THE REPRESENTATION AND INFERENCES OF HIERARCHIES

Kam-Hoi Cheng

Computer Science Department
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

UH-CS-05-21
September 8, 2005

Keywords: Artificial Intelligence, Knowledge Representation, Hierarchy, Object-Oriented, Category.

Abstract

Hierarchy is an important relationship among knowledge. We identify the basic components and the common functionalities of hierarchies, develop a new class that provides the solution, and use it in implementing the different components and capabilities of a category. First, we use it to store the generalization/specialization relationships among knowledge such as data types, polygons, and categories. Then using the transitive property of generalization, a function is implemented to infer whether a given object is a specific category. We also develop another new class to represent inheritable knowledge such as attributes and properties. Finally, we use it to represent the Geographic containment hierarchy which contains two inter-related hierarchies, one in the category level, and the other in the object level. The design of the system is done using Object-Oriented paradigms, and the system implemented in C++.

.

The Representation and Inferences of Hierarchies

Kam-Hoi Cheng
Computer Science Department
University of Houston
Houston, Texas 77204-3010

Abstract

Hierarchy is an important relationship among knowledge. We identify the basic components and the common functionalities of hierarchies, develop a new class that provides the solution, and use it in implementing the different components and capabilities of a category. First, we use it to store the generalization/specialization relationships among knowledge such as data types, polygons, and categories. Then using the transitive property of generalization, a function is implemented to infer whether a given object is a specific category. We also develop another new class to represent inheritable knowledge such as attributes and properties. Finally, we use it to represent the Geographic containment hierarchy which contains two inter-related hierarchies, one in the category level, and the other in the object level. The design of the system is done using Object-Oriented paradigms, and the system implemented in C++.

Keywords: Artificial Intelligence, Knowledge Representation, Hierarchy, Object-Oriented, Category.

1. Introduction

The area of Artificial Intelligence is very broad, and recently a comprehensive treatment of the subject matter using the rational agent approach is presented in [RUSS03]. The rational agent approach is concerned with systems that act rationally. An important first step to act rationally is to make the correct inference when needed. To accomplish that, a system needs to possess the necessary domain knowledge together with the capability to reason logically. The solution detailed in [RUSS03] is to use first order logic to represent the domain knowledge. A large scale effort in providing a knowledge base for common sense knowledge using first order logic can be found in [LENA90] and {LENA95]. One advantage for first order logic is the separation of the representation of the domain knowledge from the inference logic. Another important advantage is its completeness in inference power, in other words, any statements that can be inferred can be proved. It is exactly this great inference power that makes the speed of inference not very fast, even for some simple and well-understood relationships. One

such relationship is the hierarchical relationship among knowledge. Semantic networks [MINS68] are systems specially designed for organizing and reasoning with categories that are related in a hierarchy. These systems provide fast inferences because of its specially designed data structure. In this paper, we present an implementation of the hierarchical relationship. Our solution is similar to semantic networks except that knowledge is represented by objects. Besides keeping the efficiency and simplicity of the inference process offered by semantic networks, our solution has the added advantages of an Object-Oriented solution. One such advantage is the reusability of the implementation. Any where a hierarchical relationship is needed; its structure and functionalities may be provided by simply including an object of the newly developed class. The project, A Learning Program System (ALPS) [CHEN00], is to develop a system that mimics human learning. It is based on a simple building block premise that complicated knowledge is built on simpler knowledge. To learn complex knowledge, we need to learn simpler knowledge first, then using those to compose the complicated knowledge. What we mean by learning a specific kind of knowledge involves the identification of its important component knowledge, and the development of the functionalities needed to maintain and to use knowledge of that kind. Knowledge object may be composed by including a specific instance of each of its components. The development of a complex knowledge kind such as category is therefore decomposed into simpler sub-problems of providing the solution to each of its knowledge component. In addition, using hierarchies and polymorphism for each component, Object-Oriented solution allows mix-and-match of the different components and simpler future extensions of any component.

The rest of the paper is organized as follows. Section 2 identifies the common information and functionalities of maintaining hierarchies, and provides a brief explanation of the class that implements the solution. The usage of this newly developed class in implementing the different components and capabilities of a category is presented in the next several sections. Section 3 describes how easily we may use this new class to represent the generalization/specialization hierarchies in different kinds of knowledge, such as data types, polygons, and categories. It also explains how to use a generalization hierarchy object to implement inference functions for two different situations. Section 4 describes how we use a hierarchy object to develop another new class to represent inheritable knowledge such as attributes, formulae, and properties. This class greatly reduces the amount of teaching if such inheritable knowledge were taught to belong to the parent category instead of many children categories. Section 5 describes how we use the hierarchy class to represent the Geographical containment hierarchy which contains two inter-related hierarchies, one in the abstraction level, and the other in the object level [JING04]. Finally, section 6 presents the conclusion and some general discussion.

2. The Hierarchy Class

In ALPS, the learning program is responsible to maintain knowledge objects. Each knowledge object has multiple responsibilities. One such responsibility is to maintain the knowledge's parent-child relationship in a knowledge hierarchy. The information and

functionalities common to hierarchies are abstracted and implemented as a class. When a knowledge object is involved in a hierarchy, an object of this newly developed hierarchy class will be added to it, and it is known as the owner of the hierarchy object. We observed that knowledge may be involved in multiple hierarchies, so a knowledge object may contain multiple hierarchy objects. To locate the correct hierarchy object, a unique name is used to identify the involved hierarchy. We also observed that the hierarchy is accessed through its owner knowledge, and access may be from any position in the hierarchy, not just from the root knowledge. In addition, both its ancestors and descendants should all be accessible. Finally, it is quite possible that a knowledge object may have multiple parents in a hierarchy.

From the above discussion, the common information needed includes the name of the involved hierarchy, its children, parents, and the owner knowledge. The inclusion of pointers for both parents and children allows easy traversals up and down the hierarchy. The functions common to the hierarchies may be divided into two categories. All functions assume the action to be performed related to a host hierarchy object. The first category includes the basic functions needed to build the hierarchies in an incremental manner. These include functions to add, change, and remove a parent. They also include a function to insert in between a parent-child relationship to allow a refinement of the hierarchy. We decided not to include those functions to update the child since they may be accomplished by those parent functions accessed from the child node. The second category includes those functions that allow the program to probe the information about the hierarchy. These include functions to find parent, ancestors, children, and descendants. They also include auxiliary functions to determine if the host object is related to a specific knowledge instance as a parent, child, ancestor, or descendent. The details of these functions may be found in [JING04].

Now given this hierarchy class and its functionalities, whenever a knowledge object is taught to be involved in a hierarchy, an object of this hierarchy class is added to the knowledge object. The result is an overlay of the objects of this hierarchy class onto their owner knowledge objects. Figure 1 shows some knowledge objects before their relationships in a hierarchy were taught. Figure 2 shows those objects after their hierarchical relationships were taught. The hierarchy objects of a knowledge hierarchy are stored inside their owners' knowledge. Note that the pointers are linking the objects of the hierarchy class, not their owners. The connection is bi-directional, i.e., both children and parent pointers are maintained, and parent is assumed to be above its children in the figure. Figure 3 shows that some knowledge is also involved in a different hierarchy. Objects of the hierarchy class for the different hierarchy are represented by symbols of a different shape.

3. The Generalization/Specialization Hierarchy

First, we use objects of this hierarchy class to represent the generalization/specialization relations among knowledge. Generalization allows human to abstract the common properties of many ideas, and deal with them at a higher but

simpler level. Three earlier problems involving hierarchies includes data types [SRIN99] [KAPO00], categories [ZHAN00], and polygons [NAND01] [TANN01], respectively. The specialized codes for maintaining their respective hierarchies are now each replaced by a single "is-a" hierarchy object.

Next, we use the generalization hierarchy to implement an inference function in the category class to determine whether a given input is a specific category. For example, human "is-a" animal but not a plant. The inference can be made simply and efficiently because of the data structures embedded in the solution. Since generalization is transitive, all the function needs to do is to follow the parent pointers to search for its ancestors. The correct inference can be made as long as the hierarchical knowledge has been provided to the program.

The inference problem, however, is not limited to categories, but may involve objects of a category. These objects may possess many different aspects, each have its own generalization hierarchy. For example, consider an object of the human category, say Jack. Jack may belong to one human race and have a specific occupation. We may now teach the learning program both the human race and occupation hierarchies, and that human objects may belong to one human race and have an occupation. Suppose we taught the program that Jack is a white Civil Engineer. By using the above-mentioned function that we developed for the category class, obviously our program is able to infer that white is a human race, and Civil Engineer is an occupation. However, one would only want to infer that Jack is white or an Engineer, but not a human race, nor an occupation. To prevent erroneous conclusions such as Jack is an occupation, another inference function is added to the object class. This function is very similar to that of the category class except that it stops using the transitive law at the last level.

4. Inheritable Knowledge

Another responsibility of the category class is to maintain inheritable knowledge. For example, attributes of a category should be inheritable from its generalization categories. Similarly, knowledge such as properties and formulae are also inheritable knowledge in polygons. Inheritable knowledge allows a teacher to avoid unnecessary teaching, which can be excessive. For example, it is enough to provide the knowledge base that animal has attribute weight, instead of repeating the same fact in hundreds of thousands of different kinds of animals. Without the inheritable capability, the original class to store all properties of a polygon or all attributes of a category is basically the dictionary abstract data type. A dictionary has three basic functions, namely, insert, delete, and search; and it is easily implemented using the map library class of C++. To add the inheritable capability, we create a new class that includes an object of the original dictionary class, plus the specific hierarchy object that is responsible for inheritance, namely, the "is-a" hierarchy object. To find all inheritable knowledge, the search function not only has to look locally, but also need to search its ancestor knowledge through the hierarchy object. For example, all properties of a rectangle include not only the properties stored locally, but also those of its ancestor knowledge, such as

parallelogram, quadrilateral, etc. We may now reuse this inheritable capability simply by using an object instance of this new class.

5. The Geographical Containment Hierarchy

A different kind of hierarchies is the containment hierarchy. One such hierarchy that is specifically dealt with in this paper is the Geographical knowledge of different countries. For example, the country USA contains states, each state contains counties, each county contains cities, etc. On the other hand, some other countries, such as Canada and China, contain provinces but not states, and provinces contain cities and villages. In analyzing these containment hierarchies, we observed a number of problems that we need to address. One problem is the non-uniqueness in the names of objects, even for objects of the same category. Many cities have the same name, for example, city Pasadena exists in multiple states such as California and Texas. The street, Main, is in almost all cities of America. This problem is important because there is a need to locate the right knowledge object. Luckily, within its own context, there is only one such city or street. Our solution will exploit this context information to locate the intended object. Another problem that we observed is that there are two inter-related hierarchies. One hierarchy is in the category level, while the other is in the object level. We also observed that the existence of the hierarchy in the object level depends on the existence of the category hierarchy. In other words, in order for an object to contain another object, there must be a hierarchical relationship between the categories of these two objects. For example, country USA can contain state Texas only if country is known to contain state. However, this dependency is only permissive, i.e., not every object has to contain objects of all permitted categories. For example, country contains state, province and territory means that some countries may contain states, while other countries may contain provinces and territories. The object, country Canada, can contain both provinces and territories, which is allowed in the category level. To prohibit a hierarchical relationship that exists in the category level, one solution is to apply this knowledge explicitly to an object. However, this is rather expensive. An alternative is to assume by default that it is prohibited, which may provide an incorrect inference due to the lack of knowledge. We will adopt the first approach in our solution.

The solution to the above containment hierarchy problem is rather simple. First, to represent the containment hierarchy in the category level, a containment hierarchy object is added to each category. This hierarchy object is simply an object of the newly created hierarchy class whose hierarchy name is containment. Similarly, a containment hierarchy object is needed in the objects of the category. The same functions to maintain the generalization hierarchy of categories may be used to add the containment hierarchy, except with a different hierarchy name, containment. As for the object level, new functions are added to the object class to check the existence of the permissive hierarchy in the category level, and also to check the prohibition for individual objects. To deal with the non-unique names problem, several search functions have been implemented. All use the idea of searching for a knowledge object within a context. For example, the argument of one such function is a sequence of two arguments. To locate the “main

street" of city Pasadena of state Texas in USA, the argument is the sequence: street main, city Pasadena, state Texas, country USA. As a result, the "main street" of Pasadena of California will not be mistakenly located. Finally, given a knowledge object, we implement another function to obtain the ancestor object in the containment hierarchy of a specific category. For example, given the Houston of Texas object, the function is able to find which state, country, continent, or planet it is contained in. The implementation of all these functions basically searches the containment hierarchy using the object of the newly developed hierarchy class.

6. Conclusion

Category and its associated object classes are two of the most complicated kinds of knowledge that the learning program has to maintain. Each has a number of different responsibilities, and many of them are related to hierarchies. In this paper, we have described how different kinds of hierarchies were maintained in the learning program of the ALPS project. Two useful classes have been developed. One may be used by knowledge that is involved in hierarchies, the other by knowledge that contained inheritable knowledge. In addition, several useful functions have been implemented in the category and the object classes, respectively.

Systems developed using Object-Oriented approach has the normal advantages of reusability, maintainability, and extensibility. Using object composition technique to represent knowledge allows independent and incremental development of each component. It also allows many variations for the same kind of knowledge by mixing different variations of each component. Finally, using objects to represent knowledge allows all knowledge about a specific piece of knowledge to be stored in one single place, instead of spreading them all over the knowledge base in hundreds of thousands of separate predicates and functions as in first order logic.

Acknowledgements

The author would like to thank C. Jing in implementing the codes for the hierarchy class and the Geographical containment hierarchy. In addition, the author would like to thank the following Masters students: I. Kapoor, A. Nandigam, V. Srinivasan, U. Tanna, and L. Zhang in implementing earlier versions of codes in handling hierarchies in various kinds of knowledge.

Reference

[CHEN00] K-H Cheng, *An Object-Oriented Approach to Machine Learning,* Proc. WSES International Conference on Artificial Intelligence, June 2000, pp. 487-492.

[JING04] C Jing, *An Object-Oriented Hierarchy Learning System,* Master Thesis, Computer Science Department, University of Houston, May 2004.

[KAPO00] I. Kapoor, *A Learning Subsystem for Operators of Basic Data Types,* Master Thesis. Computer Science Department, University of Houston, May 2000.

[LENA90] D.B. Lenat and R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project,* Addison-Wesley, Reading, Massachusetts, 1990.

[LENA95] D.B. Lenat, *Cyc: A large-scale investment in knowledge infrastructure,* Communications of the ACM, 38(11), 1995.

[MINS68] M.L.Minsky (Ed.), *Semantic Information Processing,* MIT Press, Cambridge, Massachusetts, 1968.

[NAND01] A. Nandigam, *A Subsystem for Learning Properties of Two-Dimensional Polygons,* Master Thesis, Computer Science Department, University of Houston, May 2001.

[RUSS03] S. Russell and P. Norvig, *Artificial Intelligence, A Modern Approach,* 2nd Edition, Prentice Hall, 2003.

[SRIN99] V. Srinivasan, *An Object-Oriented Learning Subsystem that Learns Fractions and Complex Numbers,* Master Thesis, Computer Science Department, University of Houston, Dec. 1999.

[TANN01] U. Tanna, *Learning Parameters of Two-dimensional Polygons,* Master Thesis, Computer Science Department, University of Houston, May 2001.

[ZHAN00] L. Zhang, *A Subsystem to Maintain Class Hierarchies,* Master Thesis, Computer Science Department, University of Houston, May 2000.

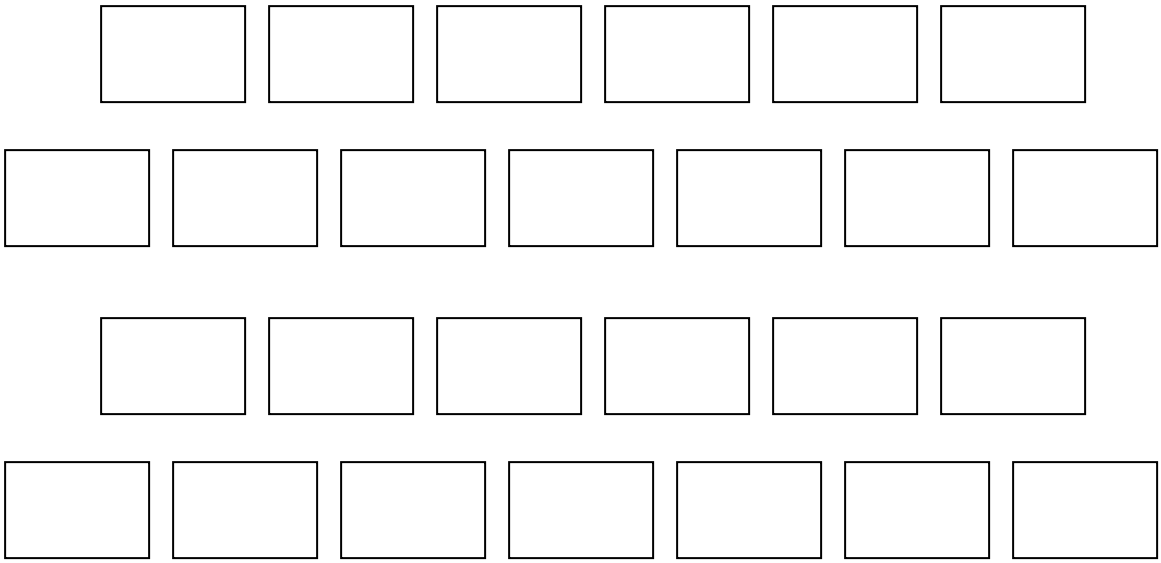

Fig. 1 Knowledge objects with no known hierarchical relationships.

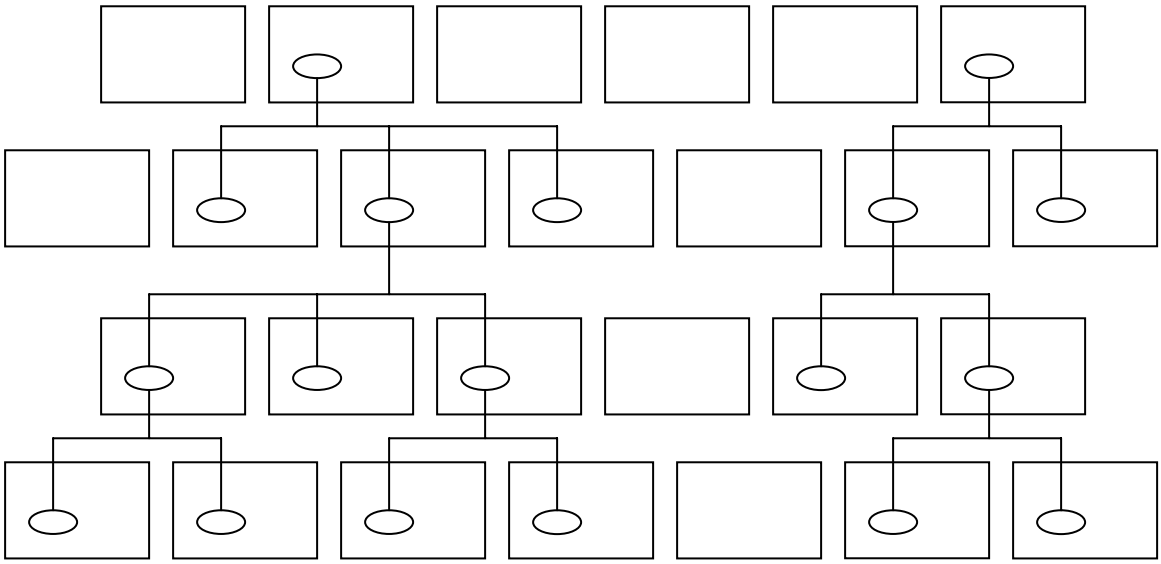

Fig. 2 Knowledge objects with known hierarchical relationships in a hierarchy.

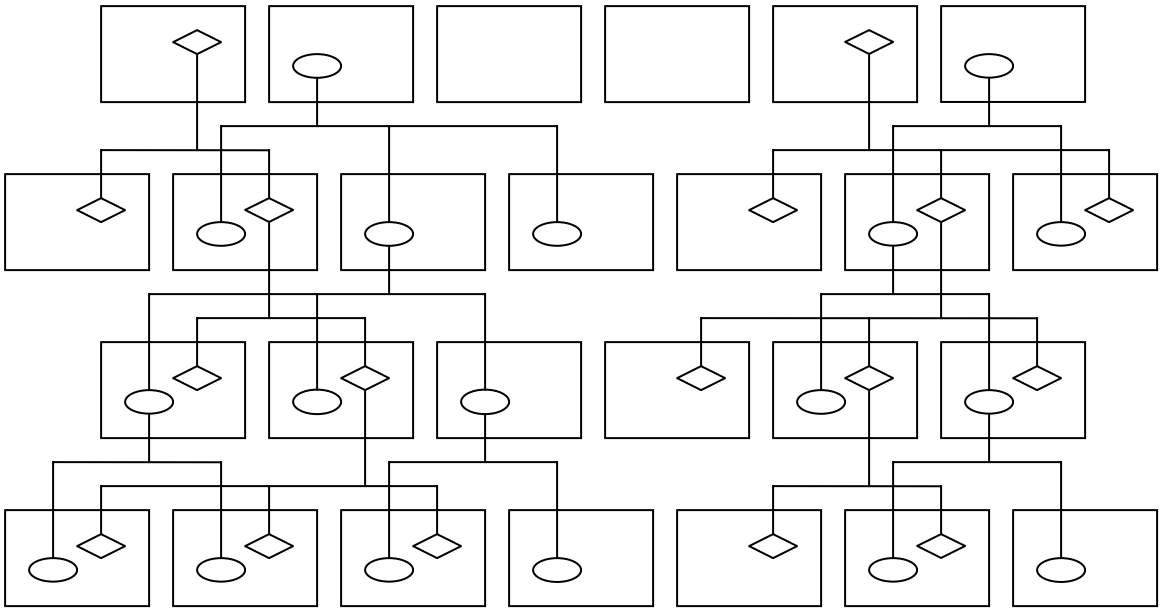

Fig. 3 Some knowledge is involved in two different hierarchies.