INTERACTIVE CLASSIFICATION

as a Knowledge Aquisition Tool

Tim Finin
Computer and Information Science
University of Pennsylvania
Philadelphia PA

David Silverman
IntelliCorp
Meno Park, CA

1. ABSTRACT

The practical application of knowledge-based systems, such as in expert systems, often requires the maintenance of large amounts of declarative knowledge. As a knowledge base (KB) grows in size and complexity, it becomes more difficult to maintain and extend. Even someone who is familiar with the knowledge domain, how it is represented in the KB and the actual contents of the current KB may have severe difficulties in updating it. Even if the difficulties can be tolerated, there is a very real danger that inconsistencies and errors may be introduced into the KB through the modification. This paper describes an approach to this problem based on a tool called an **interactive classifier**. An interactive classifier uses the contents of the existing KB and knowledge about its representation to help the maintainer describe new KB objects. The interactive classifier will identify the appropriate taxononomic location for the newly described object and add it to the KB. The new object is allowed to be a generalization of existing KB objects, enabling the system to learn more about existing obects.

2. INTRODUCTION

The practical application of knowlege-based systems, such as in expert systems, requires the maintenance of large amounts of declarative knowledge. As a knowledge base (KB) grows in size and complexity, it becomes more difficult to maintain and extend. Even someone who is familiar with the domain, how it is being represented and the current KB contents may introduce inconsistencies and errors whenever an addition or modification is made.

One approach to this maintenance problem is to provide a special **KB editor**. Schoen and Smith, for example, describe a display oriented editor for the representation language STROBE [25]. Freeman, et. al., have implemented an editor/browser for the KNET language [13, 14]. Lipkis and Stallard are developing an editor for the KL-ONE representation language [21]. There are several problems inherent in the editor paradim, for example:

- The system must take care that constraints in the KB, such as those defined via subsumption, are maintained.
- The system must distinguish at least two different kinds of reference to a KB object: reference *by name* and reference *by meaning*. A reference *by name* to an object should not be effected if the underlying definition of the object is changed by the editor. If one refers to an object *by meaning*, however, and later edits the object refered to, then the reference should still refer to the original description.

- The system must keep track of the origin of the subsumption relationship to distinguish between those explicitly sanctioned by the KB designer and those inferred by the system (e.g. by a classifier).
- Editors tend to be complex formal systems requiring familiarity with the editor and with the structure and content of the KB being modified.

This paper describes another approach to the KB maintenance problem based on a tool called an interactive classifier. This kind of tool is not as general or powerful as a full KB editor but avoids many of the problems described above. The interactive classifier can only be used to make monotonic changes to the KB. New objects can be added to the taxonomy and additonal attributes can be added to objects already in the KB. It does not allow object to be deleted or their existing attributes changed or overridden.

Although this may sound like a severe restriction, we believe that there are many situations where this is just the kind of KB update that is to be allowed. Consider, for example, the *computer configuration* problem which has been the domain of several recent expert system projects [18, 13, 20]. Such a system needs to have an extensive KB describing a large number computer components and their attributes, including their decomposition and interconnection constraints. An important feature of this domain is that new components are constantly being introduced as the underlying technology advances. Older components still need to be represented in the KB since there are many instalations in the field which may still need them. We may, however, want to predicate additional attributes of these older components to distinguish them from newer ones. For example, at some point in time we may add a new *laser printer* to the line of hardcopy devices. At a later time, we may want to add a new model, a *high speed laser printer*. This might involve adding two new objects: one to represent a generic laser printer with a attribute *printing speed* and another to represent the new high-speed laser printer. The original *laser printer* object would be seen as a specialization of the newly created generic laser printer.

Knowledge-based systems often represent declarative knowledge using a set of nodes, corresponding to discrete "concepts" or descriptions, which are partially ordered by a subsumption, or inheritance relation. One concept subsumes another if everything that is true about the first is also true about the second. Whenever a new node is added to the knowledge base, either during its initial construction or later maintenance, it must be placed in the appropriate position within the ordering - i.e. all subsumption relationships between the new node and existing nodes must be established. This is called *classification* because a subsuming node can be considered as a representation of a more abstract category than its subsumees. The notions of subsumption and automatic classification are very useful and have been offered as centeral features of several recent knowledge representation languages (see [5], [23], [7], and [2] for example).

Current classifiers require a complete description of the node to be added before they begin. (See [30] for a description of classification, and [17] or [24] for examples of a classifier for the representation language KL-ONE [4].) When the classifier is used directly by a user to add a new node, the user must know the descriptive terms in use in the existing KB and something of its structure in order to create a description which will be accurately classified. If the classifier places the new node in the wrong place, or if the description of the node contains errors or omissions, the user must repeatedly modify the node and redo classification until he is satisfied. The process of adding a node is much more efficient if done interactively, so that immediate feedback based on the contents of the KB is available to the user as each piece of information about the new node is entered.

The rest of this paper describes an interactive classification algorithm, which has been implemented in Prolog. Together with a simple knowledge representation language, this implementation forms a system called KuBIC, for Knowledge Base Interactive Classifier. The system takes a user's initial description of a new node and a (possibly empty) KB and either classifies the node immediately, if enough information has been specified, or determines relevant questions for the user that will help classify it. Thus a user who is familiar with the knowledge base may completely avoid the question/answer interaction with KuBIC, and use it only as a classifier, while someone who has never seen the knowledge base before may use the interaction to be presented with just those portions of the KB which are relevant to the classification of the new concept. The algorithm could be applied, for example, to knowledge representation systems or environments for building expert systems which contain classifiable knowledge bases, such as KEE [15], HPRL [16], SRL [12] or LOOPS [3].

3. THE REPRESENTATION LANGUAGE

In order to explore the underlying ideas of interactive classification, a simple knowledge representation language was chosen. The KB is constrained to be a tree structure, so each node has at most one parent. Nodes have single-valued attributes which represent components or characteristics that apply to the object or concept described. Values of attributes can be numbers, intervals, symbols, sets of symbols. The meaning of a set or range with multiple values is *disjunctive*; children of a node with an attribute with multiple values can have any subset or subrange (including single values) of the parent's value. Each node inherits all the attributes of its parent node, but its values can be restrictions of the parent attribute's values. Finally, no procedural attachment is allowed.

The Subsumption Relation

The tree structure of the knowledge base is formed by the partial ordering of its nodes with respect to the subsumption relation. The intended meaning of "*X subsumes Y*" is that whatever is represented by description Y, is also represented by the more general description *X*. All of *X*'s characteristics are inherited by *Y*, perhaps with some restriction. Since the subsumption relation is transitive, *Y* also inherits the characteristics of *X*'s subsumers (i.e. all its ancestors in the tree). In KuBIC, subsumption information is used to achieve *economy of description* and to *localize distinguishing information.*

Economy of description is a direct consequence of the inheritance of attributes and attribute values. Each description is considered to be a *virtual* description whose attributes are either local to the real description, or inherited from an ancestor. Only the most restricted value of an attribute appears in the attribute of the virtual description, even if the value occurs in an attribute of more than one ancestor description.

Classification is aided by the structure of the knowledge base. In such a taxonomic data base, distinguishing information is localized. Once a new description has been determined to be subsumed by node X, only X's subsumees are possible candidates for a more specific subsumer of the new description. The information stored at X's immediate subsumees allows the classifier to select questions which will determine which node is this more specific subsumer.

4. INTERACTIVE CLASSIFICATION

The interactive classification process is divided into three phases: acquiring the initial description of the new concept, finding the appropriate parent concept in the existing taxonomy (the most specific subsumer) and finding the apropriate immediate descendants in the existing taxonomy (the most general subsumees).

4.1. Acquiring the Initial Description

To make the interaction more efficient and minimize the number of questions the user has to answer, the user is allowed to specify an initial description of the new node. Attributes of the new node can be given, and a subsumer can be stated directly if known. Note that the user can say only that a node subsumes the new node, not that it is the **most specific** subsumer. If enough information is given, it is possible to classify the new node immediately without any further interaction. If not, KuBIC must determine what attributes to ask about so that classification can be completed.

If the initial description includes an attribute which in not currently in the KB, then the user is asked to supply certain information about the new attribute. In the simplified representation language used in KuBIC, this information is just the general constraint on possible values that the attribute can take on and a *question form* that the system can use to ask for a value for this attribute.

4.2. Establishing the Most Specific Subsumer

Because the characteristics of a node are shared by all its descendants, it is most efficient to search the tree for the new node's most specific subsumer (MSS) in a top-down manner, starting with the root. Two strategies are used to speed the search for the most specific subsumer: classification by *attribute profile* and classification by *exclusion*. The first stategy is used to take the partial description of the new KB object the user initialy presents and to identify a likely ancestor as low in the taxonomy as possible. The second strategy, classification by exclusion, is used to *push* the new KB concept lower in the taxonomy, eliciting new information from the user as needed. This second strategy is more basic to the interactive classifier and will be described in detail first.

Classifying Using Exclusion

Classifying by exclusion makes use of the fact that every node (except the root) has exactly one immediate subsumer, or parent. At all times during classification, there is one node which has been verified to be a subsumer of the new node, and is the most specific such node (the current most specific subsumer, or MSS). Only subsumees of this node need be considered as more-specific subsumers. Moreover, at most <u>one</u> of the immediate subsumees of this node may be a more-specific subsumer.

Exclusion therefore proceeds by looking for inconsistencies between the current description of the new node and the immediate subsumees of the current MSS. If no subsumees are consistent, the current MSS remains the actual MSS, and classification continues with the search for the new node's subsumees. If only one node is consistent, it must be verified to be a subsumer of the new node. This is done by asking the user, if necessary. If two or more nodes are consistent, attributes must be found to ask the user about which will help exclude as many of them as possible, until less than two nodes remain consistent.

The word "consistent" is a bit inadequate - what is actually meant by *"node S is consistent with the new node N"* is *"node S subsumes the* **current** *description of the new node."* (Note that this consistency relation is <u>not</u> symmetric.) Because the new node's description changes during its interactive classification as the user adds new information, it is possible for S to become inconsistent with it. Thus the meaning of the term *consistant* that we are using is similar to that used in discussions of non-monotonic logic [19].

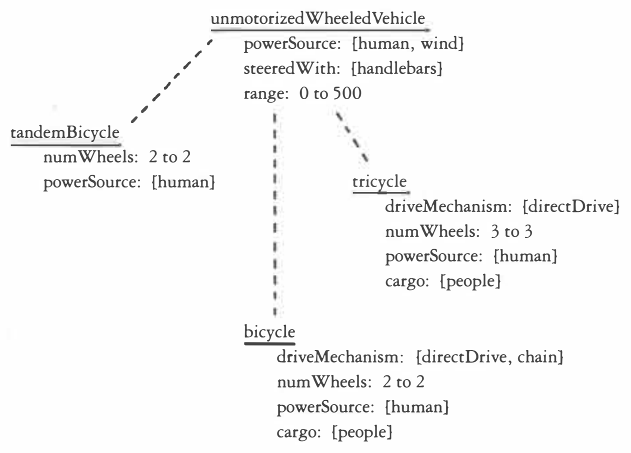
Verifying Subsumption of a Consistent Node

Because the new description is entered interactively, one attribute at a time, it is incomplete during classification. Suppose that there is only one candidate node in the set of consistent children of the current MSS. This is not enough to ensure that the candidate is a more specific subsumer of the new node since the candidate may have additional attributes that the new node does not. We are assuming, of course, that the user is giving us a *partial description* of the new KB object. If the canadiate has additional attributes, we must verify that is indeed subsumes the new node. For each such attribute, the user is presented with its value in the candidate node and is asked to confirm, deny, or restrict the value as appropriate for the new node (see figure 4-2). This is done to ensure that the values of the new node's attributes are restrictions of the values of the candidate's values. Note that the new node may have attributes which the candidate does not; this does not affect subsumption. If the user does **not** verify that the single candidate node is a subsumer of the new node, then the current MSS of the new node is established as the final MSS.

An example showing a KB fragment which requires such verification is shown in figure 4-1, and the verification interaction is shown in figure 4-2. For each node in figure 4-1, only the attributes which are either defined locally or locally restricted are displayed at that node. The new description, *tandemBicycle*, has been determined to be subsumed by *unmotorizedWheeledVehicle.* Since one attribute of *tandem Bicycle* is that it has two wheels, only *bicycle* is a consistent candidate for a more specific subsumer of *tandemBicycle.* Before asserting that *tandemBicycle* is subsumed by *bicycle*, the user is asked to verify that *tandemBicycle* also has a *cargo* attribute whose value is *people* and has a *driveMechanism* whose value is a subset of {*directDrive*, *chain*}. If the user does not agree, *tandemBicycle*'s MSS remains *unmotorizedWheeledVehicle.*

Determining the Next Question

If there are two or more candidate nodes in the set of consistent children of the current MSS, more information about the new node is required to exclude some of them. This is done by selecting an attribute to ask about, getting the answer from the user, and repeating until the set of consistent children has been reduced to zero or one node, or there are no more attributes which will help reduce the set. Two strategies are used to select an attribute to ask about from the set of attributes which apply to the set of



Dashed lines mean "subsumes," with subsumer above subsumee

Figure 4-1: KB Fragment Just Before Verification

There is evidence that the new description is a *bicycle*. I will now question you on each unverified aspects of *bicycle*. Please confirm, deny, or restrict the value, for each attribute.

What is the cargo?
cargo = [people]
Enter yes, no, or a restriction of the answer: ***yes.***

What is the drive mechanism?
driveMechanism = [directDrive,chain]
Enter yes, no, or a restriction of the answer: ***[chain].***

I've verified that *bicycle* subsumes *tandemBicycle*

Is this acceptable?: ***yes.***

Subsumer changed from *unmotorizedWheeledVehicle* to *bicycle.*

Figure 4-2: Interaction During Verification

(user's response in ***bold italics***)

consistent children: *explicit attribute ranking* and *maximal restriction.*

In our simple representation language, one can attach to a concept a list of some of the concept's attributes which are ranked with respect to their importance in classifying by exclusion. If such a ranking has been defined, then the attributes are selected in the given order. This strategy supercedes the next one, because the ranking contains external information which is not otherwise available to the system. The ranking could be based on numerical weights, but here it is a non-numerical ordering.

If there are no more attributes in the ranked list, the attribute selected to ask about is the one which *maximally restricts* the set of consistent children, in the worst case. In other words, no matter what answer is given as the value of this attribute, the minimum number of consistent children which are excluded by the answer is greater than or equal to the same minimum for any other relevant attribute. If more than one attribute is best, one is selected without regard to other considerations.

The above strategies could be augmented by using information about the particular user. Since not all questions need to be asked to perform one classification, questions which the user is more likely to be able to answer should be asked first. The user's ability to answer can be decomposed into his or her ability to understand the question, determine an appropriate response, and communicate the response to the system. The user model could be created initially by asking the user several questions intended to establish a stereotype of the user, and refined later as the user answers (or doesn't answer) questions. (See [22] and [10] for examples of this use of stereotypes.)

Classifying Using Attribute Profiles

The second classification strategy is a heuristic for searching the tree more quickly. Given the operations of determining consistency and asking the user to verify subsumption described above, if a guess could be made about possible subsumers of the new node, it would be a simple matter to verify the subsumption. A good guess is necessary, however, because the user must get involved in the verification.

The particular heuristic used in KuBIC examines the set of attributes specified by the user in the initial description to try to restrict the possible subsumers of the new node. The heuristic could also be used whenever volunteered information is allowed. It works by picking an attribute of the initial description, finding the common ancestor of all nodes in the KB which have the attribute, and using this common ancestor as a guess. The guess must be a subsumee (immediate or not) of the current MSS of the new node.

If the user verifies the guess, then it becomes the current MSS and the process continues. The user has been spared from having to answer questions about attributes of concepts which lie between the original MSS and the guess. The deeper the guess in the tree, the more questions avoided. If the user does not verify the guess, perhaps because the attribute has more than one meaning in the current KB, all is not wasted. Questions asked during verification can contribute information to the new node, or, if the attribute in question is not an attribute of the new node, KuBIC knows not to ask the question again. The system can keep guessing, whether a previous guess succeeded or not, until it runs out of attributes, or until the user becomes weary of incorrect guesses.

4.3. Establishing the Most General Subsumees

The task of classification is half completed once the most specific subsumer of the new node has been established. Finding the most general subsumees (MGS's) is the other half. Fortunately, this half is much less work because of the constraint that the KB form a tree structure.

The only possible candidates for most general subsumees are children of the MSS of the new node - i.e. siblings of the new node. (This assumes that the KB is well-constructed, so that the immediate subsumer of each node is its MSS, and the immediate subsumees are its MGS's.) Thus to find all the MGS's, it is only necessary to check whether the new node is "consistent" with each sibling in turn, and to ask the user to verify that there is no missing information about either node which misled the classifier. Note that by establishing a node as the MGS of the new node, the interactive classifier can implicitly change the descriptions of the MGS and all its subsumees - nodes which were already in the KB - because they inherit new attributes from the new node.

If the subsumption relationship is alloweed to define a latice rather than a tree, then determining the MGS's is more difficult. A newly entered node may not subsume its siblings, but could subsume some of its sibling's descendants. For example, consider a taxonomy for living things which includes a concept *livingThing* with two immediate children: *animal* and *plant*. We could use the interactive classifier to enter a new node *genderedLivingThing* to represent the concept of a *livingthing* with an attribute gender whose values come from the set {male,female}. This concept would initially be an immediate descendant of the concept *livingThing*. Neither *animal* nor *plant*, however, is a descendant of this new concept, since there are genderless animals and genderless plants. Many of their descendants, however, are subsummed by the new concept *genderedLivingThing*.

5. FURTHER WORK

There are two areas on which our current work is focused. First, we are extending the idea of an interactive classifier to a more complete representation language. Second, we are incorporating a more sophisticated *user model* to guide the interaction.

5.1. More Expressive Representation Languages

The limitations of the current work stem from the simplified nature of the knowledge representation language we have used. Using this simplified language was a conscious research stategy choice. It allowed us to focus on the notion of an interactive classifier in a simple surrounding. The two major shortcomings in KuBIC's representation language are that nodes are organized in a tree rather than a lattice, and that values of attributes must be explicit sets or intervals. Neither of these limitations should be an obstacle to extending the interactive classifier to a more general representation language. This section sketches our planned approach to such an extension (we anticipate developing an interactive classifier for the representation language HPRL [16]).

Suppose nodes were organized in a lattice structure, so that they would be allowed to have multiple subsumers. In the course of finding the new node's most specific subsumers, this would require the interactive classifier to search all paths from the current most specific subsumer down the subsumption links until a node is found that is inconsistent with the new node, or until there are no more consistent paths below a node on the path. The user would have to give enough information about the new node so that *each* child of the current MSS could be determined to be inconsistent or a subsumer of the new node, instead of stopping when the user verified one subsumer.

If attribute values are allowed to be pointers to nodes (e.g. as in KL-ONE's *value/restrictions*) then the description of a new node would depend in part on the nodes it refers to in its attributes. The algorithm described in this paper will work when the nodes referred to are already classified in the KB; only the subsumption relation would have to be changed. If they are new nodes, however, they must be interactively classified before the new node which refers to them.

5.2. More Sophisticated User Models

The second area we are working on is the incorporation of a more sophisticated model of the user. Such a model could be used to select attributes to ask about next and also to provide the user with appropriate help and guidance is answering questions. This is related to work in the context of interfaces to expert systems (see [28, 29] for example).

There has been some previous research on how expert systems get information from their users. For example, Fox [11] considered integrating reasoning with knowledge aquisition from a resource management perspective. Aikins [1] addressed the seemingly random question-asking behavior of systems which pursued lines of reasoning opportunistically, jumping around to whatever line looked most promising and asking for whatever information they needed at that point. This randomness annoyed and confused users. Aikins suggested an organization for reasoning that would result in related questions being asked together. Brooks [6] considered the amount of information systems may end up requesting from their users and found that a large number (30 or more) of requests is generally considered unacceptable. He suggested ways of cutting down on the amount of information requested, by enriching systems' models of their domains. These same sorts of considerations can be employed in the context of interactive classification.

Expert systems vary as to when they ask the user for information and when they rely on their own deductions. However, in this decision, they do not take into account the user's ability to understand and respond reliably. In Prospector [8] for example, goals are simply marked as being either "askable" or "unaskable" (never both). If the goal is "askable", the user is asked for the information. If "unaskable", the system attempts to deduce it. There are no other criteria. (On the other hand, Prospector does allow the user to change his/her answer to any question and will recompute its conclusions accordingly.) In Mycin [26], the user is only asked for information if either the system's attempt to deduce a subgoal fails - i.e., if no rules were applicable or if the applicable rules were too weak or offset each other - or the user's answer would be conclusive (e.g., lab results). In Knobs [9], a system for assisting users in mission planning, the user is asked for preferences, not facts. If the user prefers not to answer at any point, s/he can turn over control to the system and let it compute an appropriate value.

Any attempt to customize a system's way of interacting to the user at hand must allow for the fact that at times, the system is going to guess wrong - the user is not going to be able to answer its question or will answer it incorrectly or will find it annoying. Thus what we are proposing has two aspects - (1) recovering from a wrong decision (i.e., from having asked a "bad" question, and (2) modifying subsequent decisions about what sort of questions to ask. It is based on our belief that one can structure and annotate a system's inferential space in such a way that it can modify its behavior in response to the user at hand. For example, one approach might be to evaluate strategies according to how much work is required of the user to provide the information requested of him/her. This can be factored into how much work is required to: (1) understand the question; (2) acquire the information and (3) communicate the information to the system.

Of course there might be several alternative procedures the user could employ in acquiring the information the system wants, each of different difficulty for him/her, each requiring somewhat different resources. While the system's evaluation of a strategy might be based on the assumption that the user can and will use the easiest of these procedures, more refined evaluations might take into account the resources available to the particular user as well. (This information about alternative procedures - their level of difficulty and resource requirements would also be useful for certain cases where a user cannot answer the system's question, as will be discussed in the next section.)

A strategy evaluation based solely on how much work is required of the user would not be sufficient however. Another factor in the system's choice of reasoning strategy must be its a priori beliefs about the reliability of the user's information. The system should prefer a line of reasoning which depends on facts it believes the user can supply reliably over one which it believes the user can supply with less reliability.

6. SUMMARY

This paper presented the design and implementation of an interactive, incremental classifier which is used to add nodes to a hierachical frame oriented knowledge base. A knowledge representation language was defined, complex enough to resemble in certain aspects representations of current knowledge-based systems yet simple enough to allow focusing on interactive classification (for more detail and the Prolog implementation of KuBIC, see [27]). The problem of classification was described as determining most-specific and most-general subsumption relationships between the new node and nodes already in the knowledge base. Two components to the classification strategy were presented: classification using exclusion, which uses a special "consistency" relation and asks questions to exclude whole portions of the KB at a time, and classification using attributes, which uses a heuristic based on what attributes the user says the new node has to take short-cuts in the search. Both of these serve to establish the most specific subsumer; the most general subsumees are then relatively simple to find. Current work is focused on extending the concept of an interactive classifier to a more powerful representation language and incorporating a more sophisticated user model.

7. BIBLIOGRAPHY

1. Aikins, J. Prototypes and Production Rules: A knowledge representation for computer consultations. HPP-80-17, Heuristic Programming Project, Stanford University, August, 1980.

2. Hassan Ait-Kaci. Type Subsumption as a Model of Computation. In Larry Kerschberg, Ed., *Expert Database Systems*, Benjamin/Cummings Publishing Co., Menlo Park CA, 1985.

3. Bobrow, D.G. and Stefik, M. The Loops Manual. Technical report KB-VLSI-81-13, Xerox PARC, 1981.

4. Brachman, Ronald. A Structural Paradigm for Representing Knowledge. Technical Report 3605, Bolt Beranek and Newman Inc., May, 1978.

5. Brachman, R. R. Fikes and H. Levesque. KRYPTON: A Functional Approach to Knowledge Representation. 16, Fairchild Lab for AI Research, 1983.

6. Brooks, R., Heiser, J. Controlling Question Asking in a Medical Expert System. Proc. IJCAI-79, Tokyo Japan, 1979, pp. 102-104.

7. Francisco Corella. Semantic Retrieval and Levels of Abstraction. In Larry Kerschberg, Ed., *Expert Database Systems*, Benjamin/Cummings Publishing Co., Menlo Park CA, 1985.

8. Duda, R., Gaschnig, J., & Hart, P. Model Design in the PROSPECTOR Consultant System for Mineral Exploration. In *Expert Systems in the Micro-electronic Age*, D. Michie, Ed., Edinburgh University Press, Edinburgh, 1979.

9. Engleman, C., Scarl, E. & Berg, C. Interactive Frame Instantiation. Proc. First National Conference on Artificial Intelligence (AAAI), Stanford CA, 1980.

10. Finin, T. and D. Drager. GUMS1 - A General User Modeling System. technical report MS-CIS-85-30, Computer and Information Science, U. of Pennsylvania, 1985.

11. Fox, M. S. Reasoning with Incomplete Knowledge in a Resource Limited Environment: Integrating Reasoning with Knowledge Aquisition. Proc. 7th Int'l. Joint Conf. on Art. Intelligence, IJCAI, University of British Columbia, Vancouver, Canada, August, 1981.

12. Fox, M and J. Wright and D. Adam. Experiences with SRL: An Analysis of a Frame-based Knowledge Representation. In Larry Kerschberg, Ed., *Expert Database Systems*, Benjamin/Cummings Publishing Co., Menlo Park CA, 1985.

13. Freeman, M., L. Hirschman and D. McKay. A Logic Based Configurator. technical memo LBS 9, SDC, A Burroughs Company, May, 1983.

14. Freeman, M., L. Hirschman and D. McKay. KNET - A logic Based Associative Network Framework for Expert Systems. technical memo LBS 12, SDC, A Burroughs Company, September, 1983.

15. Kehler, T.P. and Clemenson, G.D. "An Application Development System for Expert Systems". *Systems & Software* (January 1984).

16. Lanam, D, R. Letsinger, S. Rosenberg, P. Huyun and M. Lemon. Guide to the Heuristic Programming and Representation Language Part 1 : Frames. AT-MEMO-83-3, Application and Technology Laboratory, Computer Research Center, Hewlett-Packard , January, 1984.

17. Lipkis, Thomas. A KL-ONE Classifier. Consul Note 5, USC/Information Sciences Institute, October, 1981.

18. McDermott, J. R1: A Rule-Based Configurer of Computer Systems. Carnegie-Mellon University, 1980.

19. McDermott, D and J. Doyle. "Non-Monotonic Logic I". *Artificial Intelligence 13*, 1-2 (1980), 41 - 72.

20. McDermott, J. XSEL: A Computer Saleperson's Assistant. In *Machine Intelligence 10*, Ellis Horwood Ltd, Chichester UK, 1982, pp. 325-337.

21. personal communication.

22. Rich, Elaine. "User Modeling via Stereotypes". *Cognitive Science 3* (1979), 329-354.

23. James Schmolze and David Israel. KL-ONE: Semantics and Classification. 5421, Bolt Beranek and Newman Inc., Cambridge MA, 1983.

24. Schmolze, J.G., and Lipkis, T.A. Classification in the KL-ONE Knowledge Representation System. Proc. IJCAI-83, Karlsruhe, W. Germany, 1983.

25. Schoen, E. and R. Smith. IMPULSE: A Display Oriented Editor for STROBE. Proceedings of the National Conference on Artificial Intelligence, AAAI, Washington, D.C., August, 1983, pp. 356-358.

26. Shortliffe, E.. *Computer-based Medical Consultations: MYCIN*. Elsevier, New York, 1976.

27. Silverman, David L. An Interactive, Incremental Classifier. Technical Report MS-CIS-84-10, University of Pennsylvania, Apr., 1984.

28. Webber, B. and T. Finin. In Response: Next Steps in Natural Language Interaction. In *Artificial Intelligence Applications for Business*, W. Reitman, Ed., Ablex Publ. Co., Norwood NJ, 1984.

29. Webber, B. and Tim Finin. Expert Questions - Adapting to Users' Needs. technical report MS-CIS-84-19, Computer and Information Science, University of Pennsylvania, 1984.

30. Woods, W. Theoretical Studies in Natural Language Understanding: Annual Report. Technical Report 4332, Bolt Beranek and Newman Inc., 1979.