OBJECT-ORIENTED IMPLEMENTATION OF A KNOWLEDGE REPRESENTATION SERVER

B R Gaines & D Freedman
Knowledge Science Institute, University of Calgary
Calgary, Alberta, Canada T2N 1N4
(gaines@cpsc.ucalgary.ca, (403) 220-5901)

Abstract: The object-oriented implementation of a knowledge representation server is 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 a C++ subset using general-purpose reusable objects. The paper argues that the object-oriented approach is important in making it 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

INTRODUCTION

Adequate knowledge representation is an essential requirement for all areas of artificial intelligence. The series of KR schemes and tools commencing with KL-ONE (Brachman & Schmolze, 1985), developing through KRYPTON (Brachman, Gilbert & Levesque, 1985) and currently culminating in systems such as CLASSIC (Borgida, Brachman, McGuinness & Resnick, 1989) has reached a maturity of technology which offers the promise of KR 'utilities' or 'services' in Levesque's (1984) terminology. This paper reports on the implementation of a KR server 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.

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 correctly and completely derive 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.

Our initial implementation has been in a subset of C++ but most of the design considerations apply to any implementation and we have parallel developments in SmallTalk, OBJ3, Lisp and Prolog. An object-oriented language is important for its modularity rather than any intrinsic contribution to the object-orientation of the resultant knowledge base (De Troyer, Keustermans & Meersman, 1986). However, the data structures and methods of the implementation objects are very important in the role of the code as a communication medium.

KNOWLEDGE REPRESENTATION SERVER ARCHITECTURE

Figure 1 shows the architecture of the "standard" KR server being described. 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:

- A knowledge acquisition module allowing graphic definition of concepts and graphic creation and editing of individuals—based on our KSS0 system (Gaines, 1988).
- A role and individual clustering module allowing interactive definition of new concepts—based on the hierarchical and spatial clustering from KSS0.
- An empirical induction module creating rules about specified subsets of individuals and transforming them to a minimal set of concepts and default rules (Gaines, 1989b).
- An analogical reasoning module generating minimum common generalizations and co-referential determinations—based on Leishman's (1989) system.
- A grapher laying out specified parts of the concept subsumption graph, concept structures and individual structures—based on an incremental layout algorithm by Watanabe (1989) that can be used interactively to support the production of clear visual knowledge structures.
- A language interface accepting and generating definitions and assertions in a CLASSIC-like language.

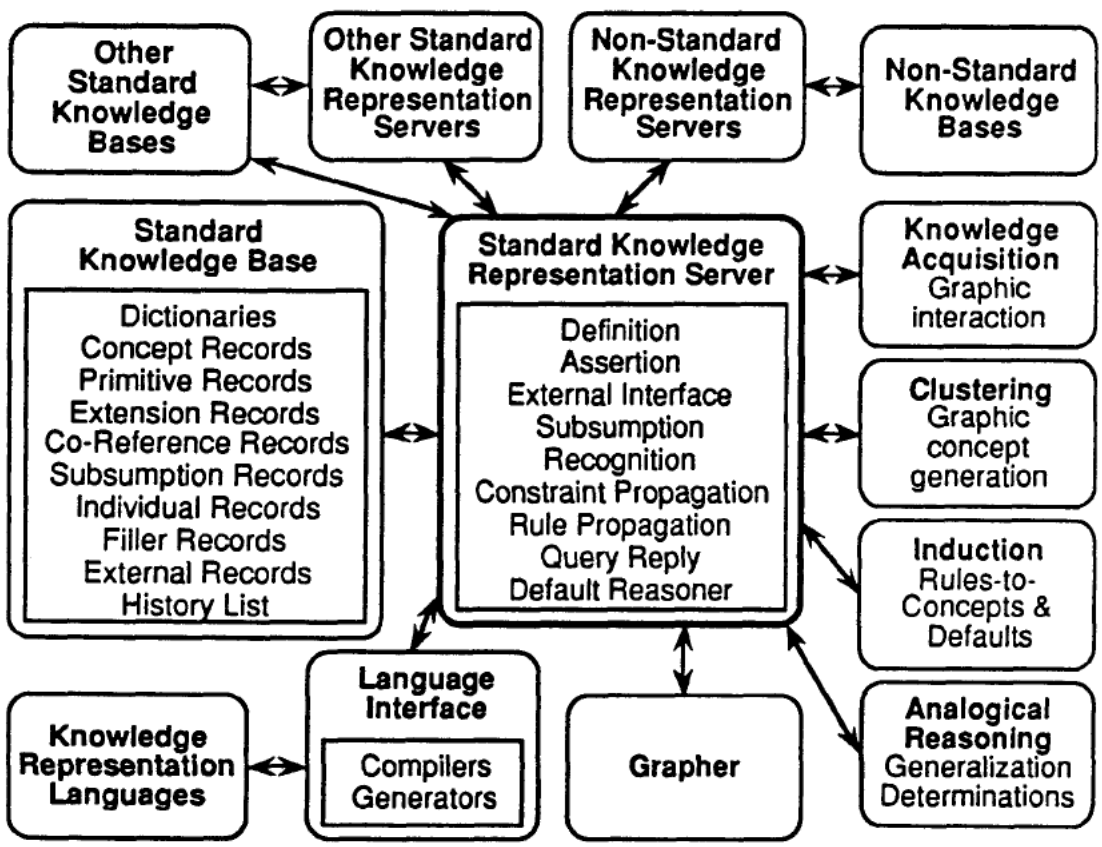

Fig.1 Architecture of the knowledge representation server

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 concept, if any, which will be asserted of any individual recognized; and, for each role constrained in the concept, the concept constraining it.
- *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.
- *Primitive records* hold subsets of primitive concepts, typically those subsuming a concept and those disjoint to them.
- *Extension records* hold subsets of individuals, typically those that may occur in a role filling set and those that must occur in it.
- *Co-reference records* hold equivalence subsets of role chains expressing co-reference constraints.
- *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.
- *Subsumption records* hold computed subsumption relations on a three-valued basis, that one concept subsumes another, does not, or cannot.
- *External records* hold translation tables that vector 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.

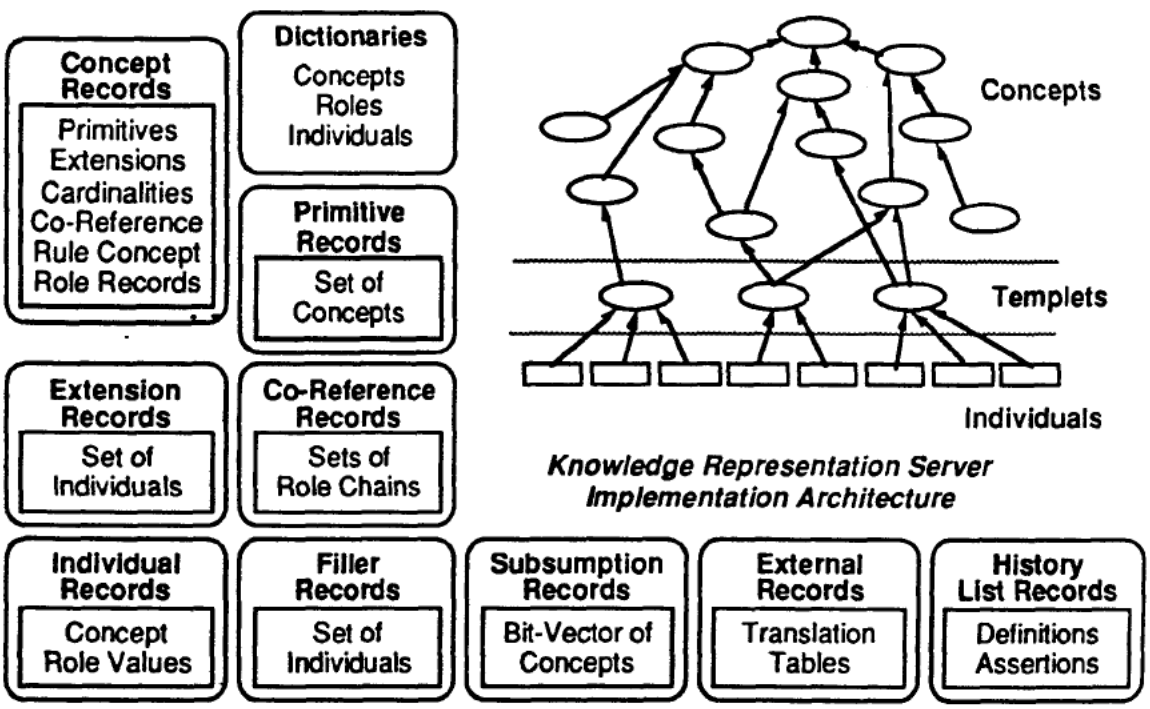

Fig.2 Data structures in the knowledge representation server

CONCLUSIONS

The C++ implementation of the server with the functionality and optimizations described, language parser and the interface to an existing grapher but not the acquisition, induction and analog modules took five person weeks. It is fast and storage efficient. We have run it with knowledge bases of several hundred concepts and individuals from our sports knowledge bases (Gaines & Vickers, 1988). 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 120KB 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 a few KB.

The KR system implemented may be regarded as itself object-oriented with concepts as classes and individuals as instances. However, there is little direct relation between the run-time KR structures and the compile-time C++ structures. The importance of the object-oriented implementation has been in achieving software engineering goals of modularity, encapsulation and reusability. Through the design of a few, reasonably generic, C++ classes we have obtained a fast, compact and reliable implementation of a powerful knowledge representation server that is easily embedded in other systems.

REFERENCES

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.

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.

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

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.

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

Gaines, B.R. (1989a). 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).

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

Leishman, D. (1989). Analogy as a constrained partial correspondence over conceptual graphs. **Proceedings of KR'89: First International Conference on Principles of Knowledge Representation and Reasoning.** pp.223-234. San Mateo, California: Morgan Kaufmann (May).

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

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