SPHINX: Current Status and Future Plan

Sangki Han
Computer and Communication Center, 2nd Lab.
Samsung Advanced Institute of Technology
P.O. Box Suwon 111, Kyungki-Do 449-900, KOREA

Younghoon Kim, Youngpyo Jun, and Jung Wan Cho
Department of Computer Science
Korea Advanced Institute of Science and Technology
P.O. Box 150, Cheongryang, Seoul 130-650, KOREA
1989

1 Introduction

We have developed a hybrid knowledge representation system called Sphinx since 1986. Sphinx project has been carried out as a part of a national project for next generation computer systems in Korea since 1984. Recently, we completed the first version of Sphinx and demonstrated it to our Ministry of Science and Technology.

Sphinx can be regarded as a descendant of the KL-ONE knowledge representation language and is influenced by Krypton and KL-TWO. Sphinx has the following characteristics:

- It consists of two major components like other hybrid systems for terminological reasoning and assertional reasoning.

- The assertional component is based on Horn clause and, consequently, Sphinx itself is a theorem prover for Horn clauses augmented with a classification-based reasoner.
The query processing of Sphinx is extended to handle more general logical queries including explicit logical connectives and quantifiers. For this, we also proposed a query evaluation mechanism reflecting knowledge in two components with some assumptions.

For efficient management of knowledge base, a simple reason maintenance and dependency-directed backtracking strategy based on the notion of the Negation-As-Failure rule is used in Sphinx.

Sphinx is implemented in Quintus Prolog (tm) on a SUN workstation. In addition, we have designed and implemented a programming environment for Sphinx using ProWindows.

As a national project, a new Prolog machine called XWAM has been also developed by Logic Machine Group in Computer Architecture Laboratory at KAIST and Sphinx is ported on this machine. Although XWAM is a parallel logic machine, Sphinx is not executed in a parallel way because of its side-effects. We plan to extend the XWAM to handle with those features and, eventually, we hope to run Sphinx on a parallel machine.

2 Current Status of Sphinx

2.1 Terminological Reasoning

The terminological component (TBox) is similar to those of other hybrid systems except its base language is executed in the logic programming framework. From the knowledge level analysis, its term-forming operations can be classified into the followings.

- Primitive Specialization: it reflects the analytic sentence.
- Conjunctive Concept: A Concept is a conjunction of $C_1$ and $C_2$.
- Specialization by Role Value Restriction: A Concept is a specialization of $C_1$ whose Role $R$ is restricted to $C_2$.
- Specialization by Existential Role Restriction I: A Concept is a specialization of $C_1$ which has a Role $R$. 
• Specialization by Existential Role Restriction II: A Concept is a specialization of C1 which has a Role R and R is restricted to a Concept C2.

• Partition: A Concept is partitioned into C1, ..., C1.

• Role Specialization: A Role is a specialization of R1 whose range RC is restricted to a Concept C1.

• Role Chaining: A Role is a chain of two Roles R1 and R2.

As you can see, this set of term-forming operations requires a NP-hard algorithm for the subsumption test. Sphinx uses an incomplete algorithm for the subsumption to build a usable system.

As for the knowledge base inspection, several operations are defined:

• Subsumption test

• Find Concept satisfying a given specification

• Find the Most Specific Generalization (MSG) and the Most General Specialization (MGS)

• Disjointness test

2.2 Assertional Component

The assertional component (ABox) is the most distinguished feature of Sphinx. It is a sort of a theorem prover for Horn clauses. The knowledge base in the ABox contains assertions, which user tells, and deduction rules which are generated along with the terminological knowledge to reflect the semantic constraints.

The ABox has its own language which can be considered as an extended logic language with which we can specify explicit logical connectives and quantifiers. For this, we devised a new query evaluation procedure which proves a given query from the whole knowledge base involving the TBox and ABox.

To implement the query evaluation procedure, we used the meta programming technique in logic programming. We have come to know that logic programming paradigm is very useful in developing extensible interpreters.
2.3 Knowledge Base Maintenance

To be a real knowledge representation system, it must permit incremental additions and retractions of knowledge at any time. So far, we permit incremental additions and retractions of assertions to and from the ABox.

To maintain the knowledge base consistently, it is required to find out the dependencies between inferences. Some systems use the so-called truth maintenance system and dependency-directed backtracking for this. In the case of Sphinx, it is very simple to maintain assertions if only facts are permitted to be added or retracted because Sphinx adopts the Negation-As-Failure (NAF) rule to infer negative knowledge.

The NAF simplifies the additions of assertions. It is only necessary to add premises to the ABox. On the contrary, to retract a fact, Sphinx must know what will be influenced by the fact to be retracted. The dependencies of facts are, however, inferable from the TBox network which contains the semantics of predicates appearing in the ABox.

From this point of view, we designed and implemented a new strategy for knowledge base maintenance for Sphinx. Its simplicity comes from the absence of the explicit negation and semantic dependencies between predicates.

2.4 Programming Environments

Currently, the programming environment of Sphinx is built using ProWindows(tm). We have built a graphical programming environment using windows, pop-up menus, and icons, including a graphical browser, editors for Concept, Role, and assertions, and a restricted natural language interface.

We also designed and implemented a high-level term-defining language facilitating the knowledge base creation and modification. It is much similar to the conventional frame language. Term definitions expressed in this language are compiled into their internal representations and they can be translated into this high-level language reversely.

So far, we have designed a graphical editor for creating, modifying, and revising the TBox network. It is being under implementation. With this facility and the graphical browser, it becomes much easier to maintain the TBox network.
3 Current Interests and Future Plan

Sphinx is an evolving system. We, Knowledge Systems Group in CALAB, want to extend it to be a development platform for further research, especially for multiagent problem solving. For this, we plan to extend the capability of Sphinx to cover temporal reasoning and knowledge and belief modelling.

For temporal reasoning, a new box (we currently call it TempBox) is being considered now. TempBox contains a set of temporal relations between temporal points and a reasoner which makes inferences temporal relations for given two time points. In other words, we will adopt the time point-based approach rather than the time interval-based approach.

We developed a model for temporal reasoning based on time points which saves the space usage for maintaining temporal knowledge and reduces the time significantly for updating them when a new temporal relation is asserted. This model, however, requires some extra time \(O(n)\) to deduce a temporal relation, while approaches based on the constraint propagation algorithm require the constant time.

We want to discuss this topic with other systems, especially the temporal constraint handler for the BACK system.

For multiagent systems, the knowledge and belief modelling facility is also required for Sphinx to handle problems in the multiagent environment. M-KRYPTON is an example issued this topic. In our next project, we are considering an extension of Sphinx to handle the multiagent problem solving to some extent. We don’t want to use the possible world semantics (PWS) since there is no efficient proof mechanism developed until now based on the PWS.

In brief, we are considering an architecture consisting of multiple Sphinxes, where each agent is represented by a Sphinx, and a new box for representing relationship among agents. This box has its own language to express the Agent Relations (AR), where those relations are represented as meta predicates. For example, consider an AR which specifies that an agent B knows what an agent A knows.

\[ \text{know-A } x \Rightarrow \text{know-B } x. \]

With this meta knowledge, our query language can be extended to be able to handle the following query.
?- ask(know-B successful-father(john)).

This query is evaluated to be true if 'successful-father(john)' is proved to be true in A’s knowledge base; that is, it is proved in a Sphinx representing the knowledge system of A. To do this, we are going to extend the query evaluation procedure to reflect the AR.

The current architecture is too restrictive to infer all epistemic relationships among agents. Our approach is directed to construct a usable system in simple domain. We already have a general purpose theorem prover for knowledge and belief based on an extended model of Konolige’s deduction model. Eventually, we hope to extend the capability of Sphinx to handle most classes of epistemic logics.

Many features of Sphinx are implemented with the meta programming technique in the logic programming. Some of us now consider the partial evaluation technique to improve the performance and extensibility.

4 What We Want to Discuss

- What Expressive Power: Logic Language for Assertion Component, temporal Reasoning, Epistemic Reasoning, and Other Reasoners

- Knowledge Base Maintenance: Reason Maintenance and Revision of Terminological Knowledge

- Applications

- User Interface for Knowledge Representation Systems

Bibliography


