstanding the original motivations as explicated in the first section, along the way, while the prototype of GNOWSYS is being built some other possible strategies of implementing semantic computing in general, and semantic web in particular, came to light. These are like the unintended consequences of the original project. I wish to share these possibilities in this section. Since these possibilities came to light while solving the problem of how should GNOWSYS be implemented, I will first start there.

3.1 Implementation

Main implementation strategy of GNOWSYS is to develop it as a hybrid database (with RDBMS, OODBMS, and distributed DB features) without encoding in XML, but with an ability to map to any XML encoding schemes available with ease. This point is specially mentioned here because the hype created among the community on XML actually made XML based knowledge representation tools very very slow. Loading an XML file representing, say of the popular English thesaurus Wordnet[29], in any such application and perform faster manage,

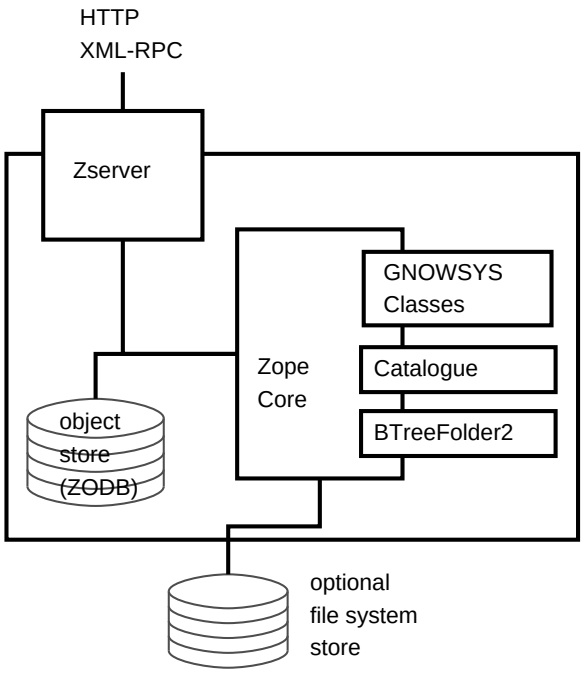

Fig. 2. A diagram illustrating the implementation model of GNOWSYS database. ZOPE's Zserver, ZODB, Catalogue, and another third-party product BTreeFolder2 provide a feature rich environment for GNOWSYS. Diagram is made based on Zope Architecture[31]

search, and retrieval operations, we will realize that XML encoding at an application layer is not a sane idea for larger knowledge bases. Keeping this in mind the physical storage of knowledge base in GNOWSYS is implemented on a web application server ZOPE (Zee Object Publishing Environment)[30], which took care of several requirements of a database, helping us to focus on implementing a faster prototype, including making it inter-operable. ZOPE provides state-of-the-art features for publishing (serving) objects on the Internet, customizing and extending existing objects, multi-user authentication, transaction management, version control, history, and most important for our purpose, a catalogue for indexing both data and metadata stored in the knowledge base. ZOPE also provides more than one architecture of creating views of the database using DTML (Dynamic Template Markup Language) and ZPT (Page Templates) templating architectures. Its in-built support for XML-RPC came in handy for easy implementation of an RCP port of GNOWSYS, GQL.

GNOWSYS is developed by using extendibility model of ZOPE. All the GNOWSYS component classes and interfacing functions are all implemented in a versatile, full-featured object-oriented programming language Python [3], in which ZOPE is also developed. The knowledge base is stored physically within ZODB, a special object storing tank, the contents of which can be accessed through ZServer which supports a wide variety of RPC protocols via HTTP.

3.2 Possibility of Semantic Web Without Unicode Markup

A notable feature of GNOWSYS is the provision of a unique URL for each object in the knowledge base. This feature is essential for using GNOWSYS as a distributed publishing tool for large structured knowledge bases like thesauri, concept bases, encyclopedia, and any such information systems including making regular dynamic web sites. But, most enabling aspect of this feature is for developing semantic web. GNOWSYS provides a unique opportunity for developing semantic web without *regular* use of markup. This is because GNOWSYS is a server listening to structured queries directly over the Internet. In the current practice the database schemes were hidden due to the intervening markup layer. While XML solves this problem by declaring the structure, it does it in an expensive way. Let us see how.

A client to server communication is an agent to agent communication. No human being is required to make the client understand the markup. Therefore, what we need is a binary markup between artificial agents, and a translation between binary markup and UI for us to see the structure and semantics. Currently the documents in WWW are structured using a markup mostly in ASCII or Unicode. It made good sense because the communication was mainly intended to be between a machine and human being. In future when the web extends to semantic web, a lot of communication is expected to take place between machines. Why should machines need such expressive (read expensive) markup, if human beings are not reading the structure?

GNOWSYS model, when frozen, like all databases will have a finite number of knowledge organizers for structuring all structurable data. Unstructured data will of course pass as Unicode. Such structured data can go without markup in Unicode, but a more economical code, reducing the required data transfer and also the need to parse. Unicode markup is possibly required for programmers to give structure, and for data exchange between heterogeneous systems. The very idea of Unicode based markup like XML was invented to meet this requirement. But, since we are slowly moving towards standards for knowledge exchange, we should move towards databases structured with a standard structure. Databases can be heterogeneous in their implementation strategies, but could nevertheless agree on standard knowledge exchange schemes, just as computing industry adopted Unicode for human interphase. If, e.g., OWL becomes a standard, then databases should be made for storing OWL schema, rather than working with fat XML files. GNOWSYS or such systems can help us move in this direction.

When this happens a browser can directly render the structured knowledge directly from the knowledge base into either textual form, controlled natural

language, tabular form, concept graphs or any such human interphase which preserves the structure and semantics. An advantage of rendering structure on the client side is the user can choose the preferred mode of drawing depending on the toolkit the client uses. Additional advantage is eliminating (1) the creation or storing of structure as Unicode based markup on the server side and (2) interpreting such markup on the client side.

Further, GNOWSYS uses a GQL, Gnowledge Query *Library*, not a *language*. Since this is a library, the query and management functions can be directly embedded in *any* programming language, eliminating the need to perform additional parsing on the server side. Agent oriented computing becomes easier since the agents talk directly to the knowledge base using GQL.

Computers had demonstrated their potential that they can 'read' and 'reply' more efficiently as databases, and programmers had demonstrated that they can manage databases easily. Massive deployment of databases everywhere and by anybody, is a good indication that this approach works. By transforming web servers into knowledge bases with RPC support we can publish large scalable semantic web sites, and not by serving through web servers publishing fat XML files containing knowledge bases.

3.3 Computing Without Syntax!

It was realized very early that GNOWSYS model can be used for visual computing without explicit syntax. A prototype was developed with the following design: Using a script, a Python library was scanned and the metadata of classes and functions (such as module name, function name, documentation string, number of variables required) were obtained and surrogate 'Function' objects were installed in the knowledge base. These objects act as an interphase for the user on one hand and the Python interpreter on the other hand. When these instances were invoked, Python interpreter was called and the function executed. We then combined such functions with logical connectives (including conditions, loops, and logic operations) using another class called 'Flow Type', which defined the flow of the program. These instances could be recursively reused in bigger programs taking advantage of the persistent storage of the objects. A demonstration of our first prototype implementation can be seen from [32].

Currently we are extending GQL to support query and management of procedural objects, so that gnowser can help in developing a visual computing environment more efficiently than doing it in HTML. Another development is to advance this support for all free software libraries, other than Python programming language, such as C, C++, LISP, Prolog, Perl, etc, so that a true visual computing suit could be built. This possibility adds a newer dimension to GNOWSYS, which is: (1) to blend functions from different languages in a single program, (2) since the functions have a unique URL a program can refer to functions spread anywhere on the Internet (possibility for semantic grid computing). This will be the basis for web services model of GNOWSYS. Interfaces for doing (1) and (2) are currently being developed.

Needless to say, such a development will have very high relevance for education, including teaching programming. The current emphasis on the syntactical marks of a program can be diverted to the semantics and logic of the program, leaving the former to a few expert computer programmers. Since more people know logic than programming, all those who can think systematically could begin to program using gnowser.

Let us be reminded that there is no language without syntax, so obviously computing without syntax is just incorrect. What is meant is to make syntax implicit by following visual rules, so that programmers can focus on logic. This qualification applies to earlier sections too. Markup exists for any language, but remains implicit knowledge to the agents (whether human or artificial). Once we have made the computers to understand the implicit rules, there remains no reason to input that feed every time. Fortunately (unfortunately sometimes) computers never forget.

4 Functionality of GNOWSYS

Based on the discussion above, one may see that it is very difficult to give an exhaustive list of applications using GNOWSYS. Only a few applications envisaged with GNOWSYS are presented here. Since the system can be used as knowledge base on personal computers, it can be deployed for desktop as well as server oriented applications.

- As a web based hybrid database (as a file system, RDBMS, OODBMS, distributed DBMS) with querying and remote management. This feature will make it highly suitable for agent oriented computing solutions across the Internet.
- For building structured knowledge bases with a higher degree of expressiveness. This will help build AI based applications using GNOWSYS as a database. Digital encyclopedia, thesauri, dictionaries or glossaries can be made very easily. Even multi-lingual databases for machine based translation can be linked from one GNOWSYS instance to another since distributed relations between objects across the network is possible.[33]
- Lessons and concepts (learning resources) can be organized according to the standard metadata suggested by LOM/SCORM models. Learning paths, evaluation tracks, usage history etc., can be persistently stored and accessed by establishing networking relations and conditions among them. Concept graphs, and semantic networks can be built easily since the highest possible granularity of learnable resources in GNOWSYS are concepts. Automated objective evaluation can be achieved by matching expert's with learner's knowledge map. Dynamics of learning (cognitive development) can be modeled.
- Using semantic computing feature of GNOWSYS, online tutorials can be built for teaching procedural skills such as mathematics, programming languages.

- Most semantic web solutions require the use of XML based representations like OWL, XTM etc. GNOWSYS is unique in its ability to provide semantic web without the use of XML, though using the data-exchange sub-system of GNOWSYS it is possible to export/import the content in XML or for that matter any other markup.
- Since GNOWSYS database is available as a web-server, dynamic web sites and web services can be made by employing the powerful ZPT, Python Script objects, and DTML templates available in ZOPE. Zope users need not use an external database, instead they can use GNOWSYS. Web sites built with GNOWSYS are "semantic web ready", for the structure of the data can be queried by software agents. Normal dynamic web sites conceal the database scheme, while GNOWSYS sites do not.
- As object sharing system can be built for huge desktop solutions that use CORBA like implementations, like GNOME/KDE etc. This will save processing time since data files are stored persistently in the database and not as XML files as is the case currently in thees desktops. This will make reusability of data across applications very efficient.
- As a personal knowledge manager (PIM) application for storing, and organizing emails, hyperlinks, notes, documents, books, blogs, files etc., with optional attributes to publish, can build a custom home page efficiently. GNOWSYS can send email through SMTP servers and receive through POP3 protocol.
- A file system with user level attributes on files is available on modern file systems like XFS. For storing very large knowledge bases GNOWSYS is being ported to XFS file system. This indicates the possibility of using GNOWSYS for a semantic desktop OS with cacheable metadata and data.

5 Status of the Project

GNOWSYS prototype building began in 2001 with the first release (0.2) in 2002. The current release version is 0.6 at the time of writing this document. The current development version is 0.7 slowly progressing for the first stable version 0.8. The code is contributed by several programmers[34].

Several modules are actively developed and it is deployed to build a community portal gnowledge.org[35], which will serve a large multi-lingual corpus of concepts of science and folk-lore in various languages with a facility to weave true propositions. It is an official GNU project, and so I hope like many other GNU projects it will continue to exist and maintained with support from developers across the globe[36].

GNOWSYS is deployed as a dynamic web server of Homi Bhabha Centre for Science Education[37]. Recently an English thesaurus was built using the entire database of Wordnet (an English thesaurus developed by Princeton University), containing around 100,000 objects and 300,000 semantic relations among them[33]. This instance is a good demonstration of the potential of GNOWSYS as a knowledge base. A prototype of active procedural objects showcasing web service model of GNOWSYS was also developed[32].

The objective of GNOWSYS as a knowledge base is very near completion, but as modeling tool for process modeling and cognitive development still remains a concept on paper. This cannot be done without introducing the time dimension in the system. A proper scheme for authoring temporal knowledge, and to build systems that are process driven is being worked out based on pi-calculus and Petri Nets[38, 21]. It is interesting to note that a flexible modeling scheme based on communication between artificial and human agents necessitates the need for process driven architectures on one hand and analogy driven architectures on the other[39]. In this context John Sowa's remarks on dealing with "knowledge soup" are relevant and provide both perspective and guidelines to the current challenges[40].

References

1. Simon, H.: Sciences of the Artificial. 2 edn. The MIT Press, Cambridge (1982)
2. Minsky, M.: A framework for representing knowledge. In Winston, P., ed.: The Psychology of Computer Vision. McGraw-Hill (1974)
3. Novak, J., Gowin, D.B.: Learning How to Learn. Cambridge University Press, UK (1984)
4. Mintzes, J., Wandersee, J., Novak, J., eds.: Assessing Science Understanding – A Human Constructivist View. Academic Press, USA (2000)
5. Fisher, K., Kibby, M., eds.: Knowledge Acquisition, Organization, and Use in Biology. Springer-Verlag, Germany (1996)
6. Fisher, K., Wandersee, J., Moody, D.: Mapping Biology Knowledge. Kluwer Academic Publishers, The Netherlands (2000)
7. Ausubel, D., Novak, J., Hanesian, H.: Cognitive Physchology: A Cognitive View. Holt, Rinehart and Winston, New York (1978)
8. Mintzes, J., Wandersee, J., Novak, J., eds.: Teaching Science for Understanding — A Human Consctructivist View. Academic Press, USA (1998)
9. Nagarjuna G.: Layers in the fabric of mind: A critical review of cognitive ontogeny. In Ramadas, J., Chunawala, S., eds.: epiSTEME 1: An International Conference to Review Research on Science, Technology and Mathematics Education, Homi Bhabha Centre for Science Education (2005) Forthcoming.
10. Kuhn, T.S.: The Structure of Scientific Revolutions. Second edn. University of Chicago Press, Chicago (1970)
11. Carey, S.: Conceptual change and science education. American Psychologist **41** (1986) 1123–1130
12. Thagard, P.: Computational Philosophy of Science. The MIT Press (1993)
13. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284** (2001) 34–43
14. Fensel, D., Hendler, J., Lieberman, H., Wahlster, W., eds.: Spinning the Semantic Web. The MIT Press, Cambridge, Massachusetts (2003)
15. Anderson, J.R.: The Architecture of Cognition. Harvard University Press, Cambridge, MA (1983)
16. Siegel, N., Goolsbey, K., Kahlert, R., Matthews, G.: The Cyc System: Notes on Architecture. Cycorp. Inc., Austin, Texas. (2004)
17. Working Group: Standard upper ontology. http://suo.ieee.org/ (2003) IEEE P1600.1.

18. Sowa, J.: Knowledge Representation: Logical, Philosophical and Computational Foundations. Brooks/Cole, USA (2003)
19. Giunchiglia, F., Shvaiko, P.: Semantic marching. In Giunchiglia, F., Gomez-Perez, A., Pease, A., Stuckenschmidt, H., Sure, Y., Willmott, S., eds.: Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems. Volume 71. (2003)
20. Object Management Group: Unified modeling language. (http://www.uml.org/)
21. Weber, M., Kindler, E.: The petri net markup language. Technical report, Petri Net Technology for Communication Based Systems (2002)
22. CL Working Group: Abstract syntax and semantics: Common logic working group (2003)
23. Sowa, J.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley Publishing Company, USA (1984)
24. KIF: Knowledge interchange format. (http://logic.stanford.edu/kif/dpans.html) Draft proposed American National Standard, NCITS.T2/98-004.
25. Berners-Lee, T.: Foreword. In Fensel, D., Hendler, J., Lieberman, H., Wahlster, W., eds.: Spinning the Semantic Web. The MIT Press, Cambridge, Massachusetts (2003)
26. Antoniou, G., van Harmelen, F.: A Semantic Web Primer. The MIT Press, Cambridge, Massachusetts (2004)
27. XTM: XML topic maps. (http://www.topicmaps.org/)
28. pyGTK: Gtk+ for python. (http://www.pygtk.org/)
29. Wordnet: A lexical database for the english langauge. http://www.cogsci.princeton.edu/ wn (2004)
30. ZOPE: Zee object publishing environment. (http://www.zope.org/)
31. ZOPE: Zope architecture diagram. http://zope.org/WhatIsZope/ZopeArchitecture (2003)
32. GNOWSYS Team: Active procedural objects. http://zope.hbcse.tifr.res.in/pr/ (2003)
33. Thesaurus: An English thesaurus implemented in GNOWSYS. http://www.gnowledge.org/wordnetbth/ (2004)
34. GNOWSYS Team: Code contributors to GNOWSYS. (http://www.gnowledge.org/)
35. Gnowledge.org: A portal for weaving gnowledge. (http://www.gnowledge.org/)
36. GNOWSYS: Home page of gnowsys. (http://www.gnu.org/software/gnowsys/)
37. HBCSE: Home page of Homi Bhabha Centre for Science Education. http://www.hbcse.tifr.res.in/ (2003)
38. Weber, M., Kindler, E.: Petri net kernel: An infrastructure for building petri net tools. Software Tools for Technology Transfer (STTT) **3** (2001) 486–497
39. Sowa, J.F., Majumdar, A.K.: Anological reasoning. In Aldo de Moor, Willfried Lex, B.G., ed.: Conceptual Structures for Knowledge Creation and Communication, Proceedings of ICCS 2003. Volume LNAI 2746., Berlin, Springer-Verlag (2003) 16–36
40. Sowa, J.: The challenge of knowledge soup. In Ramadas, J., Chunawala, S., eds.: epiSTEME-1: An International Conference to Review Research on Science, Technology and Mathematics Education, Goa, India, Homi Bhabha Centre for Science Education (2004) Slides of the talk available at http://www.jfsowa.com/talks/challenge.pdf.