KNOWLEDGE REPRESENTATION SERVERS: A GENERIC TECHNOLOGY FOR KNOWLEDGE ACQUISITION AND KNOWLEDGE-BASED SYSTEMS

Brian R Gaines, Killam Memorial Chair, & Director, Knowledge Science Institute

University of Calgary, Alberta, Canada T2N 1N4

gaines@cpsc.ucalgary.ca

The semantics and implementation structure of 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 default reasoning through computation of a three-valued subsumption relation that is able to determine the possibility of further inferences if more assertions are made. 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. Such servers provide a generic knowledge representation technology for knowledge acquisition and knowledge-based systems.

1 INTRODUCTION

Adequate knowledge representation is an essential requirement for all areas of artificial intelligence research and development. In recent years there have been major advances in the theory and practice of KR systems originating from semantic nets. In particular the series of KR schemes and tools commencing with KL-ONE [6], developing through KRYPTON [4] and currently culminating in systems such as CLASSIC [2, 3] and LOOM [30] has reached a maturity of technology which offers the promise of KR 'utilities' or 'services' in Levesque's [27] terminology. 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 [5, 37, 42] and are becoming reasonably well-defined. It is now feasible to develop KR 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' domains have high degrees of generality. Subsumption-based KR schemes offer many-sorted logics and operations upon them that are known to be important components of effective theorem provers [21], machine learning algorithms [29], default reasoning systems [41], commonsense reasoning [26], knowledge acquisition [36] and so on. The motivation for the work in this paper has been to make use of developments in knowledge representation to place a clean, fast and effective KR server at the heart of a very general knowledge support system combining a wide and heterogeneous range of knowledge acquisition techniques [14]. One of the frustrations common to many areas of research in AI is the amount of effort necessary to develop adequate general-purpose KR systems before the 'real work' can begin. Similar considerations apply to porting systems that have a high degree of generality of potential application but whose KR sub-systems have been written to be domain-specific and are inextricably intertwined with other functionality.

This paper reports on the implementation of a KR server, KRS, that took CLASSIC as its initial model, meticulously maintained the terminology/assertion distinction of KRYPTON, and emphasized software engineering goals of clarity, portability, modularity, scalability and integratibility. It presents guidelines for the development of KR servers that will simplify the tasks of others in implementing such systems. 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 KR servers are essentially implementations of intensional logics [28, 47] of abstract objects [10, 33] 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 [32, 48]. Well-structured computer code is itself a language for communicating about 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.

2 KNOWLEDGE REPRESENTATION SERVER REQUIREMENTS

The design goals for a KR server are:

- 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 KR kernel.

The initial implementation of KRS has been in a dialect of C++ but most of the design considerations apply to any implementation and we have parallel developments in SmallTalk, OBJ3, Lisp and Prolog. The use of an object-oriented language is important for its modularity rather than any intrinsic contribution to the object-orientation of the resultant knowledge base [9]. However, the data structures and methods of the implementation objects are very important in the role of the code as a communication medium.

3 FUNDAMENTAL MODEL OF THE SERVER

There are at least four conceptual frameworks for the design of a KR server:

- A logical framework which maps concepts, roles and individuals into some logical system based on first-order predicate calculus, modal logic, free logic [24, 25, 32] or some form of intensional logic of abstract objects [10, 39, 48]—these logical links are significant but none of those available matches exactly the features of a CLASSIC-like system—however, the known capabilities and problems of these logics generate very important tests of the design.
- A philosophical framework which views the system as providing ontological and epistemological facilities for representing and reasoning about the world—again inadequate, but detailed metaphysical schemas such as that of Grossman [20] provide a test for missing features.
- A cognitive framework which views the system as emulating human knowledge and belief systems and mental processes—again inadequate but providing a range of test data including mentalistic analysis of logics [23, 46] and models of the dynamics of belief systems [27].
- An algebraic framework which views the system as expressing and manipulating constraints on sets.

The algebraic framework currently provides the simplest definition capturing the semantics and structure of a CLASSIC-like system. The basic structures are:

- A set, I, of *individuals*
- A set, F, of *fillers* defined as $\mathcal{P}\mathrm{I} \cup \omega$, where $\mathcal{P}\mathrm{I}$ is the powerset of I and ω represents an open filler
- A set, R, of *roles* with cardinality r
- A *filler values* mapping, V, from I to the Cartesian product, P, of r filler sets, $V : I \rightarrow \prod_{r} F$, with a projection operator R_n from $p \in P$ to its n'th component, $p_n \in \mathcal{P}\mathrm{I}$ [$R_n V(i)$ is the set of role fillers for role R_n in individual i]
- A poset, C, of *concepts* ordered such that for some $c, d \in C$, $c < d$ [the order relation is used to represent the lack of self-reference in concept definitions]
- A *filler restrictions* mapping, S, from C to the Cartesian product, Q, of r members of C, $S : C \rightarrow \prod_{r} C$, with a projection operator R_n from $c \in C$ to its n'th component, $c_n \in C$, and

such that $R_nS(c) < c$ [$R_nS(c)$ is the concept applicable to role fillers for role R_n in concept c]

- A set of *constraints*, E, on subsets of I, with a basic recognition relation, ρ, between constraints in E and subsets of I such that there is a well-defined entailment relation, $\Rightarrow$, generated on E by the condition that $\forall$ e, f $\in$ E, e $\Rightarrow$ f iff $\forall$ $p \in \mathcal{P}$, e$\rho p \supset$ fρp [the constraints specify some test of the existence, cardinality or the set membership of the subset, or that all members satisfy one set of arbitrarily assertable predicate (primitives) and do not satisfy another set of these predicates (those primitives disjoint from the first set)—note that $\forall$ e, e$\rho\omega$ is false, that is, open fillers cannot satisfy a constraint]
- A *constraints* mapping, D, from C to E [D(c) is the primary constraint applicable to a subset of I, *p*, to be recognized as being a c with secondary constraints defined recursively for the role filler subsets of i $\in$ *p*, that $R_nV(i)$ is recognized as being a $R_nS(c)$]
- A *templets* mapping, T, from I to C [T(i)$\in$ C is the concept providing an intensional templet for i$\in$ I]
- A *subsumption* relation, $\Rightarrow$, extending the entailment relation on constraints to one on concepts such that $\forall$c, d $\in$ C, c $\Rightarrow$ d iff D(c) $\Rightarrow$ D(d) and for $1 \leq n \leq r$ $R_nS(c) \Rightarrow R_nS(d)$ [the partial order, <, on C ensures this recursive definition is determinate]
- A *recognition* relation, ρ, between concepts in C and subsets of I extending the basic recognition relation between E and $\mathcal{P}$ such that $\forall c \in C$, $p \in \mathcal{P}$, cρp iff D(T(i)) $\Rightarrow$ D(c) or this is not so but D(c)ρp and for $1 \leq n \leq r$ $R_nS(T(i)) \Rightarrow R_nS(c)$ or a particular entailment is not so but $R_nS(c)\rho R_nV(i)$
- An auxiliary *recognition* relation, ρ, between concepts in C and individuals I extending that between C and $\mathcal{P}$ such that $\forall c \in C$, i $\in$ I, cρi iff $\exists$ $p \in \mathcal{P}$ such that i $\in$ *p* and cρp [this is necessary because recognition has been defined in terms of subsets of I because of cardinality and set inclusion constraints]

Subsumption is expressed as an intensional relation to allow it to be computed for concepts that do not have extensions and correspond to 'impossible objects' [39]. The definition corresponds to the normal one for concepts with extensions. The tractability of subsumption in CLASSIC-like representation systems corresponds to the restriction of constraints to be expressible in a non-interacting canonical form such that they may be independently tested. The definitions of the subsumption and recognition relations correspond to a test of the *necessity* of an individual being recognized by one concept if it is recognized by another. They also allow for the *impossibility* of an individual being recognized by a concept (for example, typically because of cardinality or disjoint primitive constraints). Calculating this is important in plausible reasoning with open fillers in checking whether additional information might change the current conclusion.

Note that cardinality constraints have been included in the definition of concepts and that concepts are taken to apply to sets of individuals. This is helpful in implementation because a single subsumption calculation may be applied when checking roles without a separate check of cardinality constraints. It also allows constraints on slot fillers involving cardinality to be fully expressed as named concepts. The algebraic definition is useful computationally because the primitives of sets, and constraints, lead to simple data structures and "obvious" algorithms. It is also a useful framework for proving completeness and tractability of subsumption algorithms [40].

4 KNOWLEDGE REPRESENTATION SERVER ARCHITECTURE

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 interface protocols to allow the standard server to integrate other servers fully into its subsumption calculations are discussed in a later section.

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 KR servers and knowledge bases, 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 [12, 15]. 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.
- 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 [13].
- 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 [45] that can be used interactively to support 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 our 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 particularly useful for standard ontologies which may be presented effectively using drawing tools but where the drawing themselves are also the formal specification.

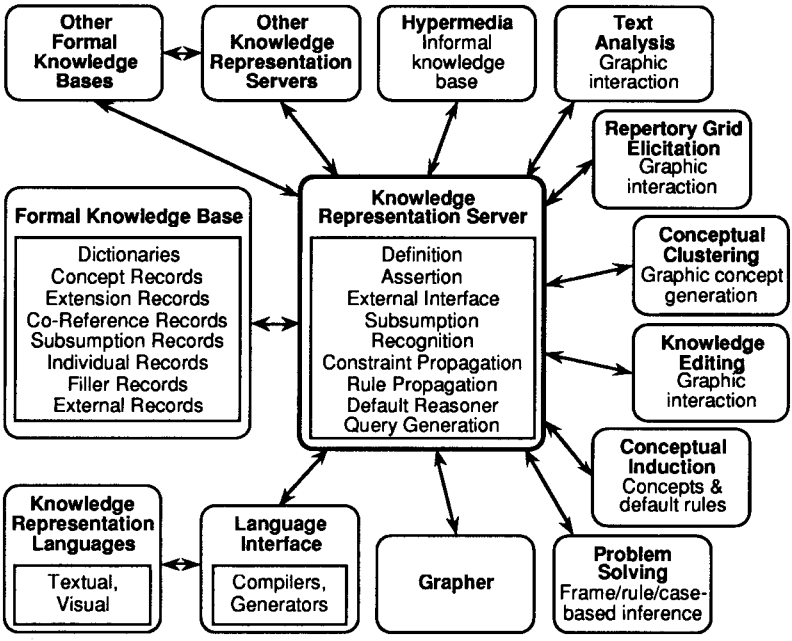

Fig.1 Architecture of the knowledge representation server

5 DATA STRUCTURES

Figure 2 shows the objects used in the server implementation and their data structures. Our 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.
- *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.

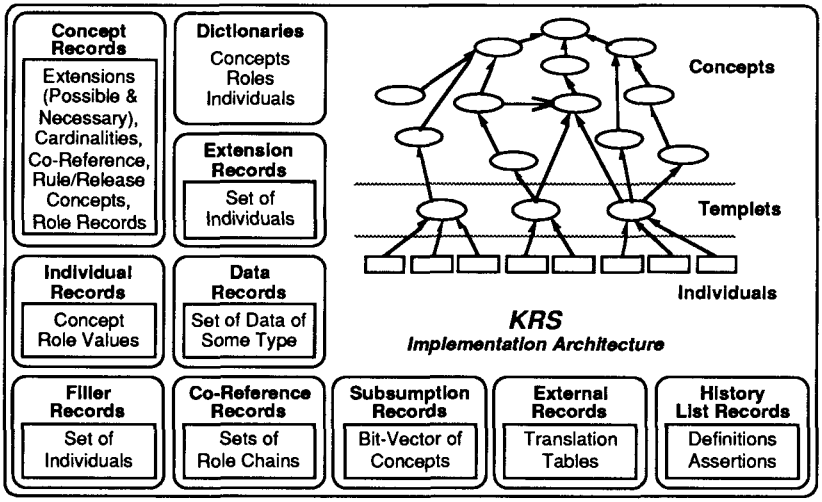

Fig.2 Data structures in the knowledge representation server

- *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.
- *External records* hold tables translating between internal and external concepts and individuals.
- *History list records* keep track of explicit and implicit definitions and assertions in such a way that the knowledge base may be rolled back to any previous state.

These data structures correspond very closely to the algebraic model of the server already described. 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.

The server supports the propagation of the consequences of definitions, assertions and retractions at several different levels:

- Extensional constraints in definitions can cause cardinality constraints to be adjusted.
- Co-reference constraints in definitions cause the corresponding role constraints at the ends of the role chains to be merged.
- Assertion and retraction of role fillers causes the role constraints to propagate to the filler individuals.
- Assertion and retraction of individual constraints causes any resulting role constraints to propagate to the existing filler individuals.
- The closure of a set of role fillers is computed based on defined or asserted constraints.

- Recognition of an individual by a concept with an attached rule causes the consequential concept of the rule to be asserted as applying to the individual.

The last three propagation conditions can interact and the server continues propagation until a fixed point is reached (which can be shown to be unique and well-defined). A good test of the server is MacGregor's [30] example of how such propagation enables the recognition of recursively defined concepts, such as lists, even though self-reference and forward reference is not allowed in concept definitions. The definition/assertion separation and the constraint propagation also satisfy Frixione, Gaglio and Spinelli's [11] requirements for the proper treatment of individual intensions.

6 INTEGRATION OF EXTERNAL KNOWLEDGE BASES

The practical requirements of knowledge acquisition, particularly the need to support elicitation from multiple experts in multiple interacting domains [15], and the support of heterogeneous knowledge bases combining many different sources and media, dictated that we pay careful attention to knowledge base integration. CLASSIC allows external individuals, such as integers, to be used as slot fillers and provides a TEST primitive that allows external concepts to be used to constrain these fillers, but does not expect external individuals to have roles or external concepts to support subsumption. The server described here supports both these extensions but does not require them, thus allowing it to operate correctly with a range of external knowledge support servers including other instances of itself.

It was a design requirement for KRS that a knowledge base within the system be partitionable into internal and external knowledge bases in such a way that no functionality changed. It is assumed that external knowledge bases have their own interfaces to which messages may be passed, and that they have various levels of capability of responding to queries. The keyword EXTERNAL in the server language is used to indicate messages to external knowledge bases and internal references to external concepts and individuals. It is followed by the name of the external knowledge base referenced, the 'realm' in CLASSIC terminology. The record structures of Figure 2 make provision for references to external concepts and individuals. An implementation distinction is made between slots filled with internal individuals only, those filled with external individuals in a single realm only (which are managed by the server for that realm) and heterogeneous fillers where individuals are mixed across realms (which are managed by the external records object as sets of individuals tagged by realm).

This system is effective for knowledge bases with individuals whose role fillers are from other knowledge bases. The only additional complication is that of supporting individuals part of whose record structures are maintained in another realm. For example, one might keep knowledge about cars in a car knowledge base, but their general roles as transportation vehicles and physical objects might be inherited from, and managed by, a more general knowledge base. It is not possible to determine the knowledge base referenced by context since roles may be designated for fillers that have not previously been specified in the templet, and the same role name may occur with different meanings in interacting knowledge bases.

The external interface assumes that external knowledge bases can answer queries about subsumption and recognition. If they cannot the queries are answered at the interface in terms of identity only which is the default in CLASSIC's TEST. The correctness and completeness of subsumption and recognition can be maintained provided the external server is able to support it within its realm. If not, only correctness will be maintained.

7 EDITING CONCEPT DEFINITIONS

KL-ONE-style knowledge structures are notoriously difficult to edit because they are based on 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 very 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 server currently provides an 'undo' and controlled 'redo' facility based on the history list. The knowledge-base can be rolled back to any previous state and then rolled forward with a modified history list. This is clean in implementation and reasonably ergonomic in terms of user activity. It does not satisfy all editing requirements but we do not yet have a good model of what this 'all' might be. An interesting theoretical foundation for the history-list approach in terms of the algebraic model is that an instance of the knowledge base may be taken as an automaton and the subsequent history list as its input-output behavior. There is then a category-theoretic adjunction between the knowledge base and its editing [19]. This approach is particularly insightful because it corresponds to the viewpoint of a user of an existing knowledge base attempting to change its structure by edits and assertions. The various constraints already existing may make the desired state unreachable. There is also a significant psychological literature concerned with this problem, for example, Kelly [22] distinguishes "core constructs" which an individual is highly reluctant to change despite evidence that they are inappropriate. In our terms, this corresponds to edits of early concepts where the propagation effects may be very difficult to control in terms of the user's intentions.

The BLIP knowledge acquisition tool developed at the Technical University of Berlin to support 'sloppy modeling' [35] 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 graphic knowledge editing tool shown on the center right of Figure 1 supports this relaxed approach. 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 [7] contact lens prescription problem has been developed. This screen compiles directly into the set of assertions and rules 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 visual language used in Figure 3 is precisely defined. Concepts are ovals, individuals rounded-corner boxes, roles (and annotation) unboxed text. The interpretation of the standard arrows with filled heads is well-defined by the types of the objects at their head and tail. Arrows with open heads denote rules. Lines without arrows connecting concepts denote that the concepts are disjoint. 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. Role objects are only noted when on paths and hence may also be used for annotation. The text analysis module generates such 'annotation' which may then be developed into a formal graphic specification by specifying objects as concepts and individuals.

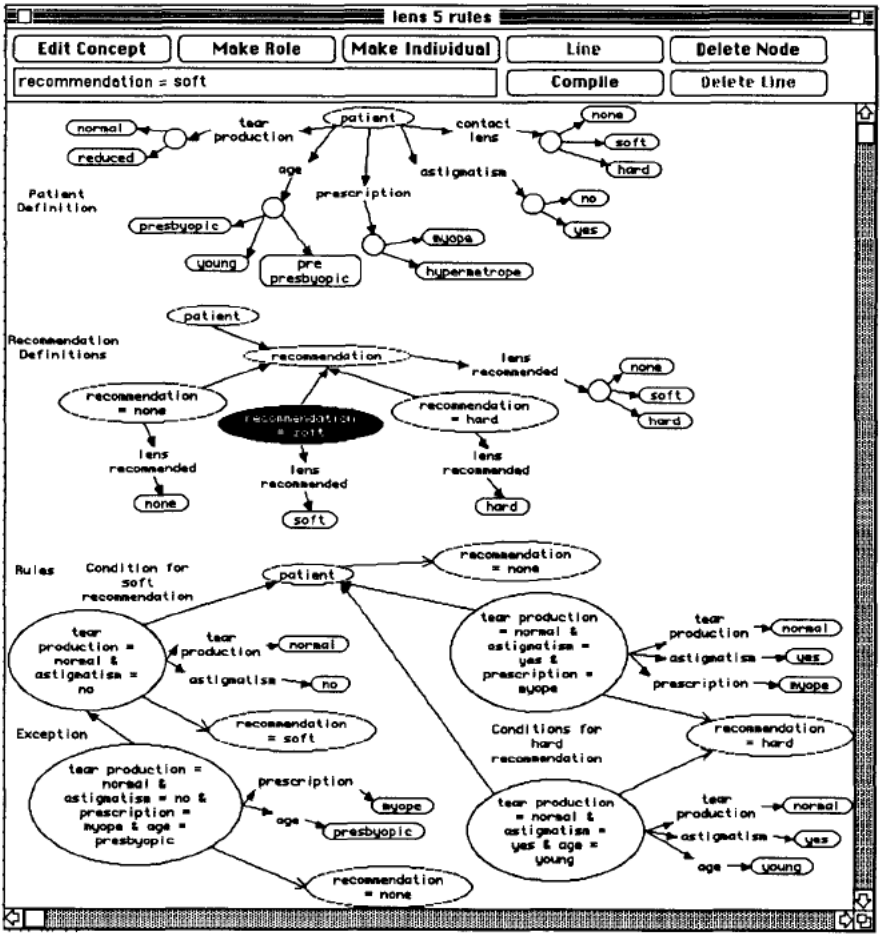

Fig.3 Graphic knowledge editor

```
concept(patient,
 (ALL tear production, (IN normal, reduced))
 (ALL astigmatism, (IN yes, no))
 (ALL prescription, (IN myope, hypermetrope))
 (ALL age, (IN young, pre-presbyopic, presbyopic))
 (ALL contact lens, (IN none, soft, hard))
)
concept(recommendation,
 patient,
 (ALL recommendation, (IN none, soft, hard))
)
concept(recommendation = none,
 recommendation,
 (ALL lens recommended, (INCLUDE none))
)
concept(recommendation = soft,
 recommendation,
 (ALL lens recommended, (INCLUDE soft))
)
concept(recommendation = hard,
 recommendation,
 (ALL lens recommended, (INCLUDE hard))
)
concept(tear production = normal & astigmatism = no,
 patient,
 (ALL tear production, (INCLUDE normal))
 (ALL astigmatism, (INCLUDE no))
)
rule(tear production = normal & astigmatism = no,
 recommendation = soft,
)
concept(tear production = normal & astigmatism = no & prescription = myope & age = presbyopic,
 patient,
 (ALL tear production, (INCLUDE normal))
 (ALL astigmatism, (INCLUDE no))
 (ALL prescription, (INCLUDE myope))
 (ALL age, (INCLUDE presbyopic))
)
rule(tear production = normal & astigmatism = no & prescription = myope & age = presbyopic,
 recommendation = none,
)
rule(patient, recommendation = none)
concept(tear production = normal & astigmatism = yes & age = young,
 patient,
 (ALL tear production, (INCLUDE normal))
 (ALL astigmatism, (INCLUDE yes))
 (ALL age, (INCLUDE young))
)
rule(tear production = normal & astigmatism = yes & age = young,
 recommendation = hard,
)
concept(tear production = normal & astigmatism = yes & prescription = myope,
 patient,
 (ALL tear production, (INCLUDE normal))
 (ALL astigmatism, (INCLUDE yes))
 (ALL prescription, (INCLUDE myope))
)
rule(tear production = normal & astigmatism = yes & prescription = myope,
 recommendation = hard,
)
```

Fig.4 Knowledge structures compiled from the diagrams of Figure 3

By contrast to the problems of consistent editing of concept definitions, the truth maintenance problem of propagating assertions and retractions of constraints and facts about individuals is relatively easy. There are still subtle problems, e.g. the same assertion may have been made a number of times through direct entry and various forms of propagation—is the user attempting to retract the fact or the overt assertion? However, these are easily overcome. The server tags assertions and inferences distinctly and maintains inverse role linkages. This makes it easy to propagate a retraction to potentially affected individuals, reset them and re-inference.

8 DEFAULT REASONING

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 default reasoning since the best understood presentation is usually the most commonly occurring situations as the major rules with the rarer situations as exceptions. CLASSIC in its current definition supports rules but not default reasoning, and there are known problems to simply incorporating exceptions as rules triggered by concepts lower in the subsumption hierarchy [18].

KRS fully supports default reasoning in that rules are checked by tracing the subsumption hierarchy and giving precedence to inferences made at lower levels. Thus, in the knowledge structure defined in Figures 3 and 4, the top level default rule is that an individual recognized as a 'patient' has 'lens recommended' set to 'none.' This is superseded by 'soft' if the patient is recognized by the subsumed concept, 'tear production = normal & astigmatism = no.' This, in its turn, is superseded by 'none' again if the patient is recognized by the subsumed concept, 'tear production = normal & astigmatism = no & prescription = myope & age = presbyopic' (see the bottom left part of Figure 3).

This reasoning is logically correct if all the facts about the patient are known. It only becomes default reasoning if the exception rule cannot be tested because some facts are unknown, for example, the patient's 'prescription' is not known. KRS can indicate whether an inference based on partial information is plausible or exact because it computes a three-valued subsumption relation which allows the system to recognize when no further assertions could make an exception rule trigger. This is termed a 'release' and may be viewed as a rule which triggers when it is impossible for an individual to satisfy a concept except through retraction. Releases may be used to manage the frame problem by discriminating between plausible reasoning based on defaults and exact reasoning based on adequate information. Thus a release attached to the concept 'tear production = normal & astigmatism = no & prescription = myope & age = presbyopic' will fire when either the 'prescription' is not 'myope' or the 'age' is not 'presbyopic' regardless of whether the other information is known or not.

The reasoning algorithm in KRS is efficiently implemented and operates rapidly even with large knowledge bases; for example, the Garvan Institute of Medical Research thyroid diagnosis knowledge base, GARVAN-ES1, consists of 650 rules [8]. The 'ripple-down rules' version of this was moved to KRS and checked on the cornerstone case database of some 650 cases, taking less than one second a case on an Apple Mac IIci.

9 KNOWLEDGE ACQUISITION METHODOLOGIES BASED ON THE SERVER

KSSn is a knowledge support system based on the knowledge representation server which uses the modules shown in Figure 1 to provide an integrated set of knowledge acquisition tools. It is designed to offer the text analysis, graphic elicitation, clustering and induction tools of KSS0 [12, 15], together with the wider range of data types and hierarchical data structures of AQUINAS [1], and the graphic knowledge editing of tools such as ONTOS [34] and CODE [44]. Figure 5 shows a knowledge acquisition methodology based on the appropriate sequence of use of the tools in KSSn.

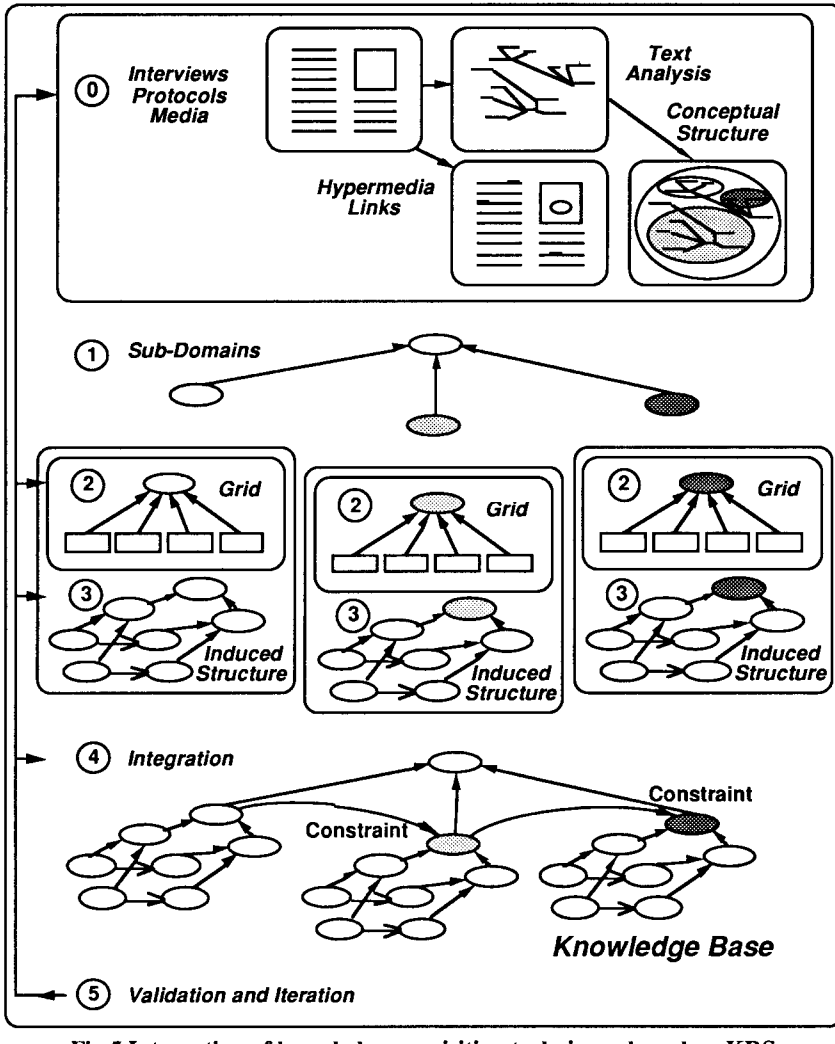

Fig.5 Integration of knowledge acquisition techniques based on KRS

- Stage ⓪ consists of gathering informal knowledge in the hypermedium system.
- Stage ① consists of using informal knowledge to elicit the major coherent sub-domains that together encompass the significant phenomena in the overall domain.
- Stage ② consists of using the repertory grid methodology in each sub-domain to elicit relevant attributes and critical cases.
- Stage ③ consists of using the conceptual induction methodology to derive concepts and rules from the grids.
- Stage ④ consists of linking the sub-domains together, generally by specifying as a constraint in one domain that some role in that domain has as value an individual in another.
- Stage ⑤ consists of testing the overall knowledge base as a performance system and iterating back through any of the earlier stages in order to develop and refine it further.

The architecture of Figure 5 is intended to be illustrative of the way the various knowledge acquisition techniques described naturally combine than a rigid framework for an integrated tool. For example, some sub-domain structures may be developed by direct editing, others by induction from databases, others by text analysis, and others by tools not yet defined or developed. The point is that techniques and tools already available come together naturally in this framework, and it also provides an open architecture for incorporating further techniques and tools. In the longer term, the architecture shown may need major extension, but currently it seems to encompass what is available to us in knowledge acquisition.

The knowledge representation server holds the knowledge structures involved at every stage of this process in a uniform fashion with full access to a range of editing, presentation and documentation tools. The associated modules for text analysis, repertory grid elicitation, conceptual induction, and so on, are relatively small and independent modules that are readily developed, changed and maintained. The use of a common underlying class library for data structures and the user interface supports the provision of a consistent data model and user interface throughout.

10 CONCLUSIONS

The main objective of this paper has been to demonstrate that open architecture knowledge representation servers are straightforward to implement now. The algebraic semantics and computational object structures described should help others undertaking such implementations. The C++ 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 in Section 7, our sports knowledge bases [17], and a variety of artificial datasets designed to stress the code. Nebel's [38] 'worst-case' example of the combinatorial explosion of anonymous concepts runs without problems—for example, his 20 concept example generates a total of 1,701 concepts and a total data structure of 460 Kbytes.

The space requirements for 100 concepts, 1,000 individuals, with an average of 20 roles of which 5 are multi-valued having an average of 3 slot fillers is 120 KB and scales up nearly linearly with the number of concepts and individuals. The small knowledge bases typical in the early stages of knowledge acquisition take only a few KB.

Keeping subsumption up-to-date appears to be a negligible overhead even with very large numbers of concept. 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 [31] and Schmolze [43] 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.

A companion paper [16] has surveyed developments in knowledge acquisition, their foundations, and some of the existing tools. Current third generation research is targeting tools for the 1990's to provide integrated knowledge acquisition environments supporting a wide range of complementary tools and techniques. This paper has described an architecture for such systems based on a general-purpose knowledge representation server. In particular, this architecture attempts to provide the richness of resources required while maintaining the strict software engineering disciplines necessary to the effective development and support of complex systems.

ACKNOWLEDGEMENTS

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada. I am grateful to Mildred Shaw for her critical comments on the first draft of this paper.

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. **International Journal of Man-Machine Studies, 26,** 3-28.

[2] 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.

[3] 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 (February).

[4] 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.

[5] 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.

[6] Brachman, R.J. & Schmolze, J. (1985) An overview of the KL-ONE knowledge representation system. **Cognitive Science, 9**(2) 171-216 (April-June).

[7] Cendrowska, J. (1987) An algorithm for inducing modular rules. **International Journal of Man-Machine Studies 27** (4), 349-370 (October).

[8] Compton, P. & Jansen, R. (1989) A philosophical basis for knowledge acquisition. **Proceedings of the Third European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW'89)** pp.75-89. Paris (July).

[9] De Troyer, O., Keustermans, J. & Meersman, R. (1986) How helpful is an object-oriented language for an object-oriented database model? **Proceedings of 1986 International Workshop on Object-Oriented Database Systems**. pp.124-132. Washington: IEEE Computer Society Press.

[10] Fine, K. (1985) **Reasoning with Arbitrary Objects**. Oxford: Basil Blackwell.

[11] Frixione, M., Gaglio, S. & Spinelli, G. (1989) Are there individual concepts? Proper names and individual concepts in SI-Nets. **International Journal of Man-Machine Studies, 30** 489-503.

[12] Gaines, B.R. (1988) Knowledge acquisition systems for rapid prototyping of expert systems. **INFOR, 26**(4), 256-285 (November).

[13] 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. San Mateo, California: Morgan Kaufmann (June).

[14] Gaines, B.R. (1989) Design requirements for knowledge support systems. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Fourth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop**. pp.12-1-12-20. Banff (October).

[15] 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 (August).

[16] Gaines, B.R. & Shaw, M.L.G. (1990) Foundations of knowledge acquisition. **Proceedings of the First Japanese Workshop on Knowledge Acquisition for Knowledge-Based Systems (JKAW'90)**. Kyoto, Japan (October).

[17] Gaines, B.R. & Vickers, J.N. (1988) Design considerations for hypermedia systems. **Microcomputers for Information Management**, 5(1), 1-27 (March).

[18] Ginsberg, M.L. (1987) **Readings in Non-Monotonic Reasoning.** San Mateo, California: Morgan Kaufmann.

[19] Goguen, J.A. (1973) Realization is universal. **Mathematical Systems Theory 6** 359-374.

[20] Grossman, R. (1983) **The Categorial Structure of the World**. Bloomington: Indiana University Press.

[21] Jackson, P., Reichgelt, H. & van Harmelen, F. (1989) **Logic-Based Knowledge Representation**. Cambridge, Massachusetts: MIT Press.

[22] Kelly, G.A. (1955) **The Psychology of Personal Constructs**. Norton: New York.

[23] Kroy, M. (1976) **Mentalism and Modal Logic**. Wiesbaden: Athenaion.

[24] Lambert, K. (1983) **Meinong and the Principle of Independence**. Cambridge: University Press.

[25] Leblanc, H. (1976) **Truth-Value Semantics**. Amsterdam: North-Holland.

[26] Lenat, D.B. & Guha, R.V. (1990) **Building Large Knowledge-Based Systems**. Reading, Massachusetts: Addison-Wesley.

[27] Levesque, H. J. (1984) A logic of implicit and explicit belief. **Proceedings of AAAI-84**. pp.198-202. San Mateo, California: Morgan Kaufmann.

[28] Maida, A.S. & Shapiro, S.C. (1982) Intensional concepts in propositional semantic networks. **Cognitive Science, 6**(4) 291-330.

[29] Manago, M. (1989) Knowledge intensive induction. **Proceedings of the Sixth International Workshop on Machine Learning.** pp.151-155. San Mateo, California: Morgan Kaufmann (June).

[30] MacGregor, R.M. (1988) A deductive pattern matcher. **Proceedings of AAAI88**. pp.403-408. San Mateo: Morgan Kaufmann.

[31] 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 (February).

[32] Mendelsohn, R.L. (1989) Objects and existence: reflections on free logic. **Notre Dame Journal of Formal Logic, 30**(4) 604-623.

[33] Meinong, A. (1983) **On Assumptions**. Berkely: University of California Press.

[34] Monarch, I. & Nirenburg, S. (1987) The role of ontology in concept acquisition for knowledge-based systems. **Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW'87)**. Reading University, UK (September).

[35] Morik, K. Ed. (1985) **Knowledge Representation and Organization in Machine Learning**. Berlin: Springer-Verlag.

[36] Murray, K. & Porter, B.W. (1988) Developing a tool for knowledge integration: initial results. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Third AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop**. pp.22-1-22-12. Banff (November).

[37] Nebel, B. (1988) Computational complexity of terminological reasoning in BACK. **Artificial Intelligence, 34** 371-383.

[38] Nebel, B. (1990) **Reasoning and Revision in Hybrid Representation Systems**. Berlin: Springer-Verlag.

[39] Parsons, T. (1980) **Non-Existent Objects**. New Haven: Yale University Press.

[40] Patel-Schneider, P.F. (1989) A four-valued semantics for terminological logics. **Artificial Intelligence, 38** 319-351.

[41] Reiter, R. (1978) On reasoning by default. **Proceedings of TINLAP-2, Theoretical Issues in Natural Language Processing.** pp.210-218. Champaign, Illinois: University of Illinois.

[42] 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. San Mateo, California: Morgan Kaufmann (May).

[43] 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 (May).

[44] Skuce, D., Shenkang, W. & Beauville, Y. (1989) A generic Knowledge Acquisition Environment for conceptual and ontological analysis. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Fourth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop**. pp.31-1-31-20. Banff (October).

[45] Watanabe, H. (1989) Heuristic graph displayer for G-BASE. **International Journal of Man-Machine Studies, 30**(3) 287-302 (March).

[46] Woods, J. (1974) **The Logic of Fiction**. The Hague: Mouton.

[47] 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.

[48] Zalta, E.N. (1988) **Intensional Logic and the Metaphysics of Intentionality**. Cambridge, Massachusetts: MIT Press.