APPLICATION OF A GENERALIST KNOWLEDGE BASED SYSTEM FOR AUTHORING SEMANTIC NETWORKS

Corneliu NITU
1 Witherspoon Cr. Ottawa, ON K2K3L6 Canada

***Abstract**: Authoring a semantic network for a complex domain is a difficult process and any deficiencies in the authoring process are likely to be reflected in a poor representation of the essential structure of the domain. The proposed generalist knowledge based system can handle both general and specialized knowledge, as semantic networks do, but offers an array of mechanisms to allow easy development of the knowledge base and to maintain better control of the domain structure representation. Thus, this new knowledge based system can be first used to capture the domain knowledge, then a semantic network can be automatically generated. This will ensure the authoring of a semantic network that captures the domain structure in a consistent way.*
***Keywords**: semantic networks, artificial intelligence, representation techniques knowledge representation schema, knowledge-based systems.*

Introduction

The proposed knowledge based system can handle a wide range of knowledge, as semantic networks do. This requires a special knowledge representation schema, flexible enough to accommodate all sorts of pieces of knowledge and relations, but still able to allow the structuring of information in a meaningful and consistent way.
In addition to these requirements, this knowledge based system provides support for the domain knowledge acquisition, data consistency checking and contradiction solving. This recommends the proposed knowledge based system as a tool to be used in the process of authoring of semantic networks. This paper presents the application of this knowledge based system for authoring complex semantic networks.

Knowledge representation using semantic networks

A semantic network can be defined as a graphical representation relating concepts and information. A concept in a semantic network is defined as a node and labeled arcs (links) define the relationships between concepts. The example semantic network shown in figure 1 describes part of a simple astronomical solar system domain. The oval nodes represent class nodes and form the overall structure of the domain. The rounded squares nodes represent the instance nodes that are subsequently attached to the class nodes. Each class node is therefore used as an anchoring point for instance node. Every instance node is connected directly to the class structure of the semantic network.

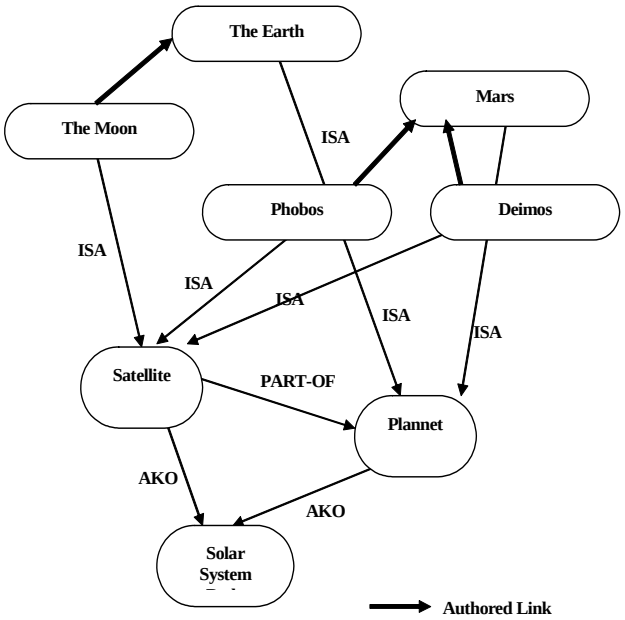

Figure 1. An example domain

The most common and important link types are the following:

ISA: relates an object to a class, i.e. it defines an instance of a class. For example, “Mars” ISA “Planet” , “Mars” being an instance of the class “Planet”. This link is unidirectional in that properties are inherited in one direction only. For example, the “Mars” node inherits all properties from the “Planet” node by virtue of this link (such properties or attributes may include the fact that a Planet orbits the Sun, thus Mars orbits the Sun). However, “Planet” does not inherit properties from “Mars”. The ISA relationship usually represents information towards the top of a hierarchy and therefore represents more general information.

AKO (a-kind-of): relates a class to another class, or may define a subset. This link type demonstrates the relationship between classes. This type is fundamental in connecting a class nodes together to form one semantic network. For example, “Planet” AKO “Solar System Body”, meaning that a “Planet” is a kind of “Solar System Body” and that a “Planet” inherits all properties of the class “Solar System Body”. The AKO link can be refered as “narrower-than”, since it represents information at the bottom of a hierarchy, which is usually more detailed or narrower in scope.

PARTOF (part-of): represents how an object is composed of other objects, or inherits only part of the parent class. This link type demonstrates how a class may be associated with component parts. For example, “Geographical Feature” PART OF “Planet”. The part-of link implies that there is a relationship between instances of classes connected via a part-of link.

HASA (has-a): relates an object to a property or attribute. This is not used to represent structural information (it is not used as a link type). Instead it may be used to represent knowledge within a class. For example, a “Planet” HASA “Diameter”. This may be used to to ensure that an object conforms to a class exactly.

The above link types allow the structure of a domain to be represented. However, further link types may be required, for example to represent dependencies. Further links may include “example-of”,“counter-example-of”, “supported-by” and “disputed-by”. Link types such as these provide a greater depth of information about a particular node. Therefore it may be necessary to separate a node into several nodes representing various levels of detail.

The semantic network is therefore a structure that connects the entire domain together. It is vital that the author connects the class nodes together in an accurate fashion.

It may not be clear cut decision whether to connect a node to a particular class node or not. The author of the semantic network may have to make a decision about connecting two class nodes together when the relationship between them is not a definite one. The author should detail this relationship within the class node so that any future author is clear about the defined relationship. It may not always be the case that such a one to one relationship can be defined, i.e. an instance node may not fit exactly into the semantic structure of the domain, but may still be valuable to the study of the domain. In this case the author should provide a suitable “non specific” semantic node. For example, the author may wish to add information about probe missions to various planets. The author may decide that it is not appropriate to add the semantic node “Probe Missions” as a-kind-of “ Planet”. However, it may be appropriate to have information about the Voyager missions to Jupiter, with the “Jupiter” instance node. The author could therefore define the semantic node “Related Material” as part-of “Planet”. It is important that this facility is used sparingly, otherwise the benefits of structuring the domain could be lost.

Authoring semantic networks using the generalist knowledge based system

The main goal of the knowledge representation schema presented in this section is to allow storage / handling of large amounts of highly structured interrelated knowledge.

In order to achieve this, the following issues have to be addressed:

Knowledge should be organized in such way to reflect the natural class-instance structure of the domain.

An inheritance mechanism should be supported. Inheritance allows knowledge to be stored at the

different levels of abstraction. It allows representation of taxonomically structured information and ensures the sharing of common properties among classes.
It should be possible to define an abstract piece of knowledge using another abstract pieces of knowledge.
The building blocks of the knowledge base
An abstraction captures the qualitative aspects of the problem. Describe the important properties and relationships of the domain. Figure 2. presents the format of an abstraction and the terminology used in this paper to describe it.

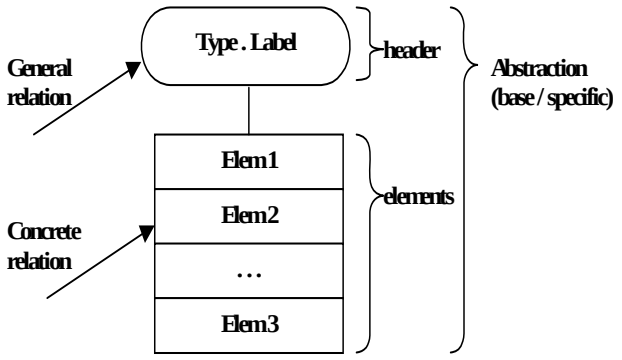

Figure 2. Abstraction format

Before going into details and describing each of the concepts involved, the basic meaning of an abstraction should be presented.
Abstractions are pieces of knowledge that are involved in every rational process. They may represent everything, from the most abstract concept to the most concrete thing. Despite their apparent simple structure, abstractions may represent very complex pieces of knowledge. In this case, they hide the entire underlying complexity. The rational process which use these abstractions may be or not aware / interested in all this complexity (in most cases, not).
All of the components of an abstraction are very important in fully defining the abstraction:
The header (Type and Label)
The elements of the abstraction
The relations each element has with other abstractions or elements.

Header

An abstraction allows representation of general principles as well as specific situations. In order to give a sense of the relative degree of abstraction, the type and label are used.
The *type* indicates the meaning of one abstraction. For example, if one abstraction has the type “bird”, it should be clear what the abstraction is about. Abstractions with the same type should have the same elements.
The *label* provides information about the relative degree of generality of a certain abstraction. The label indicates if an abstraction is a base abstraction or a specific abstraction.
The label can take the following values:

“base”: the *base abstraction* has a high degree of generality and can be considered as being the most general between all abstractions with the same type.

<name>: this is called *a specific abstraction*. The identity of the specific abstraction has certain significance.

<unique number>: again, this is a specific abstraction. It is like an unnamed individual of a type.

There is a subtle distinction between the two abstraction label types: there is no difference in the way an abstract notion is defined, but in the type of the entities to which their elements have relations to:

Elements of a base abstraction have relations mainly to base abstractions. This indicates possibilities.

Elements of a specific abstraction have relations mainly to specific abstractions and elements of specific abstractions. This indicates known facts.

Elements

The elements are very important, since they define “first hand” the abstraction. The elements can be anything from the following (but not limited to):

Enumeration of possible values (e.g. “yellow” and “red” for an abstraction called “color”).

Parts of an abstraction (e.g. “engine” for an abstraction called “car”).
They are giving an indication of the nature of the relations, which start from the elements.

Relations

They are semantic relationships, which describe:
the ways in which a system’ parts are combined
relations agent – action – recipient.
interactions between parts
causal relationship between events occurring over time.
Relations are necessary in description of complex systems and in classification. The relations do not have names or labels, and their meaning should be deduced from the element from which they start.
Relations can start only from an element of an abstraction. They may point toward:
an abstraction.
an element of an abstraction.
It is allowed to have elements with multiple relations. This is not considered a contradiction.
Each relation has a validity factor associated with it. This factor gives a measure of the confidence in that relation and, at the same time, its sign shows either the relation reflects a fact known to be true or known to be false.

Conditional relations

These are relations whose existence depends on certain conditions. They have the same meaning as regular relations, but their existence cannot be taken as granted.
The format of the conditional relations and their usage has been described in Nitu (2000).

Expanding the knowledge base

At any moment, the knowledge base contains a “world description”. Starting from this, the system should:
actively infer additional pieces of knowledge and continuously process in order to confirm/invalidate existing pieces of knowledge.
The system is not a passive repository of complete described situations. It is expected that a new situation will be described only partially. Based on the knowledge already existing in the KB (abstract and particular situations) the system should react by formulating and reasoning.
The net result of this activity is:the inference of new facts about the new situation and the identification of additional pieces of knowledge that should be provided in order to better place the new situation in the context.
The most important aspect that needs to be addressed is the mechanism of creation of new abstractions.
First, let see how new abstractions are “inserted” in the knowledge system. There are two mechanisms for the creation of abstractions starting from existing abstractions from the knowledge base. These are described in the following sections.

Cloning – creation of new abstractions

The classical dividing lines between the class type and its instances are blurring in this knowledge representation model. The well-known notion of class type has a little significance in this model. In fact, a base abstraction doesn’t differ too much from a specific abstraction. A specific abstraction may have more relations than a base abstraction of the same type, and those relations may be better specified (eg. Bird has color vs. My_Bird has color yellow.).
Lets name *cloning* the process of creation of new abstraction of the “same type” as other abstraction. This means, all abstractions will have the same elements. The Type field in the header of the new abstraction will be the same as the old abstraction.
A very important characteristic of this model is that it is possible to create a new abstraction both by cloning a base abstraction or a specific abstraction.
The process of cloning has mainly the following parts:
All relations of the source abstraction have to be analyzed.
Some of the old relations may be dropped, some new relations may appear.

For each of the relations of the new abstraction, there are several choices:

The relation has exactly the same destination as in the source abstraction.

The relation may point to another element of the same abstraction.

The relation may point to another abstraction (with the same type, however).

The Figure 3. illustrates the process of cloning on an example:

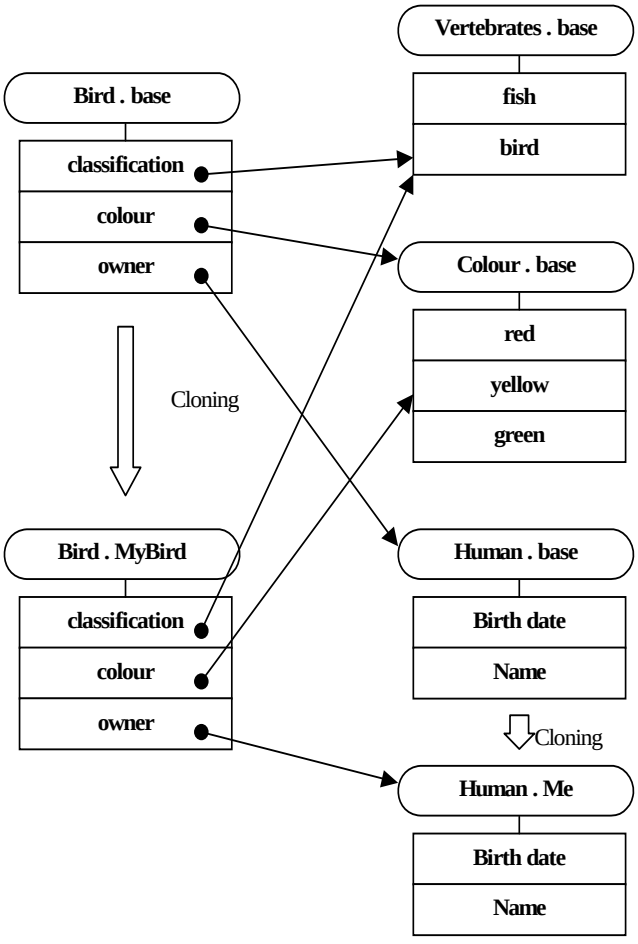

Figure 3. Example of cloning

Of course, the set of conditional relations is transmitted to the new abstract notion. However, it may be updated to accommodate:
The new name of the abstract notion
The creation of abstractions for own need.

Inheritance – creation of new abstract notions

The mechanism of inheritance permits the creation of new abstraction with a new type. This mechanism works in exactly the same way as cloning; it is in fact only a special case of cloning.
These are the reasons to employ inheritance mechanism:
A specific abstraction is considered to be a good starting point for further derivation. In this case, that specific abstraction will be cloned and the clone will receive a new type and the label "base".
New elements have to be added to a base abstraction.
When a new base abstraction has to be created through multiple inheritance.
Figure 4 illustrates the use of the inheritance mechanism through an example.

Normally, inheritance should be used only when cloning cannot be used with the same effect. There are situations when both cloning and inheritance can be used for the same result. The knowledge engineer will choose the best approach, depending on the future needs of knowledge base expansions.

From the presentation of the knowledge base building blocks and the mechanisms used for expanding the knowledge base, the following rules can be drawn:

Those abstractions with the label set to "base" are mapping to classes in the semantic network domain. The other abstractions map to instances.

The ISA relations from semantic networks can be deduced from the type field in the abstraction's header. If two abstractions have the same type, the one with the label "base" is the class and the another abstraction is an instance. However, this implicit information regarding the relation between class and instance may not be easy to extract. In the interest of an efficient generation of semantic networks, the relation class-instance has to be explicitelly stored in the knowledge base.

The HAS-A relations from semantic networks map naturally to the elements of an abstraction. The same applies for PART-OF relations.
The A-KIND-OF relations, defining inheritance relations in the semantic networks domain, have to be explicitly defined in the knowledge base.

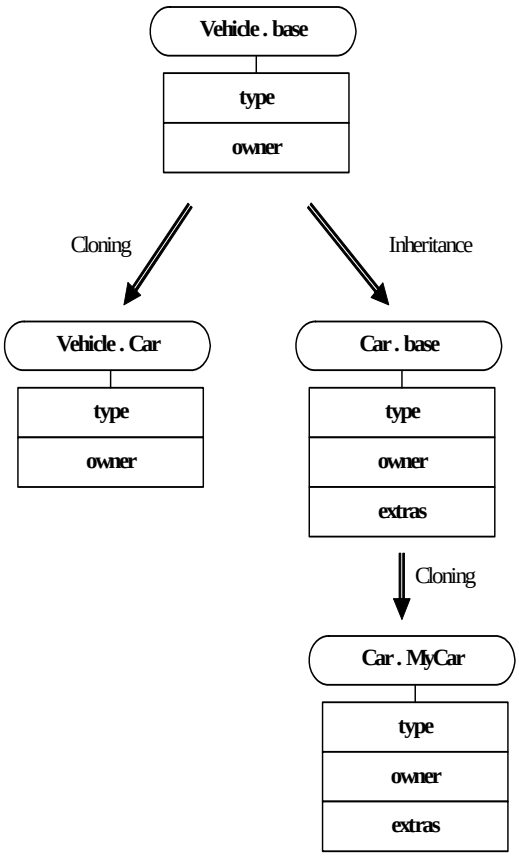

Figure 4. Example of inheritance

If, for a certain domain, the knowledge acquisition is done following the above described rules, then it will be straightforward to generate a semantic network starting from the final knowledge base.

Conclusion

The main idea behind this representation is that the way in which abstract concepts are inter-connected defines the meaning of those concepts.
The model used for the knowledge base organization can be mapped, under certain restrictions, to the model used for the knowledge representation using semantic networks.
By first employing the new proposed knowledge based system to capture the essential structure of a domain and then generating automatically a semantic networks, a better control of the domain structure representation is achieved.

References

[1] Luger, G.F. and Stubblefield, W.A. (1998) *Artificial Intelligence – Structures and Strategies for Complex Problem Solving*, Addison Wesley Longman
[2] Nitu, C. (2000) *Distributed System for Knowledge Representation and Decision Making,Int.Conference*, Int.Conference DAS 2000
[3] Nitu, C. (2001) *A Knowledge Representation Schema for a Generalist Knowledge Based System*, Int.Conference CSCS 2001
[4] Nitu, C. (2002) *A Distributed Reasoning System for Paralel and Distributed Knowledge-based Systems*, Int. Conference, DAS 2002
[5] Nitu, C. (2003), *Application of a Generalist Knowledge Based System in the Control of Optical Switches*, Int.Conference CSCS 2003
[6] Patterson, D. (1990) *Introduction to Artificial Intelligence & Expert Systems*. Prentice-Hall
[7] Rich, E. and Knight, K (1991) *Artificial Intelligence*, McGraw-Hill.