EMPIRICAL INVESTIGATION OF KNOWLEDGE REPRESENTATION SERVERS: DESIGN ISSUES AND APPLICATIONS EXPERIENCE WITH KRS

Brian R Gaines

Knowledge Science Institute, University of Calgary, Alberta, Canada T2N 1N4
gaines@cpsc.ucalgary.ca

Abstract

The design and application of KRS, a knowledge representation server in the KL-ONE, KRYPTON, CLASSIC family, are described. The server is designed as an open architecture module that can be used as a stand-alone service or embedded in other systems. It accepts the constraints necessary to make subsumption and recognition tractable, and maintains a careful distinction between definitions and assertions. It is implemented as a class library in an object-oriented language using generic, reusable objects. The approach taken to the integration of the server with external knowledge representation servers of similar or dissimilar types is analyzed. The server supports reasoning with exceptions and incomplete data through computation of a three-valued subsumption relation that is able to determine the possibility of further inferences if more assertions are made. KRS demonstrates that it is now possible to develop knowledge representation servers on a par with floating-point arithmetic units and numeric libraries, as modules with well-defined functionality and fast, reliable performance. It is also proving a useful tool for the empirical investigation of some large-scale knowledge representation server applications.

Introduction

The practical application of knowledge representation and reasoning systems presents a major opportunity for empirical research on fundamental issues of artificial intelligence. However, practicality often involves a balance of pragmatism and principles, with the pragmatism winning out if the principles lead to significance performance degradation. This introduces artefacts in empirical research based on many practically applied AI tools. As illustrated in the recent studies of Clancey [10] and Compton and Jansen [11], rational reconstruction of significant applications has become an important paradigm in overcoming such artefacts, but the empirical validation of such reconstructions itself requires reimplementation with more principled tools.

The utopian objective is the empirical study of significant applications of highly principled knowledge-based systems. This involves systems designed in such a way that maximum achievable performance is delivered through a transparent implementation that verifiably exhibits the principles under investigation. The problems of achieving this combination of significant application, deep principles and proper software engineering are not unique to artificial intelligence research—for example, operating systems, databases and real-time systems have all had similar needs for, and limitations on, empirical studies.

There have been two recent developments that open up new possibilities for meaningful empirical research in artificial intelligence: new insights into the complexity issues affecting knowledge representation and reasoning [7, 35]; and the development of object-oriented programming environments offering precise definition and efficient implementation of extensible abstract data types [38]. The advances in artificial intelligence allow requirements specifications to be developed for knowledge representation and reasoning systems with known, and practically applicable, performance curves [4]. The advances in software engineering allow these specifications to be implemented cleanly in such a way that their conformance with the requirements can be verified and their algorithms can be optimized to achieve the best possible performance.

The series of knowledge representation schemes and tools commencing with KL-ONE [8], developing through KRYPTON [6] and currently culminating in systems such as CLASSIC [4, 5] and LOOM [26] has reached a maturity of technology which offers the promise of knowledge representation 'services' in Levesque's terminology [24]. The logical foundations of the subsumption relation, techniques for its correct and complete calculation, and the interaction between representation power and the tractability of subsumption algorithms have been widely studied [7, 32, 33, 35]. It is now feasible to develop knowledge representation servers on a par with floating-point arithmetic units and numeric libraries, as software (and perhaps ultimately hardware) modules with well-defined functionality and fast, reliable performance. As with arithmetic units, such servers by no means solve all the problems of a particular application domain, but they do greatly reduce the burden of system development, allowing effort to be focused on the specifics of particular systems rather than on what should be general utilities.

The application requirement that motivated the development of the knowledge representation server reported in this paper was that of *knowledge acquisition*, from human experts and through induction from datasets, for the development of knowledge-based systems. Acquisition has long been recognized as a major bottleneck in expert system development [23], and research on knowledge acquisition has developed a massive literature in recent years [2, 3, 17, 28] We have developed knowledge acquisition toolkits combining graphic knowledge editing, repertory grid elicitation, conceptual clustering and empirical induction [13, 19, 20] that have been widely used both in other research groups and, industrially, as commercial products [18, 34]. The generality and range of applications of such toolkits has been limited by the representation and reasoning schema used (that of entity-attribute databases with categorical data types and of production rules). Problems have also arisen in managing the complexity of integrating many different techniques within the toolkits [1].

The system reported in this paper is the result of a complete redesign of these knowledge acquisition toolkits as a set of principled modules communicating with, and through, a very general knowledge representation server. The modules and the server have been implemented as a class library in C++, both for practical reasons of portability and speed, and also for the potential to develop a simple architecture based on the use of a few reusable classes whose operation is very clear and can be easily verified. A number of the new acquisition tools have now been used on a range of projects, and by other developers, for applications including the rational reconstruction of some large-scale knowledge-based systems.

This paper reports on the design and application of a knowledge representation server, KRS, that took CLASSIC as its initial model, maintained the terminology/assertion distinction of KRYPTON, and emphasized software engineering goals of clarity, portability, modularity, scalability and integrability. Even with these practical goals, features of the implementation raise a number of theoretical issues, some of which are deep both in terms of logic and philosophy. Subsumption-based knowledge representation servers are essentially implementations of intensional logics [25, 41] of abstract objects [12, 30] and design decisions at the coding level have to be guided by a proper implementation of such logics, some aspects of which are still open questions [29, 43]. Well-structured computer code is a medium supporting empirical investigation of the relevant theoretical issues. Minimal implementations in code, data and time resources usually correspond to natural underlying algebraic and logical structures, and non-obvious design decisions often correspond to underlying philosophical problems.

Knowledge representation server requirements

The design goals for KRS were:

- To offer a representation system that is general-purpose and rich enough in its descriptive capabilities to serve in a wide variety of applications.
- To derive correctly and completely the deductive consequences of constraints expressed in the representation, accepting such limitations on the constraints expressible as are necessary to achieve this.
- To act as a light-weight service to other modules that use heuristics to overcome these limitations at the expense of correctness or completeness.
- To be cleanly structured so that data functionality, structures, and algorithms are visible and easily checked.
- To have a well-defined interface that is callable from a wide range of languages so that the server is readily embedded in other systems.
- To support a number of knowledge bases, closely and loosely coupled, homogeneous and heterogeneous—this raises significant implementation and theoretical issues that will be discussed in detail.
- To support both small and large knowledge bases, with ease of modification of some underlying data structures if necessary.
- To be portable at the code and dataset level, both conceptually and practically.
- To re-use as much existing general-purpose code as possible, for example for set manipulation, and to make as much of the additional code as possible re-usable for other purposes—maintenance and portability are both supported by the use and generation of building bricks rather than special-purpose code.
- To offer a range of supporting utilities, such as parsers and graphers, on a modular basis as options separable from the knowledge representation kernel.

Figure 1 shows the architecture of KRS. It is designed to manage multiple standard knowledge bases and to interface to other servers either of the same or differing architectures.

The various sub-modules attached to the server are options offering specific additional functionality. The modules are (reading clockwise from the top left):

- Interface modules to other formal knowledge bases and knowledge representation servers, including databases.
- A hypermedia module allowing informal knowledge structures in text and images to be captured, accessed and linked. The linkage structure itself is held as a knowledge base.
- A text analysis module allowing documents to be analyzed in terms of word usage, and associations between significant words to be graphed—based on TEXAN in KSS0 [13] This enables protocols and technical documents to be used to initiate knowledge acquisition.
- A repertory grid expertise transfer module allowing graphic definition of concepts and graphic creation and editing of individuals—based on the elicitation screens of KSS0 [20].
- A conceptual clustering module allowing interactive definition of new concepts—based on the hierarchical and spatial clustering from KSS0.
- A knowledge editing module allowing the interactive development and editing of knowledge structures through a visual language.
- A conceptual induction module creating rules about specified subsets of individuals and transforming them to a minimal set of concepts and default rules—based on the Induct algorithm [14].
- A problem solving module supporting frame, rule and case-based inference from the knowledge structures.
- A grapher laying out specified parts of the concept subsumption graph, concept structures and individual structures—based on an incremental layout algorithm by Watanabe [40] that supports the production of clear visual knowledge structures.
- A language interface accepting and generating definitions and assertions in formal knowledge representation languages, both textual and visual.

The language interface supports a table-driven top-down recursive-descent compiler for CLASSIC-like languages. The tables can be changed dynamically in a prelude to the compiler allowing a syntax that makes clear distinctions between definitions and assertions to be used in examples, and a less verbose language to be used for system development. The same tables are used to output definitions and assertions in the language so that graphically entered concepts and individuals may be described, stored and ported to other systems in a textual form. Like all the peripheral modules, the language interface module is easily changed for another, and there is no implied commitment to the current language.

The language interface also supports visual knowledge representation languages similar to those of the graphic knowledge editor but allowing offline drawings to be parsed as formal specifications. This is useful for standard ontologies which may be presented effectively using drawing tools but where the drawing themselves are also the formal specification.

Data structures

Figure 2 shows the objects used in the server implementation and their data structures. The basic class library provides variable-length record objects which are sub-classed to provide dictionaries, sets and record objects for concept and individual records. The objects are:

- *Concept records* hold the constraint defining a concept: its primitives and those disjoint from them; its extensions as the subsets of individuals that must be, and may be, included in any subset recognized; its cardinality and co-reference constraints on any subset recognized; its rule and release concepts, if any, which will be asserted of any individual recognized, or impossible to recognize, respectively; and, for each role constrained in the concept, the concept constraining it.

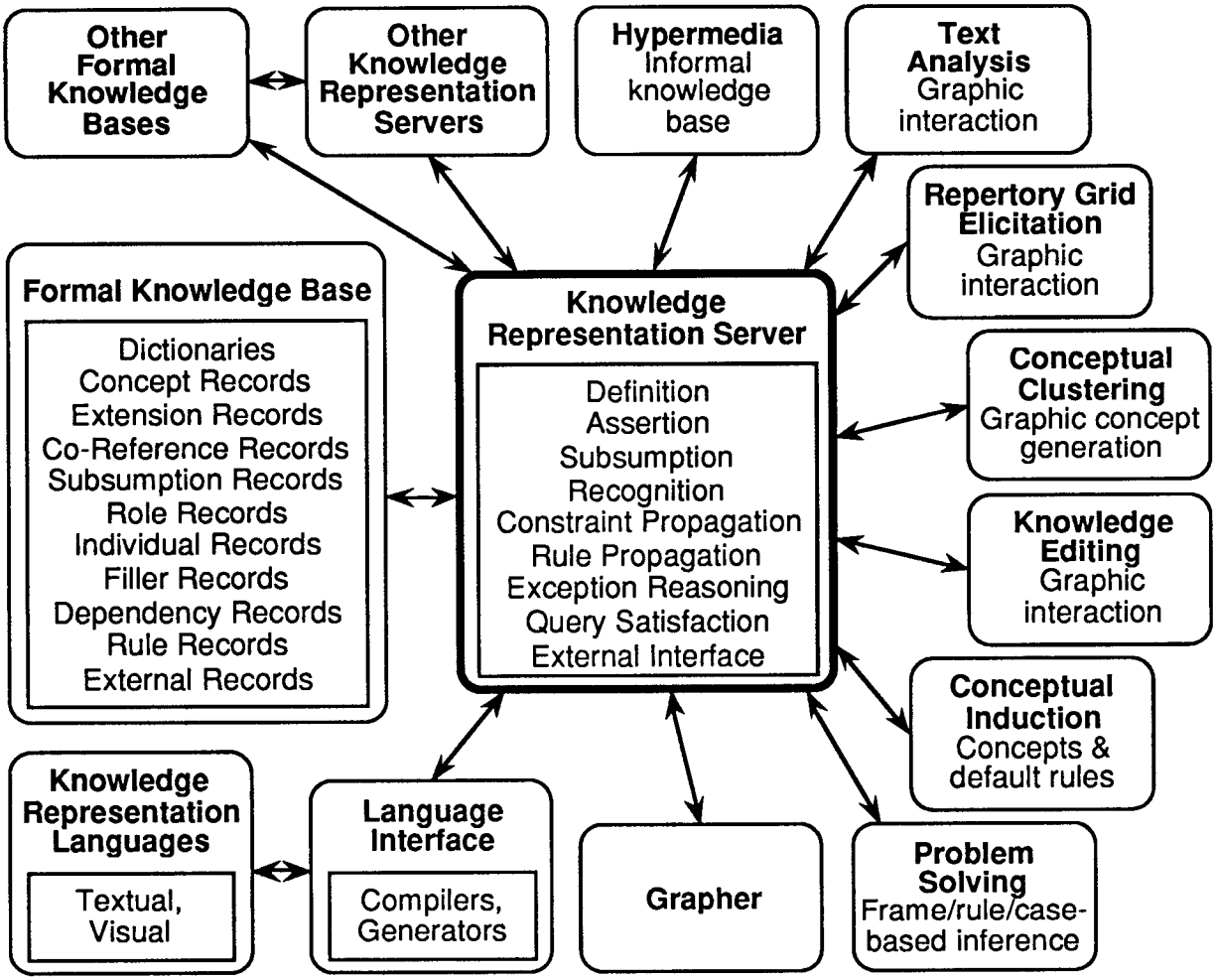

Fig.1 Architecture of the knowledge representation server

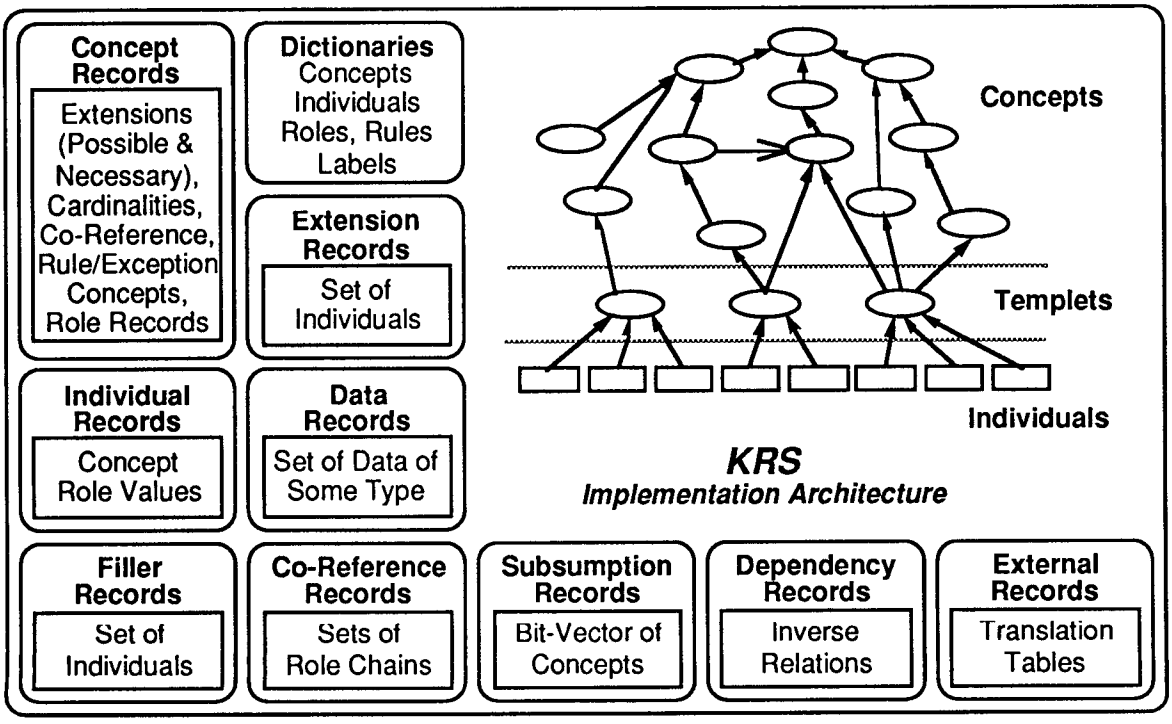

Fig.2 Data structures in the knowledge representation server

- *Individual records* hold the templet concept defining intensional constraints upon an individual and, for each role constrained in the templet or filled in the individual, a specification of the role fillers.
- *Filler records* hold subsets of individuals that fill slots.
- *Dictionaries* recognize concept, role and individual names and generate an accession number used to index other structures. Syntactically there is a clear separation between concepts, roles and individuals and a lexical item can occur in more than one dictionary with different meanings.
- *Extension records* hold subsets of individuals, typically those that may occur in a role filling set and those that must occur in it.
- *Data records* hold subsets of external individuals that are explicitly recognized by the server, notably integers, reals, dates and strings.
- *Co-reference records* hold equivalence subsets of role chains expressing co-reference constraints.
- *Subsumption records* hold computed subsumption relations on a three-valued basis, that one concept subsumes another, does not, or cannot.
- *Dependency records* keep track of inverse relations between individuals, supporting explicit inverse roles, efficient rule propagation and efficient retraction.
- *External records* hold tables translating between internal and external concepts and individuals.

A number of optimizations are applied in the implementation to minimize storage and maximize speed. For example, special codes are used for the empty, universal and singleton sets such that the set objects never store them, and subsumption is calculated in such a way that only a fast lookup is needed to check subsumption between role concepts rather than a recursive call.

One advantage of the object implementation is that these optimizations can be put in override methods in sub-classes so that the kernel implementation is small and easily checked but potentially slow, and the full implementation can be checked against the kernel for correctness and optimized for speed. This supports the requirement in empirical studies of ensuring that what is implemented conforms to the theory being tested.

The server supports the knowledge representation facilities defined for CLASSIC [4], with the following extensions:

- Automatic filling of *inverse roles*.
- *Negative set definitions* can be used for extensions, for example, that a role cannot be filled by red or blue. This is technically simple, causes no increase in complexity, and leaves subsumption well-defined, but substantially increases representation power, particularly in applications where the open-world semantics are significant.
- *Integer, real, date* and *label* (string) data types are fully supported and the accompanying range constraints are incorporated in subsumption computation.
- *Rules* are named objects supporting exceptions.
- Subsumption and recognition are *three-valued*, true, false and open.

The implementation separates facts, individual conceptual constraints, and inferences. This makes rule propagation and retraction fast and simple, and enables light-weight 'possible worlds' to be supported. It is also very significant in the knowledge acquisition context in enabling the sources of responses to queries to be shown clearly to users.

Editing concept definitions

KL-ONE-style knowledge structures are difficult to edit because they are require the 'top-down' definition of general concepts and their stepwise refinement to more specific concepts. This is reasonable in well-established domains where the ontology has been developed as part of a professional discipline, but it is rarely a suitable basis for knowledge acquisition in novel domains. Experts often develop their knowledge structures 'bottom-up' from cases through commonality to general ontologies, and find a 'top-down' approach unnatural. There are a number of ways of overcoming this problem—the conceptual clustering and induction tools develop concept hierarchies 'bottom-up' from cases—but general concept editing services are difficult to define.

The problem is to design an interface allowing the person editing to adequately express his or her intentions. For example, if a concept is edited it is simple enough to maintain referential integrity by disallowing use of concepts defined after it in the definition, and even this constraint can be relaxed in a more intelligent editor that places a new or edited concept in an appropriate place in the referential order if one exists. However, it is difficult to determine whether the user intends, when editing, that subsumption relations be preserved and hence that edits should be propagated, and, if so, whether the user really wants full propagation or the option of checking what is occurring.

The BLIP knowledge acquisition tool developed at the Technical University of Berlin to support 'sloppy modeling' [31] provides a conceptual system development and editing tool that computes all consequences of definitions dynamically, including induction from cases. For the initial stages of knowledge acquisition even this may be too strong in that experts often generate inconsistent knowledge structures but are not ready to resolve the inconsistencies until they have a developed a fairly large-scale model giving them an overall perspective on the task and domain.

The KRS graphic knowledge editing tool shown on the center right of Figure 1 supports this relaxed approach [16]. Knowledge structures can be developed and edited using drawing tools providing a natural visual language. The structures can extend across multiple screens supporting modularity and ontology 'libraries.' Structures can be freely duplicated and definitions may be fragmented across screens. The consistency of the definitions is determined only when they are compiled into a formal knowledge structure and inconsistencies are graphically highlighted.

Figure 3 shows a screen from the graphic knowledge editor on which the minimal five rule default reasoning solution of Cendrowska's [9] contact lens prescription problem has been developed. This screen compiles directly into a set of assertions and rules in KRS as shown in Figure 4. The top diagram on the screen defines the concept of a patient with possible attributes and values. The center diagram defines the concepts of a lens recommendation and three potential recommendation. The lower diagram defines the default rules.

The grapher is highly interactive. Objects, and groups of objects, may be dragged to new locations and the lines remain connected. Objects may be double-clicked to select them for editing—both text and object type may be edited. Popup menus associated with each object (see upper right of Figure 3) enable the connecting lines and arrows to be entered simply.

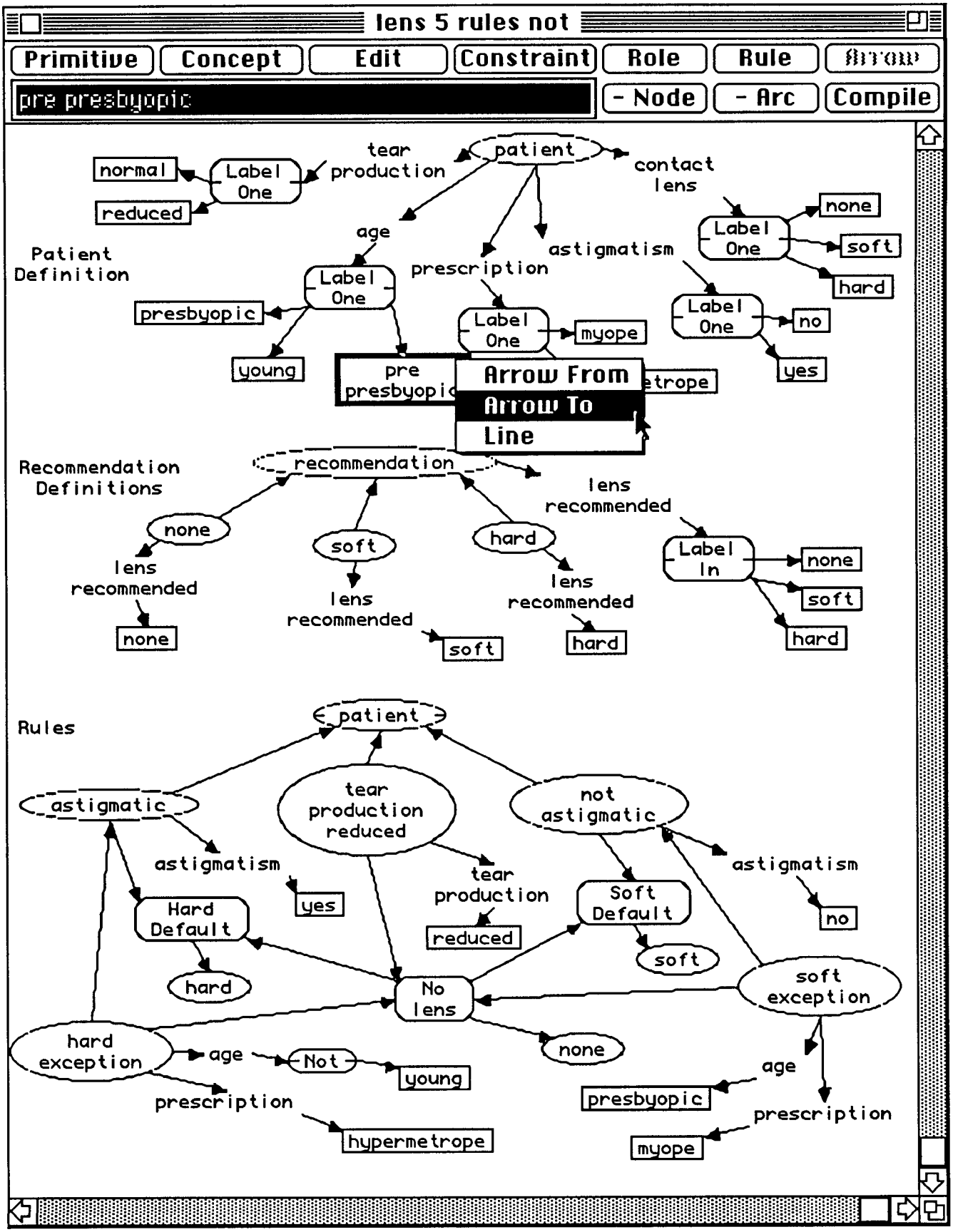

Fig.3 Graphic knowledge editor

```
Primitive(patient,
  (All tear production, (Label One normal, reduced))
  (All astigmatism, (Label One yes, no))
  (All contact lens, (Label One none, hard, soft))
  (All age, (Label One young, presbyopic, pre presbyopic))
  (All prescription, (Label One myope, hypermetrope))
)
Concept(recommendation,
  (All lens recommended, (Label In none, hard, soft))
)
Concept(none, recommendation,
  (All lens recommended, (Include none))
)
Concept(soft, recommendation,
  (All lens recommended, (Include soft))
)
Concept(hard, recommendation,
  (All lens recommended, (Include hard))
)
Concept(astigmatic, patient,
  (All astigmatism, (Include yes))
  (Then Hard Default)
)
Concept(not astigmatic, patient,
  (All astigmatism, (Include no))
  (Then Soft Default)
)
Concept(tear production reduced, patient,
  (All tear production, (Include reduced))
  (Then No lens)
)
Concept(hard exception, astigmatic,
  (All prescription, (Include hypermetrope))
  (All age, (Not young))
  (Then No lens)
)
Concept(soft exception, not astigmatic,
  (All prescription, (Include myope))
  (All age, (Include presbyopic))
  (Then No lens)
)
If(Hard Default, hard)
If(Soft Default, soft)
If(No lens, none)
Unless(No lens, hard, soft)
```

Fig.4 Resultant knowledge base compiled

The visual language used in Figure 3 is precisely defined. Concepts are ovals, primitive concepts are ovals with small horizontal lines inside each side, individuals rectangles, roles (and annotation) unboxed text, rules rounded-corner boxes, constraint expressions rounded-corner boxes with small horizontal lines. Lines without arrows connecting primitive concepts denote that the concepts are disjoint, and connecting roles that they are inverse. The interpretation of the arrows is overloaded but well-defined by the types of the objects at their head and tail, e.g.:

concept → concept	definitional subsumption
concept → role → concept	definitional role with conceptual constraint
concept → role → constraint	definitional role with extensional, cardinality or numeric constraint
constraint → individual	extensional constraint
individual → concept	asserted constraint on individual
individual → role → individual	asserted value of role for individual
concept → rule → concept	production rule
rule → rule	first rule exception to second

The compiler traces concepts and individuals through their outgoing arrows, builds a dependency table for concepts, highlighting circular links that it will not use, and generates concept definitions and individual assertions.

Reasoning with exceptions

One important application of the server in knowledge acquisition is to support the restructuring of rules inductively derived from elicited datasets as a minimal set of concepts and rules that can be presented to the expert in an easily understood form for checking and validation. This involves reasoning with exceptions since the best understood presentation is usually the most commonly occurring situations as the major rules with the rarer situations as exceptions. KRS supports reasoning with rules that are exceptions to other rules (represented in the grapher by an arrow from the the exception rule to the other rule). Recognition of an individual as falling under a concept that triggers the exception rule implies that the rule to which it is an exception will not be triggered.

For example, in the knowledge structure defined in Figure 3, the top level default rules are that an individual recognized as 'astigmatic' has 'lens recommended' set to 'hard', and one recognized as 'not astigmatic' has 'lens recommended' set to 'soft.' These are superseded by 'none' if the patient is recognized by the exception concepts, 'tear production reduced', 'hard exception', or 'soft exception.'

Open-world semantics are fully supported in rule triggering and exception computation in that the recognition of an individual by a concept triggering rules is checked to be true, impossible, or open. Both the true and the open conditions act to prevent the rules to which it is an exception from being triggered. This ensures that the basic reasoning in KRS is monotonic and logically sound. It also allows default reasoning and assumption-based truth maintenance systems to be added to the server without affecting its basic operation. The set of open concepts for each individual is well-defined and a default reasoning system can commence with the assumption that none will become satisfied in such a way as to act as exceptions to other rules. If this leads to a contradiction, 'no-good' combinations of concept/individuals may be developed.

An extension of the KRS rule system has been developed which implements rules not as a separate primitive but as a side-effect of a set aggregation primitive termed an 'intensional role' [15]. The fillers of intensional roles are defined by a concept, rather than extensionally by a prescribed set of individuals. The left hand side of a rule corresponds to the concept defining the role fillers, and the right hand side to the conceptual constraint on the role.

Integration

An obvious criterion against which to test a server is its ease of integration with other systems. KRS is accessible at three levels, as an extensible C++ class library, as an embedded, callable module, and as a server that may be accessed through messages using an inter-process communication protocol which also operates across a network. Many of our own applications involve the further development of the class library, but there have also been a number of applications by other system developers that use KRS strictly as a server.

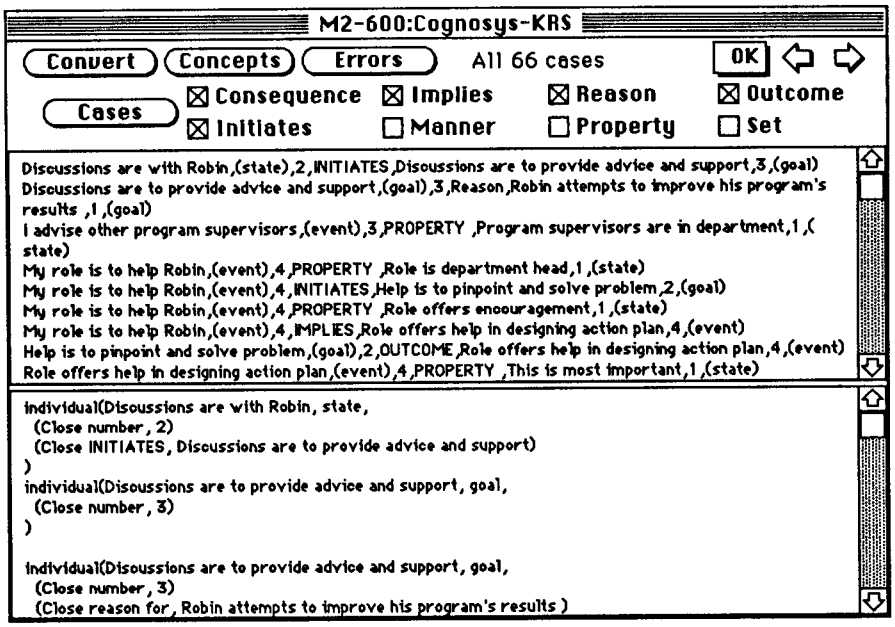

Fig.5 Cognosys data and query generator for calls to KRS server

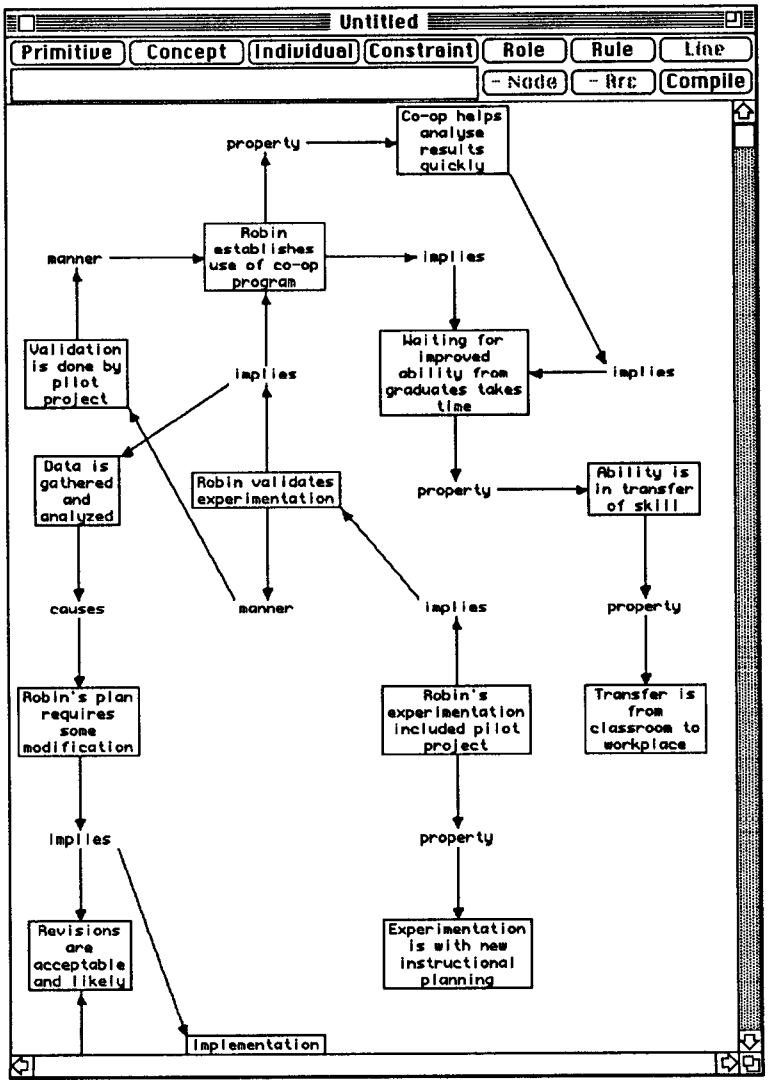

Fig.6 Some data displayed as a result of a Cognosys query

Figures 5 and 6 show KRS integrated with Woodward's [42] Cognosys application, a knowledge acquisition tool designed to analyze free text forms of expert protocols and interviews in terms of Graesser and Clark's [22] *implicit knowledge structures*. These involve four types of knowledge object among some combinations of which there can be up to eight kinds of relations. Cognosys provides a wide variety of graphic interactive tools for the analysis of the data structures but did not have the means for checking, applying and displaying them.

To make use of KRS, a module was written within Cognosys in which the Graesser and Clark structures were represented as concepts in KRS's intermediate language, and the data structures in terms of them were represented as individuals. Inter-process communication was used to transmit these to the KRS server together with user-generated requests to graph individuals satisfying certain relations, or the conceptual structure, or errors in its application determined by type-checking rules written for KRS. Reverse communication from the server to Cognosys allows interaction with the individuals in the KRS grapher to retrieve the original text protocols in Cognosys. Figure 5 shows a version of the Cognosys query generator for interaction with KRS in a server mode, including the original data structures and their translation to KRS intermediate language. Figure 6 shows part of a knowledge structure graphed in KRS in response to a Cognosys query. The graph layout here is fully automatic, designed to maximize the use of space and minimize line crossings.

Similar interfaces have been written for Shaw's repertory grid knowledge acquisition tools [37] where the concepts and cases elicited and rules induced are communicated as a knowledge structure to KRS and may be graphed or used in inference. The repertory grid tools already had export capabilities for a number of expert system shells and it was possible in this case to regard KRS as such a shell. In terms of adequacy of representation and speed of inference it was well-suited to the heuristic classification type of knowledge bases generated through the repertory grid approach. The grapher proved effective as a second-stage knowledge acquisition tool enabling a number of separate knowledge bases, each elicited for some small, coherent part of a domain, to be linked together to form an overall domain model.

Empirical experience in different problem domains

It is interesting to examine KRS as an empirical research tool—what is being learnt from its applications? This section gives three examples exemplifying different aspects of empirical research.

Analysis of constructive intractability results

Some of the intractability results for term subsumption logics in the literature are constructive in offering a 'worst-case' example whose impact on a working system can be investigated empirically. It is of interest to determine the use of space and time in such problems. Nebel [33] gives a set of concept definitions where computation of subsumption is intractable because of the growth of anonymous concepts. Figure 7 shows an example set of concepts in KRS input language commencing with a primitive concept C12 and going through a sequence of concepts down to C0 defined in terms of preceding concepts. Figure 8 shows the total concepts generated, the time in seconds and the space in bytes for Nebel's examples 0 through 20 with KRS running on a Macintosh IIci. The derived time and space per concept are useful performance parameters in estimating how large a concept space KRS and similar tools can cope with in more realistic problem examples.

```
primitive(C12)
concept(C11, (all r1, C12) (all r2, C12))
concept(C10, (all r1, C11) (all r2, C11))
concept(C9, (all r1, C10) (all r2, C10))
concept(C8, (all r1, C9) (all r2, C9))
concept(C7, (all r1, C8) (all r2, C8))
concept(C6, (all r1, C7) (all r2, C7))
concept(C5, (all r1, C6) (all r2, C6, C12))
concept(C4, (all r1, C5) (all r2, C5, C10))
concept(C3, (all r1, C4) (all r2, C4, C8))
concept(C2, (all r1, C3) (all r2, C3, C6))
concept(C1, (all r1, C2) (all r2, C2, C4))
concept(C0, (all r1, C1) (all r2, C1, C2))
```

Fig.7 Nebel's intractability example (No.12 with 13 concepts)

No.	Concepts	Time sec	Space bytes	Time/Conc	Space/Conc
0	2	0	1,304	0	652
1	3	0	1,392	0	464
2	5	0	1,501	0	300
3	7	0	1,606	0	229
4	11	1	1,825	0	166
5	15	1	2,040	0	136
6	23	1	2,490	0	108
7	31	1	2,925	0	94
8	45	2	3,899	0	87
9	59	4	4,714	0	80
10	86	8	6,699	0.1	78
11	114	15	8,927	0.1	78
12	163	30	13,654	0.2	84
13	213	61	18,298	0.3	85
14	298	128	28,828	0.4	97
15	384	264	39,805	0.7	104
16	533	610	65,635	1.1	123
17	686	1,284	97,724	1.9	142
18	958	3,046	166,987	3.2	174
19	1,241	6,967	260,698	5.6	210
20	1,701	15,029	459,732	8.8	270

Fig.8 KRS performance on Nebel's 'worst-case' examples

Performance with large rule sets

The Garvan Institute of Medical Research thyroid diagnosis knowledge base, GARVAN-ES1, has been in use for many years, covering some 60 diagnoses through some 650 rules and having associated case datasets of over 20,000 patients. It provides a realistic test case for 'expert system' applications and is of particular interest as it is being re-analyzed by a number of researchers using different techniques.

In particular, Compton and Jansen [11] have used it to examine the improvements in knowledge acquisition and maintenance possible with their 'ripple-down rules' representation based on a subsumption hierarchy with dependencies between rules at the same level. The KRS rule system fully supports the ripple-down technique—the rule set becomes a single chain in which the first rule to fire acts as an exception to all others.

The ripple-down version of the Garvan rules, 547 rules, together with 669 cases evaluated on 17 attributes, were loaded and run in KRS on a Macintosh II. The total number of concepts generated was 1035, and the space occupancy was 315Kbytes, of which 86K was for concepts, 141K for the subsumption bit-map and 58K for the individual cases. The diagnoses took 0.4 seconds each on average.

Performance in a resource allocation application

Some of the interesting questions which may be investigated empirically about knowledge representation servers with restricted semantics is how much they can achieve in themselves in a certain application domain, how much has to be added in additional reasoning modules, and how domain-specific are these additional modules. To encourage comparative evaluation of different tools on the same problem, the European Knowledge Acquisition Workshop in 1990 instigated Project Sisyphus in which a collection of knowledge-based system development tasks are being circulated to tool developers and users. This section reports on our experience with a room configuration task submitted from the GMD in Germany which is being worked on there and at other sites under an ESPRIT project [39].

The problem is one of allocating individuals to rooms under a set of, possibly conflicting, constraints, many of which involve relations between types of individual. The data set consists of sample cases and protocols from an expert solving some of them. The problem is an interesting one for a term subsumption knowledge representation system since it goes well beyond simple classification and involves some kind of planning under constraints.

The KRS grapher proved a very effective tool in entering the basic knowledge structures since the organization chart, building layout, and so on, have a natural graphical representation which it was easy to mimic in the grapher. This made validation of the knowledge structures particularly simple. Figure 9, with the grapher in 'fit to window' mode, shows the conceptual structures at the top, facts about a sample case in the center, and the problem-solving strategies at the bottom. Figures 10 and 11 are more readable full-size images of part of the knowledge structures.

It was not clear at the start of this project how much contribution KRS could make to the problem solving component. The GMD group had attached a constraint satisfaction engine to a KL-ONE-like knowledge representation system in their solution. It was remarkably easy, however, to program most of the solution within KRS without additional features in a number of different ways. The cleanest solution seems to be a blackboard, or agenda, mechanism, in which the classification of people posts requests on an agenda which are satisfied by the classification of rooms. The symmetry of this consumer-supplier arrangement is particularly attractive in dealing with peculiarities where the logic reverses, such as finding another smoker to fill a room to which a smoker has already been allocated.

The agenda mechanism was programmed as a set of concepts, individuals and rules in KRS. The only external component involved in the solution is a simple selection mechanism that accepts solutions placed on the agenda and implements them by asserting that a person is allocated to a room. This needs an external mechanism to choose between alternative solutions. However, this choice involves no 'intelligence', only the capability to make a succession of arbitrary choices and to backtrack by retraction should a sequence lead to an impasse in which no further solutions are possible, that is, a simple search procedure.

On the KRS side the only extension to CLASSIC-like features which was essential was the provision of automatic inverse role filling. Abstractly, inverse roles allow sets to be formed of individuals satisfying concepts (essentially implementing the axiom of comprehension in set theory), and this turns out to be an important problem-solving strategy for this class of problems.

Conclusions

The main objective of this paper has been to demonstrate that open architecture knowledge representation servers are straightforward to implement using an object-oriented language, and offer opportunities for the empirical investigation of principled artificial intelligence in practical application. The object-oriented implementation of the server with the functionality and optimizations described is fast and storage efficient, and was easily implemented. We have run it with knowledge bases of several thousand concepts and individuals from a variety of sources including the GARVAN-ES1 described above, our sports knowledge bases [21], and a variety of artificial datasets designed to stress the code.

Keeping subsumption up-to-date appears to be a negligible overhead even with very large numbers of concepts. Recognition has not been a problem with the knowledge acquisition datasets that tend to consist of numbers of self-contained knowledge islands. However, it is possible to define knowledge structures that involve substantial propagation over large sets of individuals each time a new assertion is made and this would cause interactivity problems if it was a significant application requirement. This is a domain problem rather than one of server implementation and it would be reasonable to expect it if the system were used to investigate, for example, the mathematics of algebraic structures. Similar problems would arise if configurational problems involving many interacting constraints were represented for solution in the system.

The implications of accepting the restrictions necessary to make subsumption and recognition tractable in the server have still to be investigated in the context of knowledge acquisition. So far, KRS provides such a major increase in representation capabilities compared with previous knowledge acquisition systems that it has not been a major issue. MacGregor [27] and Schmolze [36] have remarked that users want more representation power, not less. Our design philosophy is that the analogy to arithmetic units holds, and it is appropriate to develop clean, complete and fast units with limited capabilities, and then supplement them with other units providing additional functionality ranging from general theorem provers to highly domain-specific modules. This is particularly important in a knowledge acquisition environment where one has to present a clean conceptual model and user interface to users who may never have used a computer before and are attempting to develop overt knowledge structures where none previously existed.

Acknowledgements

This work was funded in part by the Natural Sciences and Engineering Research Council of Canada. I am grateful to Paul Compton for access to the Garvan ES-1 and to Mildred Shaw for critical comments on this paper.

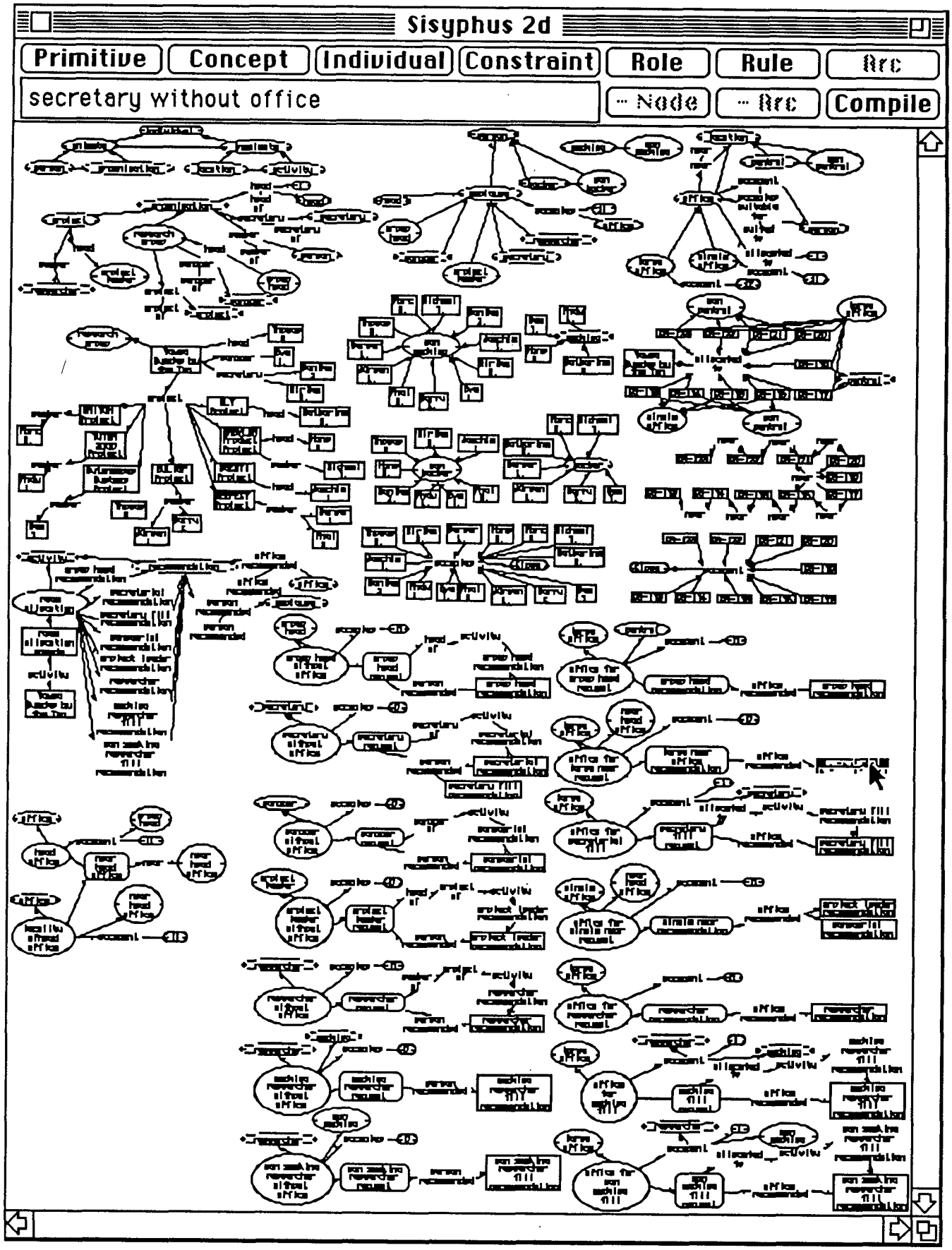

Fig.9 Room allocation knowledge structures

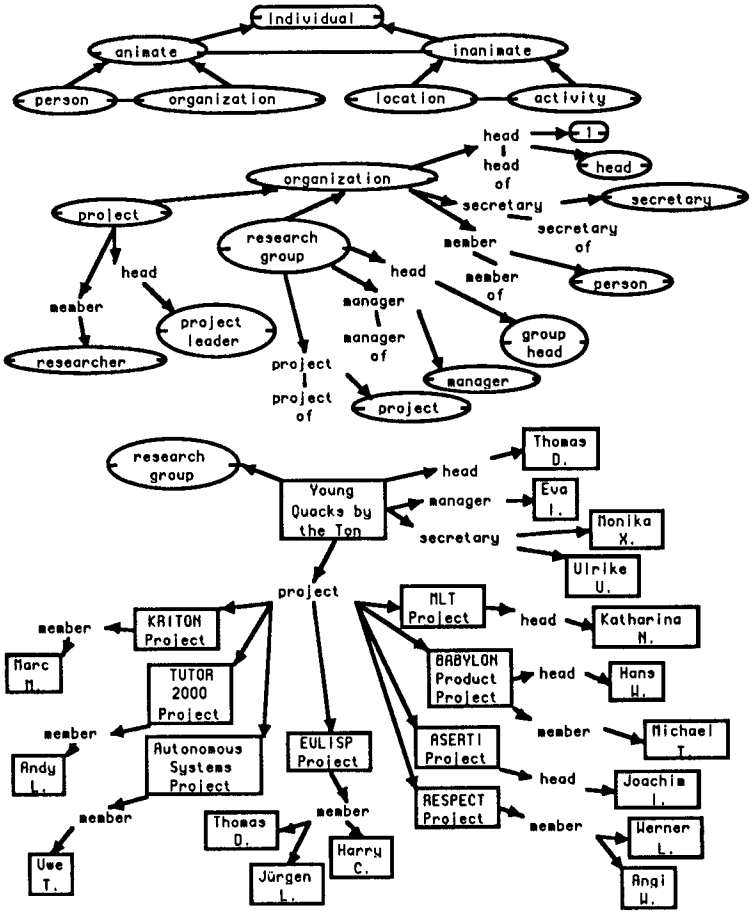

Fig.10 Top-level concepts, concepts of organization, and instance of an organization (from Fig.9)

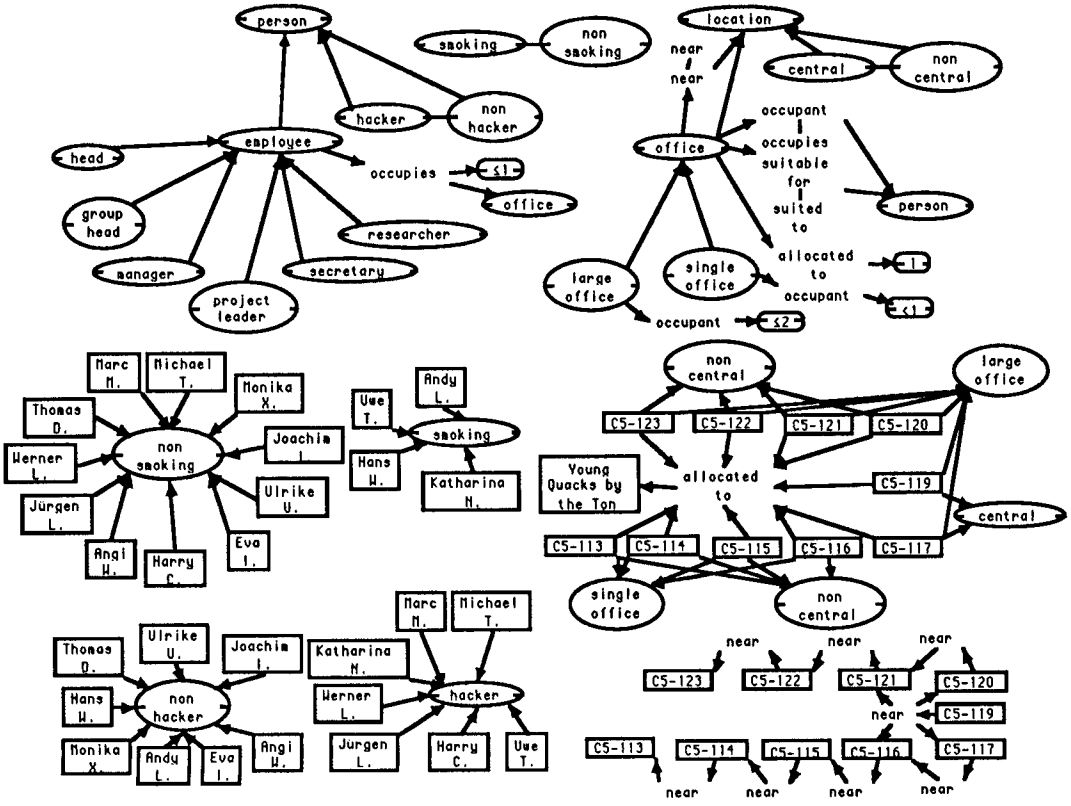

Fig.11 Concepts of person and location, and instances (from Fig.9)

References

1. Boose, J.H. & Bradshaw, J.M. (1987). Expertise transfer and complex problems: Using AQUINAS as a knowledge acquisition workbench for knowledge-based systems. **Int. Journal of Man-Machine Studies, 26,** 3-28.
2. Boose, J.H. & Gaines, B.R., Eds. (1988) **Knowledge Acquisition Tools for Expert Systems.** Academic Press.
3. Boose, J.H. & Gaines, B.R., Eds. (1990) **The Foundations of Knowledge Acquisition.** London, Academic Press.
4. Borgida, A., Brachman, R.J., McGuiness, D.L. & Resnick, L.A. (1989) CLASSIC: a structural data model for objects. Clifford, J., Lindsay, B. & Maier, D., Eds. **Proceedings of 1989 ACM SIGMOD International Conference on the Management of Data.** pp.58-67. New York: ACM Press.
5. Brachman, R. J., Borgida, A., McGuinness, D. L., and Resnick, L. A. (1989) The CLASSIC knowledge representation system, or, KL-ONE: The next generation. **Workshop on Formal Aspects of Semantic Networks.** Santa Catalina Island, California.
6. Brachman, R. J., Gilbert, V.P. & Levesque, H. J. (1985) An essential hybrid reasoning system: knowledge and symbol level accounts of KRYPTON. **Proceedings of IJCAI85.** pp.547-551. Los Altos: Morgan Kaufmann.
7. Brachman, R. J., & Levesque, H. J. (1984) The tractability of subsumption in frame-based description languages. **Proceedings of AAAI-84.** pp.34-37. San Mateo, California: Morgan Kaufmann.
8. Brachman, R.J. & Schmolze, J. (1985) An overview of the KL-ONE knowledge representation system. **Cognitive Science,** 9(2) 171-216.
9. Cendrowska, J. (1987) An algorithm for inducing modular rules. **International Journal of Man-Machine Studies 27** (4), 349-370.
10. Clancey, W.J. (1990) Implication of the system-operator metaphor for knowledge acquisition. **Proceedings of the First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop: JKAW'90.** pp.413-430. Tokyo: Ohmsha.
11. Compton, P. & Jansen, R. (1990) A philosophical basis for knowledge acquisition. **Knowledge Acquisition** 2(3).241-258.
12. Fine, K. (1985) **Reasoning with Arbitrary Objects.** Oxford: Basil Blackwell.
13. Gaines, B.R. (1988) Knowledge acquisition systems for rapid prototyping of expert systems. **INFOR, 26**(4), 256-285.
14. Gaines, B.R. (1989) An ounce of knowledge is worth a ton of data: quantitative studies of the trade-off between expertise and data based on statistically well-founded empirical induction. **Proceedings of the Sixth International Workshop on Machine Learning.** pp.156-159. Morgan Kaufmann.
15. Gaines, B.R. (1991). Integrating rules in term subsumption knowledge representation servers. **AAAI'91: Proceedings of the Ninth National Conference on Artificial Intelligence.** Menlo Park, California: AAAI Press.
16. Gaines, B.R. (1991). An interactive visual language for term subsumption visual languages. **IJCAI'91: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.** San Mateo, California: Morgan Kaufmann.
17. Gaines, B.R. & Boose, J.H., Eds. (1988) **Knowledge Acquisition for Knowledge-Based Systems.** London, Academic Press.
18. Gaines, B.R. & Linster, M. (1990). Integrating a knowledge acquisition tool, an expert system shell and a hypermedia system. **International Journal of Expert Systems Research and Applications** to appear.
19. Gaines, B.R. & Shaw, M.L.G. (1989) Comparing the conceptual systems of experts. **Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.** pp.633-638. Los Angeles, California: Morgan Kaufmann.
20. Gaines, B.R. & Shaw, M.L.G. (1990). Eliciting knowledge and transferring it effectively to a knowledge-based systems. **IEEE Transactions on Knowledge and Data Engineering** to appear.
21. Gaines, B.R. & Vickers, J.N. (1988) Design considerations for hypermedia systems. **Microcomputers for Information Management,** 5(1), 1-27.
22. Graesser A.C. & Clark, L.F. (1985). **Structures and Procedures of Implicit Knowledge.** New Jersey: Ablex.
23. Hayes-Roth, F., Waterman, D.A. & Lenat, D.B., Eds. (1983) **Building Expert Systems.** Reading, Massachusetts: Addison-Wesley.
24 Levesque, H. J. (1984) A logic of implicit and explicit belief. **Proceedings of AAAI-84.** pp.198-202. San Mateo, California: Morgan Kaufmann.
25. Maida, A.S. & Shapiro, S.C. (1982) Intensional concepts in propositional semantic networks. **Cognitive Science,** 6(4) 291-330.
26. MacGregor, R.M. (1988) A deductive pattern matcher. **Proceedings of AAAI88.** pp.403-408. Morgan Kaufmann.
27 MacGregor, R.M. (1989) The evolving technology of the KL-ONE family of knowledge representation systems. **Workshop on Formal Aspects of Semantic Networks.** Santa Catalina Island, California.
28 Marcus, S. (Ed.) (1988) **Automating Knowledge Acquisition for Expert Systems.** New York: Kluwer.
29. Mendelsohn, R.L. (1989) Objects and existence: reflections on free logic. **Notre Dame J. of Formal Logic, 30**(4) 604-623.
30. Meinong, A. (1983) **On Assumptions.** Berkely: University of California Press.
31. Morik, K. Ed. (1985) **Knowledge Representation and Organization in Machine Learning.** Berlin: Springer-Verlag.
32. Nebel, B. (1988) Computational complexity of terminological reasoning in BACK. **Artificial Intelligence, 34** 371-383.
33. Nebel, B. (1990) **Reasoning and Revision in Hybrid Representation Systems.** Berlin: Springer-Verlag.
34. Rappaport, A. & Gaines, B.R. (1988). Integrating knowledge acquisition and performance systems. *Proceedings of the Australian Joint Artificial Intelligence Conference.* pp.317-336. Adelaide.
35. Schmidt-Schauss, M. (1989) Subsumption in KL-ONE is undecidable **Proceedings of KR'89: First International Conference on Principles of Knowledge Representation and Reasoning.** pp.421-431. Morgan Kaufmann.
36. Schmolze, J. (1989) Terminological knowledge representation systems supporting n-ary terms. **Proceedings of KR'89: First International Conference on Principles of Knowledge Representation and Reasoning.** pp.432-443. San Mateo, California: Morgan Kaufmann.
37. Shaw, M.L.G. & Gaines, B.R. (1987) KITTEN: Knowledge Initiation and Transfer Tools for Experts and Novices. **International J.of Man-Machine Studies 27**(3), 251-280.
38. Shriver, B. & Wegner, P. (1987) **Research Directions in Object-Oriented Programming.** Massachusetts: MIT Press.
39. Voß, A., Karbach, W., Drouven, U., Lorek, D. & Schuckey, R. (1990) Operationalization of a synthetic problem. **ESPRIT Basic Research Proj. P3178 REFLECT Task I.2.1 Report.**
40. Watanabe, H. (1989) Heuristic graph displayer for G-BASE. **International Journal of Man-Machine Studies, 30**(3) 287-302.
41. Woods, W.A. (1975) What's in a link: Foundations for semantic networks. Bobrow, D.G. & Collins, A.M. (Eds) **Representation and Understanding: Studies in Cognitive Science.** pp.35-82. New York: Academic Press.
42. Woodward, J.B. (1990) Knowledge engineering at the front-end: defining the domain. **Knowledge Acquisition, 2,** 73-94.
43. Zalta, E.N. (1988) **Intensional Logic and the Metaphysics of Intentionality.** Cambridge, Mass.: MIT Press.