RECENT DEVELOPMENTS IN NIKL

Thomas S. Kaczmarek
Raymond Bates
Gabriel Robins
USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90292

ABSTRACT

NIKL (a New Implementation of KL-ONE) is one of the members of the KL-ONE family of knowledge representation languages. NIKL has been in use for several years and our experiences have led us to define and implement various extensions to the language, its support environment and the implementation. Our experiences are particular to the use of NIKL. However, the requirements that we have discovered are relevant to any intelligent system that must reason about terminology. This article reports on the extensions that we have found necessary based on experiences in several different testbeds. The motivations for the extensions and future plans are also presented.

1. INTRODUCTION

Our work on NIKL is motivated by a desire to build a principled knowledge representation system that can be used to provide terminological competence in a variety of applications. To this end, we have solicited use of the system in the following applications: natural language processing, expert systems, and knowledge-based software. Our research methodology is to allow application needs, rather than theoretical interests, to drive the continued development of the language. This methodology has allowed us to perform an empirical evaluation of the strengths and weaknesses of NIKL. It has also helped us identify general requirements for terminological reasoning in intelligent systems.

We classify the improvements that we have made or plan to make into three broad categories:

1. Expressiveness - enhancements to the terminological competence represented in NIKL and the inferences NIKL can make regarding the subsumption relationship,
2. Environment - enhancements to the tools that accompany NIKL for both maintaining knowledge bases (knowledge acquisition) and reasoning about the terminology defined in the knowledge base, and
3. Support - enhancements to user documentation, the reliability and the availability of the implementation.

*This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335. Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion of DARPA, the U.S. Government, or any person or agency connected with them.

This paper will concentrate on enhancements made to the expressiveness of NIKL but will also describe some improvements and additions made to the NIKL environment. An introduction to NIKL will be included as background material and enhancements to the support of NIKL will be mentioned for the sake of completeness.

2. BACKGROUND

KL-ONE was designed by [Brachman 78] to "circumvent common expressiveness shortcomings**." It was designed to embody the principles that *concepts* are formal representational objects and that *epistemological* relationships between formal objects must be kept distinct from conceptual relations between the things that the formal objects represent. KL-ONE defined an "epistemologically explicit representation language to account for this distinction."

A KL-ONE concept is described by "a set of functional roles tied together by a structuring gestalt." Concept definitions "capture information about the functional role, number, criteriality and nature of potential roles fillers; and 'structural conditions', which express explicit relationships between the potential role fillers and give functional roles their meaning." A overview of the KL-ONE system has been published by [Brachman & Schmolze 85].

2.1. The classifier

An important consequent of the well-defined semantics of KL-ONE is that it is possible to define a *classification* procedure to determine the subsumption relationship for concepts in a KL-ONE network. A detailed description of the semantics of the KL-ONE classifier have been published by [Schmolze & Lipkis 83]. The classifier for KL-ONE deduces "that the set denoted by some concept necessarily includes the set denoted by a second concept but where no subsumption relation between the concepts was explicitly entered." Classifiers for KL-ONE and NIKL have been developed at ISI.

The desirable properties for the classification algorithm are soundness (no incorrect inference is made), completeness (all correct inferences are made), and totality (the algorithm always halts). Theoretical analysis work done by [Brachman & Levesque 84] has determined the limits on the expressiveness if completeness of the classification algorithm is to be maintained.

**A more complete discussion of the kinds of shortcomings Brachman is concerned with can be found in [Brachman 85]

Work on NIKL has concentrated on the issue of soundness, forgoing completeness in favor of increased expressiveness. An efficient implementation has also been a goal of the NIKL effort and the NIKL classifier is in fact nearly two orders of magnitude faster for large networks than the KL-ONE classifier.

2.2. Classification-based reasoning

The NIKL classifier provides a general weak method for categorizing descriptions of objects. It is insufficient as the sole inference mechanism for an intelligent system but it can be used very effectively (and efficiently) in what we have termed *classification-based reasoning*.

Most uses of KL-ONE and NIKL rely heavily on this kind of reasoning. It consists of a classification-reasoning cycle. The application first creates a new description of some partial result and then classifies this into a static network describing knowledge of the problem domain. Based on the result of classification, additional inferences are drawn about the partial result and a new description is constructed. These inferences are the result of some rule or procedure that examines the network looking for inferences that it is capable of making. The new description that results may achieve the goal of the reasoning cycle, in which case reasoning terminates. More typically, further classification and redescription are required and there is a continuation of the reasoning cycle.

One way of thinking about this reasoning cycle is to think of the classifier as selecting applicable rules based on the terminology that is used to describe the task domain and the problem at hand. The selection of the rules is within the terminological system, i.e., based on the definitions of terms. However, the rules are outside the terminological component and expressed in some other language.

An example of classification based reasoning can be found from the Consul application of NIKL. Consul used classification extensively in the process of interpreting natural language requests for interactive computing services. Suppose the user might typed the request, "Tell me about my afternoon meeting." Consul's natural language frontend described this request (in NIKL) as calling for a "tell-about action" where the agent to be told was the logged-in-user and object to be described was a meeting that was further described as being in the afternoon and in the possession of the same logged-in-user. **Tell-about-action**, **meeting**, **afternoon**, and **logged-in-user** were all defined as NIKL concepts. **Object** and **agent-to-be-told** were defined roles of the **tell-about-action** concept. **Owner** and **time** were roles on the concept **meeting**.

The description of the request was classified into the NIKL knowledge base which represented user expectations and system capabilities. Since Consul did not immediately understand how to execute this request, it looked for redescription rules. It sought an applicable rule that was most specific, as defined by the inheritance taxonomy. In this example, the description of the request classified under the concept, **tell-the-logged-in-user**, which was defined as a specialization of a tell-about-action whose agent must be the logged-in-user. **Tell-the-logged-in-user** had a rule attached to it that caused redescription of the request as a **display-action**. Note that had the request classified between **tell-a-user**, which subsumes **tell-the-logged-in-user**, and **tell-the-logged-in-user** it would have been redescribed as a **prepare-mail-action**. Having done this redescription Consul was not finished. This particular request continued to be refined until it was redescribed as a composite operation. The final result requested the system to display the result obtained by retrieving a meeting with a time in a specific range from a particular file in the logged-in-user's directory.

2.3. NIKL's evolution from KL-ONE

NIKL's name is evidence of the fact that it is thought of as a New Implementation of KL-ONE. Despite this, there are major differences between NIKL and KL-ONE. These are in addition to the emphasis on the efficiency of the classification algorithm already mentioned. Many of the differences are a direct result of the influence of work on KRYPTON by [Brachman et al. 83]. Close cooperation between the NIKL design team and the KRYPTON designers resulted in many system similarities despite a strong distinction on the issue of completeness.

The major difference between NIKL and KL-ONE involves the representation and use of roles.*** At the time NIKL was designed, use of KL-ONE had uncovered a need for revisions of the ideas about roles. For example, explicit structural conditions were no longer used to define the meaning of roles because of the inadequacy of the original formalization and lack of useful consequences of these conditions. In addition, the notation required in KL-ONE for relating roles in concepts (which included relations such as modifies, differentiates, and individuates) were cumbersome. The idea of thinking of roles as two-place relations and concepts as one-place relations emerged, and roles took on a new significance. Roles were defined as having a domain and a range, organized in a separate taxonomy, thought of as representations of relations, and assumed to be used consistently.

This "enlightened" view of roles was one of the lessons learned through the application of KL-ONE. It represents a case where something learned through the use of a specific tool has general applicability. Attributes of concepts ought to be formalized and have well-defined semantics also. This discovery is not unique to the KL-ONE community--it is an adaptation of the ideas found in systems build around first-order logic. The "re-discovery" of this idea simply under scores its importance.

3. THE STATUS OF NIKL

A NIKL implementation was first developed approximately two years ago. Since then it has been in use principally at ISI and at Bolt, Beranek, and Newman Inc., which contributed to the design of the system. Several "browsing" tools, syntactic support, and graphing tools have been developed and used to construct and maintain knowledge bases. A natural language paraphraser to assist users in understanding networks was also developed but has not been heavily used. Various inference mechanisms driven by the classifier have also been implemented.

*** Actually there are significant differences beyond those having to do with roles if one takes KL-ONE to be defined by the original formalization rather than the then current implementation, which did not support much of the formalism.

Applications of KL-ONE and NIKL have been in the areas of natural language processing (see the publications of [Bobrow & Webber 80, Sondheimer 84, Sidner 85, Mark 81]), expert systems (see the work of [Neches et al. 85]), and software description(see the publications of [Kaczmarek et al. ??, Wilczynski 84]). Large networks, in excess 1500 concepts, have been developed in these environments.

This experience with NIKL has led us to consider certain extensions to the language, its environment, and the implementation. The following sections will describe the extensions we consider important and explain the motivation and status of each. The extensions have been divided into roughly three categories: terminological competence, environment, and implementation.

3.1. Terminological Competence

By terminological competence we mean the ability of the system to *represent* and *reason about* various distinctions that a modeler might need to capture in defining concepts. For example, the ability to restrict the range and number of role fillers for a particular functional role adds to the terminological competence. Inferring that if a person has at least one son, then the person has at least one child (based on the fact that son is a specialization of child) is another example of terminological competence.

The benefits derived by enhancing terminological expressiveness are analogous to the benefits derived form the data typing mechanisms found in modern programming languages. Support for various abstractions, such as lists, ranges, and enumerations, make it easier for the programmer to produce correct and more compact programs. Support for all of these data types could be built in assembly language, but the programmer would be responsible for choosing a suitable representation and supporting it (e.g., defining a constructor function and doing error detection on modification). In a similar way, support for the reasoning that NIKL does could be built with frames, flavors, or well-formed-formulae, but applications would have to supply and usually duplicate reasoning to support them. This leads to a view of NIKL as a better data typing mechanism for intelligent systems. This is certainly one of the roles of NIKL in the applications we have seen. There is another however--the classification based reasoning cycle--which was described earlier.

The following sections will describe our efforts in this area.

3.1.1. Disjointness and covers

One addition to NIKL that was absent in KL-ONE is support for disjoint and covering sets. A collection of concepts can be declared as being disjoint, i.e., have no common extensions in the world. A collection can also be declared as a cover of another concept, i.e., all extensions of the covered concept must be described by at least one of the members of the covering. These two declarations can be combined to form partitions.

Having the ability to define disjointness, covers, and partitions has led to a more streamlined design of systems using NIKL. There are many cases where generic problem solving techniques require this kind of information. By making it an abstraction supported by the language, applications are freed from the responsibility of representing and supporting it.

Furthermore NIKL provides limited inferences based on these notions. This frees the application from responsibility to supply this generic reasoning. A description of the limited, though useful, reasoning performed by NIKL follows:

- As a result of disjoint classes, NIKL can determine if a concept is coherent or not. For example, a person all of whose children are both males and females, would be marked as being incoherent if male and female were declared as being disjoint. An incoherent description is admissible in NIKL but is assumed to not have any extension in the world. The discovery of incoherence almost always indicates an error in the model. As a result, discovery of incoherence has proven to be valuable during knowledge acquisition. It is one example of how well-defined semantics can assist in the construction and use of terminological knowledge bases.

  With respect to covers, a simple inference procedure is available to deduce the existence of other covers. For example, suppose male and female cover sex, spouses have a sex role that is restricted to sex, and that husband and wife are specializations of spouse. Further assume the only difference between husband and wife is a restriction of the sex role to male and female respectively, then NIKL can infer that husband and wife cover spouse. Needs for this kind of reasoning have come about in using NIKL for expert systems where certain methods of problem solving are applicable only when some covering exists. NIKL's current inferential capabilities for covers are limited to simple cases such as the one presented in this example. Plans call for expanding these capabilities as needed by applications.

3.1.2. Reasoning about role restrictions

The inclusion of an explicit role hierarchy in NIKL allows the system to infer certain properties of concepts. The example of calculating minimum number restrictions for the son and child roles presented above illustrates one kind of inference. In that example, we have propagated a minimum number restriction up to a more general relation. Obviously, we can also propagate a maximum down to a more specialized relation. These are two inferences that we have recently added to NIKL.

Another inference involves value restrictions for roles. It is illustrated by the network definition seen in Figure 3-1. The NIKL specification for this example can be paraphrased as follows:

- doctors, famous, and rich are primitive concepts[****]
- surgeons are a primitive specialization of doctors,
- very famous is a primitive specialization of famous,
- relative is a primitive relation,
- cousin is a primitive specialization of relative
- any concept that fills the role of famous cousin must fill the role of cousin and be famous,
- any concept that fills the role of rich relative must fill the role of relative and be rich and,

[****] A primitive concept or relation corresponds to the notion of a "natural kind", i.e., a predication that can only be determined by an oracle. To NIKL this means that no concept may be placed beneath this one in the hierarchy unless the concept specification explicitly says to do so.

- any concept that fills the role of rich cousin must fill the roles of rich relative and cousin.
- all the rich cousins of an "A" must be doctors,
- all the famous cousins of any "B" must be surgeons and all the rich relatives of any "B" must be very famous,

From this specification, NIKL infers the following:

- all of A's rich cousins are rich doctors,
- all of B's rich cousins are rich and very famous surgeons,
- all of B's famous cousins are famous surgeons, and
- all of B's rich relatives are rich and very famous.

```
(DEFCONCEPT Doctor primitive)
(DEFCONCEPT Famous primitive)
(DEFCONCEPT Rich primitive)
(DEFCONCEPT Surgeon primitive (specializes Doctor))
(DEFCONCEPT Very-Famous (specializes Famous))

(DEFRELATION Relative primitive)
(DEFRELATION Cousin primitive (specializes Relative))
(DEFRELATION Famous-Cousin (specializes Cousin) (range Famous))
(DEFRELATION Rich-Relative (specializes Relative) (range Rich))
(DEFRELATION Rich-Cousin (specializes Rich-Relative Cousin))

(DEFCONCEPT A (restrict Rich-Cousin (VR Doctor)))
(DEFCONCEPT B (restrict Rich-Relative (VR Very-Famous))
              (restrict Famous-Cousin (VR Surgeon)))
```

Figure 3-1: Example of role reasoning

Figure 3-2 graphically depicts the network after classification has been performed.

The conclusions illustrated in the figure are derived from the following line of reasoning. All of B's rich cousins are rich relatives and therefore very famous, so they are all also surgeons (since all the famous cousins of B are surgeons), making them doctors as well. It follows then that B specializes A since all of its rich cousins are rich and very famous surgeons, which is a specialization of rich doctors. The current classifier for NIKL supports this kind of reasoning based on the role hierarchy.

Our plans for enhancing reasoning about role restrictions include adding logic to account for coverings and disjointness in the role hierarchy. For example, if we knew that the roles, son and daughter, are disjoint and that they cover the role, child (i.e., form a partition) then we can determine the maximum and minimum number restrictions for child based on the number restrictions for son and daughter. Similar kinds of inferences can be made involving the value restrictions.

3.1.3. Roles and relations

One of the criticisms of KL-ONE and NIKL was an incomplete treatment of roles. In KL-ONE the semantics for roles was determined only by other constructs that were described for concepts. In previous versions of NIKL, all roles were primitive. Work in natural language text generation has pointed out the need for a more uniform treatment because sentences may need to describe the relationships that exist between concepts. This requires giving relations the same status as concepts in the network and establishing a correspondence between restrictions of roles at a concept and the relations those restrictions refer to. We have thus adapted a position where roles are thought of as two place relations that are defined in the concept hierarchy. We have implemented this strategy by allowing the user to define relations that may then be used as roles. The example above in Figure 3-1 illustrates this capability. Under this new implementation, relations are represented as concepts in the same hierarchy with all other concepts. All relations have at least two roles, a range and a domain.

One implication of this support is that it has allowed the user a simple way to say things such as "a car, one of whose tires is flat." In the previous implementation, the user would have to specify and name a primitive role that specialized the tire role for a car and then restrict the value of that role.

A more significant improvement results from removing an unfortunate consequence of this old procedure (which resulted from the primitiveness of all roles). The result was that nothing would classify as a kind of the concept being defined unless the user added the same role (presumably by referring to it by name) and restricting if to the same range (or some specialization of it). In the current implementation, we "gensym" a relation that specializes tire and restrict its range to flat. Any other similar or more specialized relation resulting from a restriction, for example,

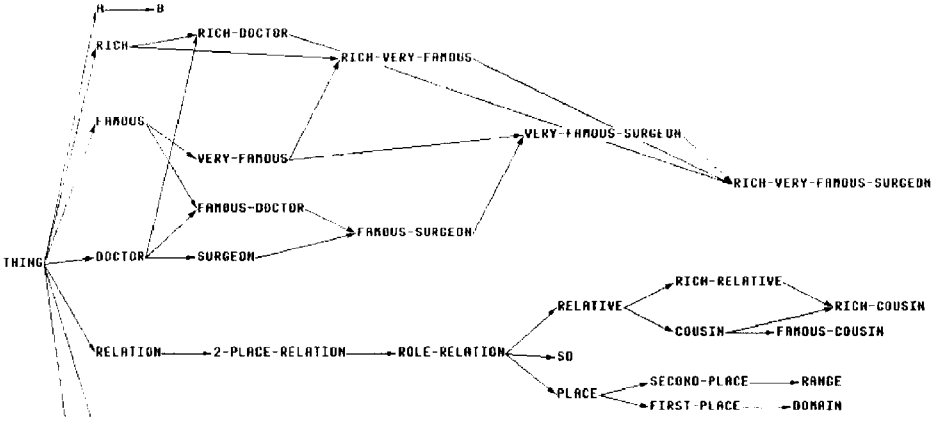

Figure 3-2: Graph of taxonomy defined in Figure 3-1

the one generated by "a car with a blown-out tire," will either merge with the gensymed relation or classify as a specialization of it. Thus, classification of a car with a blown-out tire under a car with a flat tire can happen without having to refer to a specific (and primitive) flat-tire role in the specification of the car with a blow-out.

Since relations are now part of the concept hierarchy, we can define other properties for roles and declare disjointness and coverings. One consequence of this is that we have simplified the development of support for reasoning about number and value restrictions for roles based on these notions. Another is that we can specify more completely the meaning of a relation.

3.1.4. Cycles in the network

The current NIKL classifier cannot reason effectively about cycles in the network. A cycle occurs whenever one classification depends on another. The classifier has two user selectable strategies that it follows when a cycle is encountered. In the first, the classifier stops trying to draw inferences about any of the concepts in a cycle when one is encountered. Typically a large collection of static concept specifications are presented to the classifier. It recursively descends the known hierarchy to find and classify those new concepts that have no dependencies on any other new concepts. It then unwinds the recursion and forms the newly classified hierarchy as a result. In this first mode, if the classifier discovers a cycle, it simply declares the concepts classified and warns the user about the existence of the cycle.

The second mode involves trying to proceed with classification after sorting the items of the cycle based on the number of dependencies for the items. The classifier operates on each of these in the order determined by the sort. The concept with the fewest dependencies on the others is first in this ordering. If there are concepts with equal numbers of dependencies the order depends on the time of the introduction of the concepts into the network. This has the unfortunate consequence that the result may depend on the order in which concepts were mentioned.

In the past, cycles were generally considered errors in modeling so the inadequacies of either of these modes was considered inconsequential. However, with the current support for roles as relations cycles now become more prevalent. For example, if the child relation is used to define a person, then person cannot be classified until the relation child has been classified. But if the domain of child is person, then it cannot be classified until person is classified. Obviously, a cycle results. In this situation, the second mode of classification described above performs reasonably, though incompletely (in the formal sense).

A more sophisticated classification control strategy could obviously result in a more complete classification. We have designed, and are in the process of implementing and testing, what we call the *incremental classification control strategy*. Under this regime, the classifier will maintain dependency links for all concepts and use an iterative approach to classification. When a cycle is encountered, the classifier will do the best it can with the concept with the fewest dependencies. It will then classify all those concepts that depend on that one and eventually (because of the cycle) try to reclassify the original concept after having done its best on the dependent concepts. This approach obviously cycles and needs a termination condition. The incremental classifier will stop classification when the network has reached a quiet state, i.e., no new inferences can be drawn, or some user-settable number of dependency cycles have been completed. This strategy will allow more inferences to be made by the classifier and will also provide the basis for a much improved knowledge acquisition environment. Details of the implications for acquisition will be presented later in Section 3.2.4.

3.1.5. Partial orderings

One glaring shortcoming of KL-ONE and NIKL has been an inability to define sequences. Requests for this capability have come from nearly all applications[*****] We have examined the requirements and designed a more general capability that supports partial orderings on roles.

The partial orderings in NIKL represent relations that exist between role fillers. Support includes knowledge (in the classifier) about the reflexive, antisymmetric, and transitive nature of partial orderings. One partial ordering may be a specialization of another and they are defined in the concept hierarchy like all other relations.

The NIKL user can make several different kinds of statements about the partial orderings of the role fillers. One states that all the fillers of a particular role must be ordered by a particular relation. For example, the statements of a computer program are ordered by the lexically-before relation. A second kind of statement is that all the fillers of one role are related to all the fillers of another role by a particular ordering. An example is a statement that the initialization steps of a while loop come before the termination tests, which in turn come before the steps in the body. The final kind of statement declares that the fillers of one role are the *immediate* predecessors (or successors) of the fillers of another. An example is the statement that one statement of a program is immediately lexically-before another.

Classification will involve the determination of subsumption between partially ordered sets (posets), which is a fairly expensive operation. The expense includes the construction of the representation of posets as graphs and the determination of whether one graph is a subgraph of another. The design of the implementation is such that overhead caused by this enhancement will be minimal for concepts that do not involve use of this feature.

3.1.6. Necessary and sufficient conditions

The NIKL classifier represents a particular kind of classification, one that depends on certain logical properties. There are other kinds of classification that depend on domain specific knowledge. One such kind of classification involves the definition of *sufficient conditions*. The idea is that the presence of certain evidence is sufficient to draw a conclusion if there is no contradictory evidence. For example, one might be willing to say that any mammal with a human DNA structure must be a kind of human unless there is evidence to the contrary even though we do not have evidence for upright posture, opposing thumbs and so forth.

[*****] Various extra-NIKL schemes have been adopted in past work to handle this problem. In past applications, it was not necessary for the classifier to deal with sequences so a special purpose sequence reasoner could be used.

Such reasoning has heretofore been unavailable in NIKL and KL-ONE. In light of this one can characterize the definitions of current NIKL concepts as stating necessary conditions (since no part of the description could be missing) and sufficient conditions (since the presence of them is sufficient evidence for the classifier to draw specialization conclusions). The exception to this is for concepts marked as primitive, which indicates that no set of sufficient conditions can be found.

One proposal for adding sufficient conditions would allow the user to state that some collection or collections of roles were sufficient. For example, if you know that an animal has four legs and a trunk or a finger on the end of its nose (and there is no contradictory evidence, such as it lives in a tree) then it is an elephant. Another would allow the user to state that one concept *implies* another. In this scheme, each set of sufficient conditions would be represented as a separate concept that implies the concept in question. The advantage of this approach is that it easily accounts for structural descriptions and partial orderings. We are in the process of evaluating these two proposals.

The important lesson is that sufficiency reasoning about terminology seems to be useful is several domains. We have experienced the need for it.

3.1.7. Negation

Negation is a problem for the classification algorithm as has been shown by the work of [Brachman & Levesque 84]. Nevertheless, it is a notion that nearly all applications find useful. Since we cannot admit negation and maintain tractability for the classifier, we have provided other mechanisms and conventions that seem to satisfy most users. One convention is the use of zero as the minimum and maximum number restriction for a role restriction. For example, a verb phrase with no time modifier can be modeled this way.

The ability to define partitions as disjoint covers provides a way to talk about complements, which are akin to negation. This is another addition to NIKL that was the result of expressed desires for negation. The strategy exemplified in these two capabilities, namely, providing something different than what the user asked for but which meets the requirements of the application is very much a part of our methodology for continuing the evolution of NIKL.

3.2. The Environment

The NIKL environment consists of tools that aid in knowledge acquisition and reasoning. Our experience has led to the generation of tools in both of these areas.

3.2.1. Assertions

Recording and reasoning about extensions of the terminological knowledge represented in NIKL is considered to be outside of NIKL itself and part of the environment. An *ad hoc* assertional mechanism[******] was developed for use with the CUE and Consul applications (see, [Kaczmarek et al. 83]). A more systematic approach has led to the development of a major tool for reasoning about assertions by [Vilain 84] of Bolt Beranek and Newman. This tool, KL-TWO, combined the RUP package of [McAllester 82] with NIKL. KL-TWO provides a truth maintenance package that is very useful in some applications. However, it is inappropriate for large data bases and for certain kinds of applications where efficient implementations of the assertions are required.

To correct these deficiencies (for certain applications) we have planned two other hybrid systems. The first involves coordination between the conceptual hierarchy defined in NIKL with the schemata for a commercial relational data base. With this scheme we plan to use NIKL in applications requiring the kinds of semantic browsing techniques found in the work of [Patel-Schneider et al. 84] and [Tou et al. 82]. The second involves using NIKL in coordination with the knowledge representation aspects of a knowledge-based software development paradigm. Here we are actively involved in using NIKL to define a type hierarchy and relations for the AP5 language of [Cohen & Goldman 85].

3.2.2. Reformulation

As was previously mentioned, classification-based reasoning is a common mode of use of NIKL. The terms, reformulation and mapping, have been used in KL-ONE applications to refer to this kind of activity. Currently there is a reformulation facility available that is used in the expert system research of [Neches et al. 85]. This mechanism is used to satisfy goals by expanding plans. Within the paradigm of their project, reformulation is used to generate an expert system based on a knowledge of the domain and expert problem solving knowledge. In this methodology, goals, methods, and plans are all expressed in NIKL and the expert system shell uses these to generate the expert system for a particular domain and set of goals and methods. While the facility provided was designed for a particular use, the mechanism is generic and can be applied to any number of other applications.

3.2.3. Graphic-based editing

The KL-ONE community has a rich tradition of drawing pictures with "circles and arrows." A graphical representation of concepts and networks has always been a part of the language. As the expressiveness of NIKL has increased, the cleanliness of the graphs has diminished, but nevertheless, the graphs remain useful.

We have developed an integrated set of acquisition tools in a window-based workstation environment. The tools include a graph of the concept hierarchy, an EMACS editing window, and a LISP interaction window. Within the LISP interaction window, the environment can produce highly formatted ("pretty-printed") descriptions of concepts. The atoms in these formatted displays, which refer to concepts and relations, as well as the nodes of the graph and the text in the edit buffer are all mouse sensitive and known to be NIKL constructs by the environment. This allows the user to move from one window to another in a coordinated way. It also allows the user to refer to a NIKL object simply by pointing at it in any of the various views of the network. A natural language paraphraser has also been added to this environment to assist in the understanding of the network.

We also have a tool to graph the definition of a particular concept. This tool has proven to be less useful than originally thought. While drawing concept specifications on paper with a pencil is extremely useful, we haven't been able to duplicate the free flowing expressiveness of that mode of design. Work on the

[******] This scheme was built around the KL-ONE notion of a *nexus*

human factors of the tool and the inclusion of higher level operations (the current level is, for example, add a role) are anticipated. However, the tool is useful in terms of providing a graphic presentation of a concept. The deficiencies become obvious in creating or editing a concept definition.

3.2.4. Incremental classification

A major problem with the NIKL environment arises from the batch nature of the classifier. The example in Figure 3-1 illustrates some of the many inferences that the classifier makes. For example, deciding that the user really meant rich cousins to be rich doctors, not just doctors. This kind of inference can be particularly troublesome for the user because NIKL frequently needs to generate new concepts that the user has not explicitly defined. Usually NIKL cannot pick an appropriate name for the concepts it generates. In many cases the need to generate a new concept arises from the fact that the user has inadvertently omitted the concept or made some modeling error. A better acquisition environment can be obtained by having the classifier interact with the user whenever such a concept must be generated. The user could then choose an appropriate name, decide there is an error, or tell NIKL that the concept will be defined later.

The example of interaction arising from new concepts being generated is just one case in which interaction during classification can improve the modeling environment. The control strategy that will be employed in the incremental classifier will be much more supportive of the kind of interaction that knowledge acquisition requires.

The dependency information that the incremental classifier will keep can also be used to enhance the modeling environment. This information is particularly useful for editing a concept definition and then making sure the network is properly updated and for supporting various kinds of analysis tools.

3.2.5. Surface language support

As part of our efforts we have used a general lexical analysis and semantic interpretation package developed by [Wile 81]. This package gives a flexible surface language that allows easy modifications to accommodate extensions to NIKL as we develop them. It also opens up the possibility of defining highly application dependent surface languages.

3.3. The Implementation

The current version of NIKL is in Common LISP and we have experimented with its use on a variety of workstations and mainframe implementations of Common LISP. The integrated acquisition environment depends on some specific tools found in the Symbolics ZETALISP environment. We are actively pursuing the development of similar facilities that rely on a Common LISP implementation of a form and graphics package that requires only modest customizations for various graphic environments.

4. SUMMARY

NIKL is an evolving knowledge representation tool based on KL-ONE. The experiences gained in a variety of applications have shaped the current implementation. Principal enhancements made to NIKL that were in direct response to applications needs were: the representation of roles more uniformly with concepts, support for negation, a connection to an assertional truth maintenance system, support for domain specific reasoning (triggered by classification), and more complete inferences drawn as a result of having a relation hierarchy. Further enhancements have also been suggested and continue to be developed. They include: the representation of sequences and orderings, the availability of sufficiency reasoning in the classifier, more complete inferences regarding cycles in the models, and coordination with an assertional component that supports efficient data base access.

In addition we have implemented and continue to develop tools for the knowledge acquisition environment. This work has also been sensitive to the needs that have arisen out of several application environments. Principle developments include a Common LISP implementation, and an integrated tool set that features graphic representations, formatting and paraphrasing tools, and flexible lexical analysis support. The addition of a more interactive editing style and various analysis tools is forthcoming.

ACKNOWLEDGMENTS

The development of NIKL and our plans have been the result of interactions with a number of AI researchers. Many of these were developers or experienced users of KL-ONE or one of its variants. The rest were potential or new users. The contributors represent many different organizations and research interests. The following have all made, and in many cases continue to make, significant contributions: Don Cohen, Neil Goldman, Bill Mann, Norm Sondheimer, Bill Swartout, Bob Neches, Don Voreck, Steve Smoliar, Ron Brachman, Victoria Pigman, Peter Patel-Schnieder, Richard Fikes, Ramesh Patil, Jim Schmolze, Rusty Bobrow, Marc Vilain, Bill Mark, David Wilczynski, Mark Feber, and Tom Lipkis.

REFERENCES

[Bobrow & Webber 80] Robert Bobrow and Bonnie Webber, ""Knowledge Representation for Syntactic/Semantic Processing," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, August 1980.

[Brachman 78] Ronald Brachman, *A Structural Paradigm for Representing Knowledge*, Bolt, Beranek, and Newman, Inc., Technical Report, 1978.

[Brachman 85] Ronald Brachman, "I Lied About the Trees," *AI Magazine VI*, 1985.

[Brachman & Levesque 84] Ronald J. Brachman and Hector J. Levesque, *The Tractability of Subsumption in Frame-Based Description Languages*, Fairchild Research Laboratories, Technical Report, 1984.

[Brachman & Schmolze 85] Brachman, R.J., and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, August 1985, 171-216.

[Brachman et al. 83] Ronald Brachman, Richard Fikes, and Hector Levesque, "KRYPTON: A Functional Approach to Knowledge Representation," *IEEE Computer*, September 1983.

[Cohen & Goldman 85] Cohen, D. and Goldman, N., *Efficient Compilation of Virtual Database Specifications*. 1985.

[Kaczmarek et al. 83] T. Kaczmarek, W. Mark, and N. Sondheimer, "The Consul/CUE Interface: An Integrated Interactive Environment," in *Proceedings of CHI '83 Human Factors in Computing Systems*, pp. 98-102, ACM, December 1983.

[Kaczmarek et al. ??] T. Kaczmarek, W. Mark, and D. Wilczynski, "The CUE Project," in *Proceedings of SoftFair*, July 1983.

[Mark 81] William Mark, "Representation and Inference in the Consul System," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, IJCAI, 1981.

[McAllester 82] D.A. McAllester, *Reasoning Utility Package User's Manual*, Massachusetts Institute of Technology, Technical Report, April 1982.

[Neches et al. 85] Robert Neches, William Swartout, and Johanna Moore, "Explainable (and Maintainable) Expert Systems," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 382-389, International Joint Conferences on Artificial Intelligence and American Association for Artificial Intelligence, August 1985.

[Patel-Schneider et al. 84] Peter F. Patel-Schneider, Ronald J. Brachman, and Hector J. Levesque, *ARGON: Knowledge Representation meets Information Retrieval*, Fairchild Research Laboratories, Technical Report 654, 1984.

[Schmolze & Lipkis 83] James Schmolze and Thomas Lipkis, "Classification in the KL-ONE Knowledge Representation System," in *Proceedings of the Eigth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

[Sidner 85] Candace L. Sidner, "Plan parsing for intended response recognition in discourse," *Computer Intelligence 1*, 1985.

[Sondheimer 84] Norman K. Sondheimer, Ralph M. Weischedel, and Robert J. Bobrow, "Semantic Interpretation Using KL-ONE," in *Proceedings of Coling 84*, pp. 101-107, Association for Computational Linguistics, July 1984.

[Tou et al. 82] Tou, F.F., M.D. Williams, R. Fikes, A. Henderson, and T. Malone, "RABBIT: An Intelligent Database Assistant," in *Proceedings AAAI-82*, pp. 314-318, AAAI, 1982.

[Vilain 84] Mark Vilain, *KL-TWO, A Hybrid Knowledge Representation System*, Bolt, Beranek, and Newman, Technical Report 5694, 1984.

[Wilczynski 84] David Wilczynski and Norman Sondheimer, *Transportability in the Consul System: Model Modularity and /acquisition*. 1984.

[Wile 81] David S. Wile, *POPART: Producer of Parsers and Related Tools System Builders' Manual*. 1981.