KNOWLEDGE ACQUISITION BASED ON AN OPEN-ARCHITECTURE KNOWLEDGE REPRESENTATION SERVER

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

Abstract: General-purpose knowledge acquisition systems need to be based on a fast, powerful and extensible knowledge representation technology. This paper describes an open architecture knowledge representation 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. A knowledge acquisition architecture based on the server is described.

INTRODUCTION

The motivation for the work in this paper has been to make use of developments in knowledge representation to develop a clean, fast and effective KR server as the core of a very general knowledge support system combining a wide and heterogeneous range of knowledge acquisition techniques (Gaines, 1989b). 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.
- 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.

KNOWLEDGE REPRESENTATION SERVER ARCHITECTURE

Figure 1 shows the architecture of the 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 hypermedia module allowing informal knowledge to be acquired as linked networks of text, pictures, sound, videos, and so on.
- A repertory grid module allowing graphic definition of concepts and graphic creation and editing of individuals—based on our KITTEN and KSS0 systems (Shaw & Gaines 1987; Gaines, 1988).
- A role and individual clustering module allowing interactive definition of new concepts—based on the hierarchical and spatial clustering from KSS0.
- A knowledge editing module allowing graphic editing of concept definitions.
- A conceptual induction module creating rules about specified subsets of individuals and transforming them to a minimal set of concepts and default rules (Gaines, 1989a).
- A problem solving module allowing the concepts and rules developed to be validated by testing with additional cases.
- 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.
- An interface to other knowledge bases and servers.

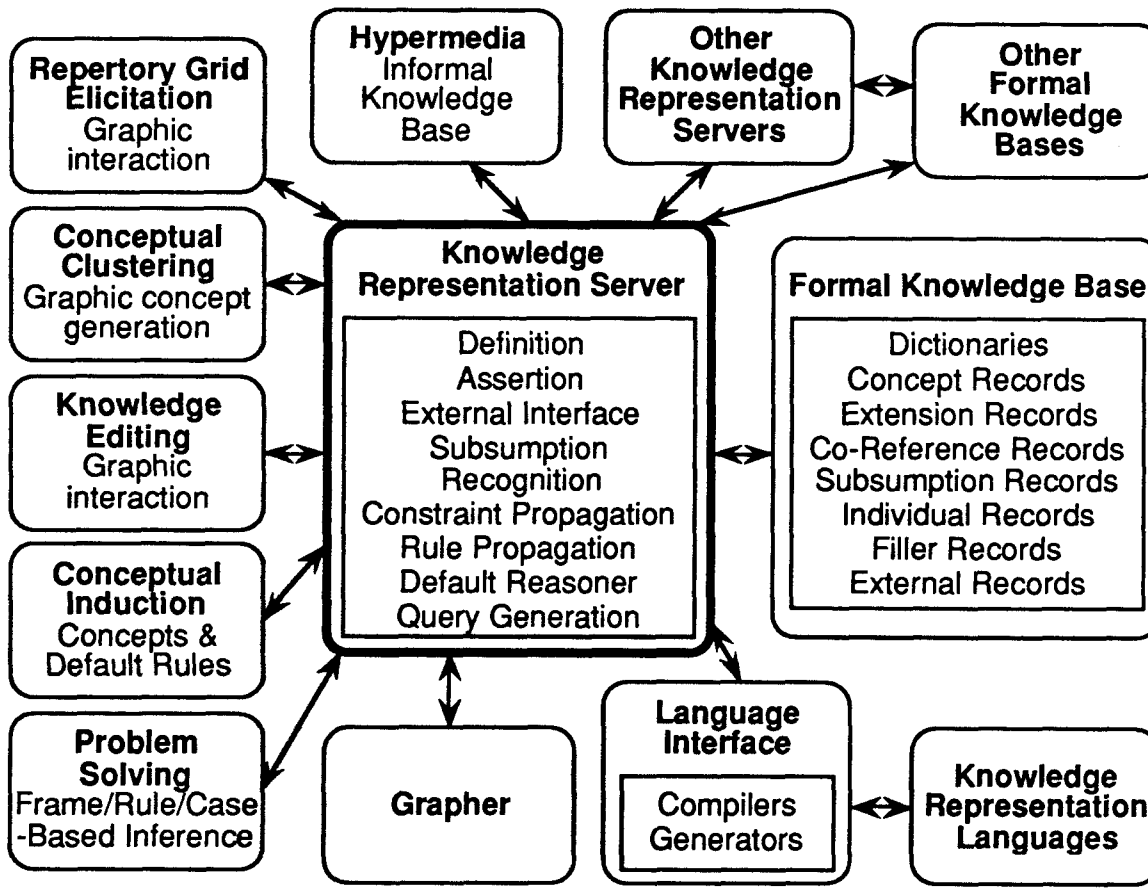

Fig.1 Knowledge support based on a knowledge representation server

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 extensions as the subsets of individuals that may be, and must 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.
- *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.
- *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.

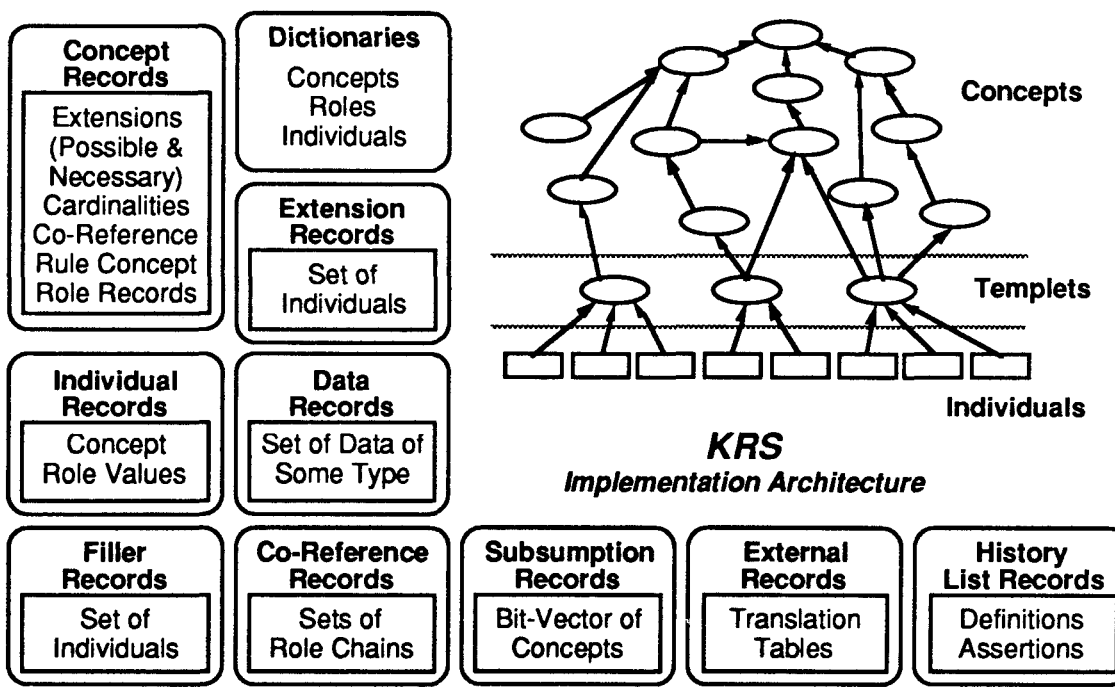

Fig.2 Data structures in the knowledge representation server

THE SERVER IN USE IN KNOWLEDGE ACQUISITION

The server supports the data structures in our previous tools such as KSS0, and the graphic user interface for concept and individual elicitation have been carried over from these tools. An 'entity-attribute grid' corresponds to a single concept whose roles are the grid attributes, and whose recognized individuals are the grid entities. Thus, there is a unique correspondence between concepts and grids, and previously successful knowledge acquisition methodologies based on grids are subsumed in the new system (Gaines & Shaw, 1989). What is added is the capability to represent and manage a hierarchy of grids in a principled fashion. The need for such hierarchies has been demonstrated in Boose and Bradshaw's (1987) development and application of AQUINAS. The application of semantic network tools for developing hierarchical knowledge representation schema to knowledge acquisition has also been demonstrated in a number of developments and applications (Boose & Gaines, 1989). The system of Figure 1 combines the grid and network approaches in a theoretically well-founded and technically well-supported architecture.

Some basic advantages of combining grid and network systems can be seen through a simple example of knowledge acquisition previously presented (Gaines, 1989c). The dataset for Cendrowksa's (1987) contact lens prescription example is readily acquired through the graphic elicitation tools in KSS0,. The underlying rules may then be derived by the Induct tool (Gaines, 1989a), and transferred to an expert system shell. However, in previous tools the rules themselves could not be shown to the user in a natural visual form where their operation is readily understood and checked. In the new tool the Induction module structures the rules as a hierarchy of concepts showing the default reasoning. Figure 3 shows the contact lens rules represented as four sub-concepts triggering five default rules. The conceptual structure of the problem is now clear, that normal tear production together with astigmatism generally leads to a hard lens recommendation, and that normal tear production without astigmatism generally leads to a soft lens recommendation.

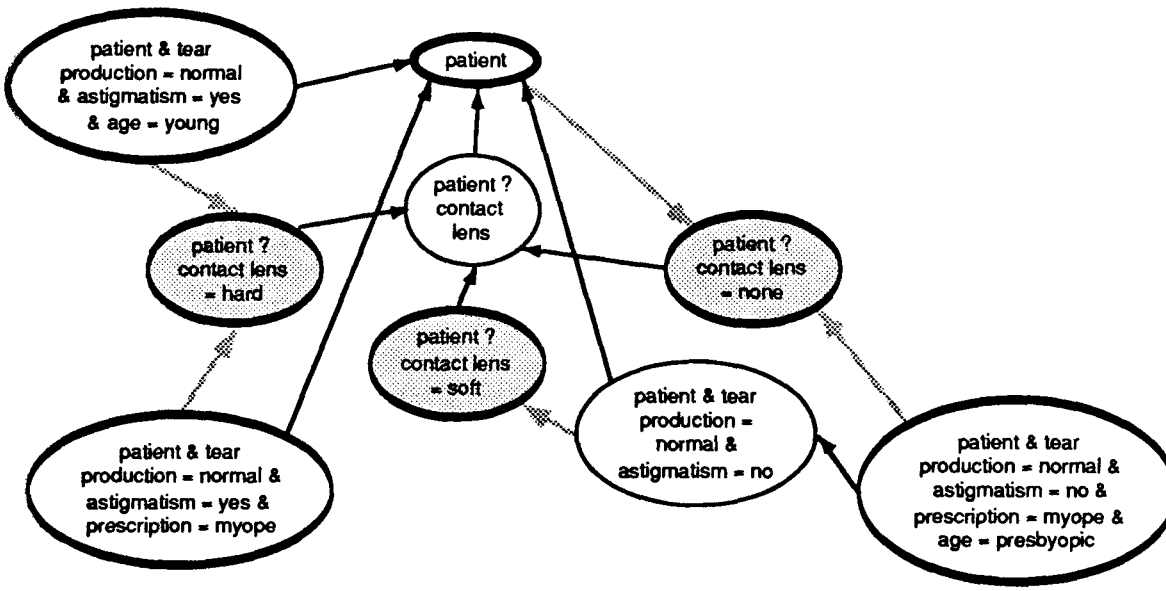

Figure 3 Rules presented as default reasoning in a concept hierarchy

The knowledge base underlying Figure 3 may be edited at the conceptual level or at the underlying dataset level from which the concepts are inductively derived. New cases are automatically classified in terms of existing concepts, and new concepts are automatically placed in the subsumption hierarchy and used to classify existing cases. The grapher output is active, and double-clicking and popup menus may be used to access underlying data structures or annotation in an associated hypermedia system.

KNOWLEDGE ACQUISITION METHODOLOGIES

The server-based knowledge acquisition architecture of Figure 1 is designed to support a range of knowledge acquisition methodologies. In particular, the open architecture allows additional tools and domain-specific acquisition methodologies to be readily added. Figure 4 shows how a number of existing methodologies may be integrated in the basic tool.

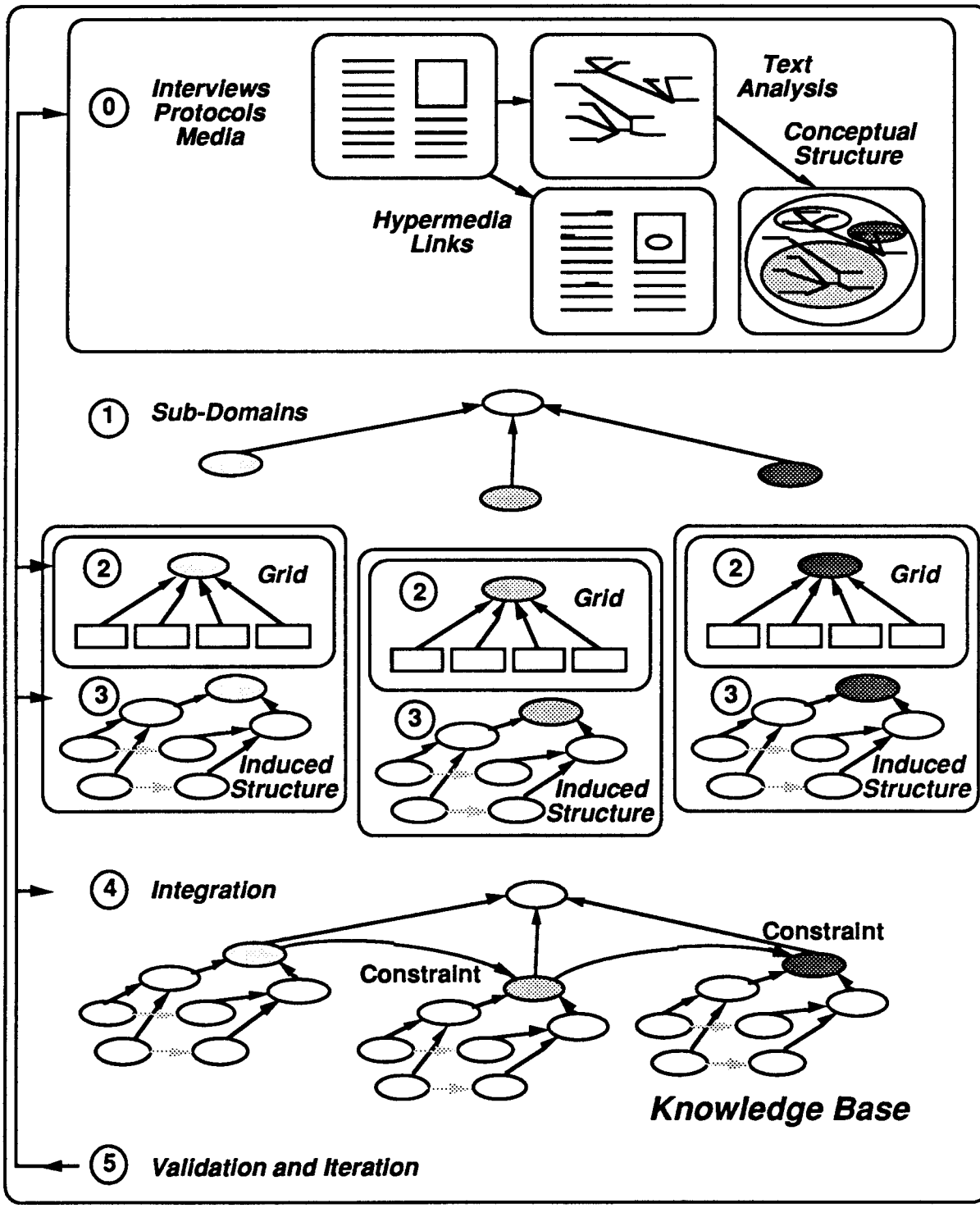

Fig.4 Integration of knowledge acquisition methodologies

Stage ⓪ consists of the acquisition of informal knowledge from interviews, protocols and media.

Stage ① consists of using the 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 and iterating back through any of the earlier stages in order to develop and refine it further.

The architecture of Figure 4 is intended to be illustrative of the way various knowledge acquisition techniques 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 the techniques and tools that we have already 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.

CONCLUSIONS

The C++ implementation of the server 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), and with knowledge bases of thousands of complexly-related concepts generated from Nebel's (1990) 'worst-case' examples. 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 combination of net, induction and grid based knowledge acquisition tools opens up new opportunities for knowledge acquisition applications. The extensible support of a wide variety of data types in the server reduces the limitations of current tools in a wide variety of practical applications. The server is itself a well-principled inference engine for classification and default reasoning problems. Its open architecture with well-defined script languages and interfaces makes it simple to interface to existing numerical tools and expert system shells.

ACKNOWLEDGEMENTS

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada. I am grateful to many colleagues for discussions over the years that have influenced the research described in this paper, in particular John Boose, Jeff Bradshaw, Bill Clancey, Debbie Leishman, Marc Linster, Alain Rappaport, Doug Skuce, Brian Woodward and other colleagues at the Knowledge Acquisition Workshops world-wide. I am particularly grateful to Mildred Shaw for research collaboration that has shaped many of the ideas reported in this paper.

KNOWLEDGE ACQUISITION METHODOLOGIES

The server-based knowledge acquisition architecture of Figure 1 is designed to support a range of knowledge acquisition methodologies. In particular, the open architecture allows additional tools and domain-specific acquisition methodologies to be readily added. Figure 4 shows how a number of existing methodologies may be integrated in the basic tool.

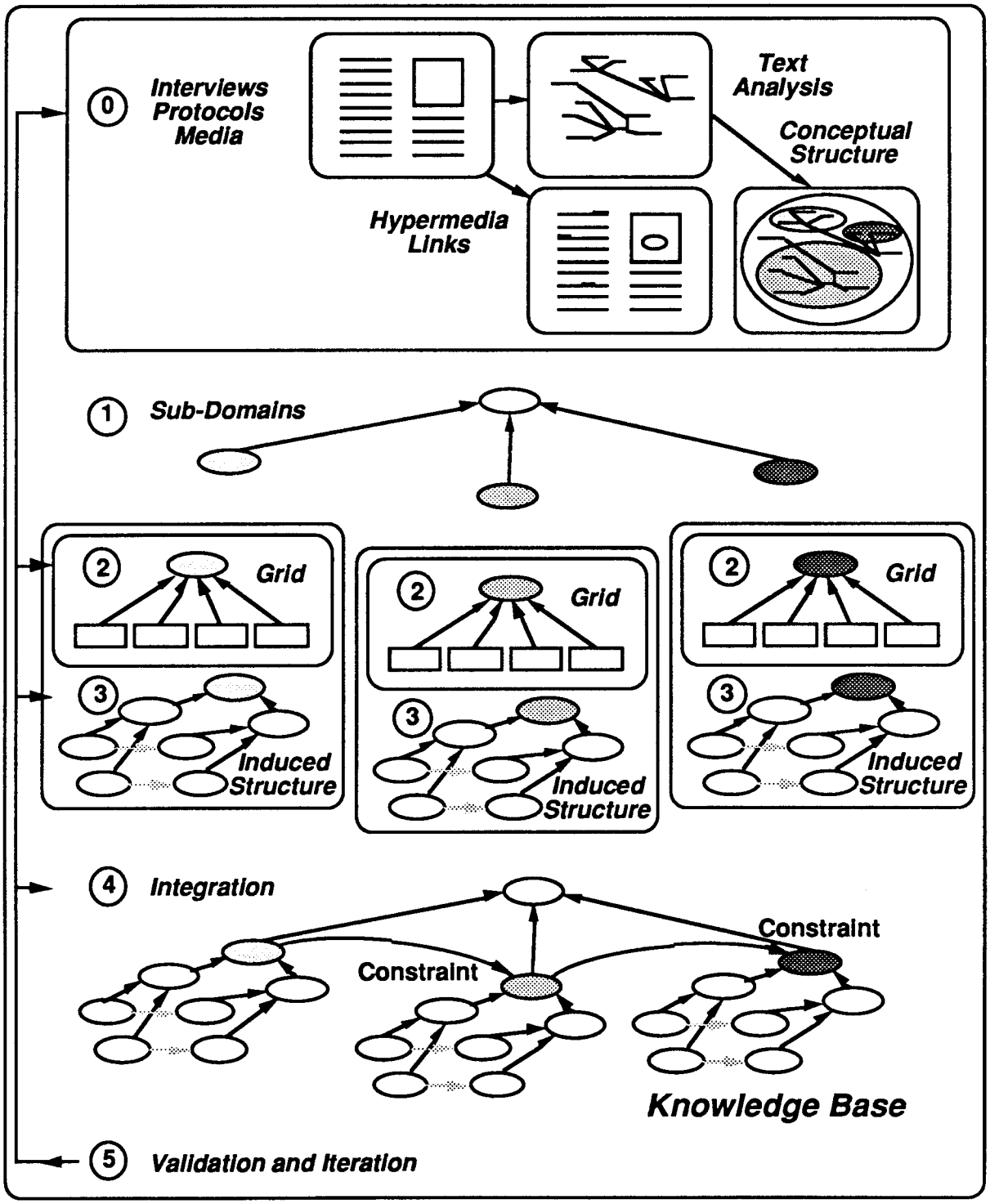

Fig.4 Integration of knowledge acquisition methodologies