

Knowledge Acquisition via Tracked Repertory Grids

Randy Wolf and Harry S. Delugach
Computer Science Department
The University of Alabama in Huntsville
Huntsville, AL 35899 U.S.A.
rwolf@cs.uah.edu and delugach@cs.uah.edu

July 31, 1996

Abstract

One of the more valuable and flexible forms of knowledge acquisition is based upon the use of repertory grids. A useful extension of repertory grids can be created by providing a method of semantically linking associated constructs and repertory grids. This network of grids is a semantic network with nodes consisting of individual repertory grids and links acting as 'tracks.' A track is a generalization of the laddering process used by repertory grid systems. These linked repertory grids which are acquired using the natural language interface of repertory grids can form an operational definition of a problem solving method.

1 INTRODUCTION

The ability to acquire complex knowledge structures is a capability that an increasing number of computational systems must have. As time passes, the knowledge acquisition requirements of computational systems will become progressively more difficult to satisfy. It is important that the ability to acquire knowledge keep pace with the demand. Knowledge acquisition (KA) techniques have consistently adopted a useful but constraining philosophical approach: In general, KA is best provided by KA being applied to specific problems and problem-solving methods. Although this is demonstrably a reasonable guide to use during the initial work on KA capabilities, future systems will require a more generally applicable KA capability.

There are two KA techniques that are suitable candidates for use as a basis for a more generally applicable KA capability. These two KA techniques are the repertory grid technique [Kelly 1955] and the meta-technique of PROTEGE-II [Eriksson, Shahar, Tu, Puerta, and Musen 1994]. The repertory grid technique is suitable because of its bias towards natural language, its domain independence, and its built-in methodology for filling in a single repertory grid. Domain experts are most comfortable and are more accurate when they are able to use the terminology and domain concepts with which they are familiar. By interacting with the domain experts using natural language, some of the difficulty of the knowledge acquisition process is eased. Repertory grids

| | | | | | |
|---|---|---|---|---|---------------------------------------|
| 5 | 1 | 1 | 5 | 5 | Symbolic/Numeric |
| 5 | 5 | 5 | 2 | 1 | Widely available/Not widely available |
| 1 | 1 | 1 | 5 | 1 | Scientific/Business |

ADA LISP PROLOG COBOL FORTRAN

Figure 1: A Repertory Grid.

acquire knowledge about the domain in the terminology used by the knowledge source. Because a repertory grid can analyze any subject, knowledge can be acquired about a wide range of topics. Because there is a methodology for completing a single grid, the knowledge acquisition as practiced by repertory grids is guided and not ad hoc during the process of acquiring the information needed for a particular grid.

PROTEGE-II is suitable because it uses meta-knowledge (knowledge about knowledge) to provide a technique (a meta-technique) for using varying techniques (non-meta-techniques) to solve differing problems¹. By unifying a variety of role-specific knowledge acquisition techniques under a common umbrella, PROTEGE-II plans to retain the power of each role-specific technique. PROTEGE-II unifies the techniques by allowing redefinition of each technique in terms of a broadly applicable methodology (the board-game method). The ability of the PROTEGE-II system to work with varying techniques and the broad applicability of the board-game method are the sources of the system's meta capability. The retention of the power of role-specific systems while simultaneously achieving the generality of a meta system is indicative of the advantage of approaches similar to PROTEGE-II.

A more generally applicable KA capability can be created by extending repertory grids. This extension is achieved by generalizing the laddering process used in AQUINAS [Boose and Bradshaw 1987]. Laddering applies a question to an existing construct to elicit a new construct. Laddering is used to acquire either a more generalized construct or a more specialized construct, depending on the question. The laddering process can be generalized by realizing that there are more than two useful types of questions which can be asked about a construct. Generalization is presently supported by asking the user why a construct is true. Specialization is supported by asking the user how a construct is characterized. Depending on the construct, other questions might provide useful information. The answers to these questions would be constructs but would not necessarily be generalizations or specializations of the base construct. For example, suppose a grid like the one in Figure 1 was describing high level languages.

A new type of question might inquire about who uses certain types of languages. The user group for a certain language can be used to identify the language, thus supporting the heuristic classification process. The answers to these new types of questions can also be used to support other processes such as making repertory grids operational. The ability to create an operational repertory grid is useful during two endeavours. One such endeavour is the capture and solution of synthetic problems although repertory grids are usually considered useful only for analytic problems. The other endeavour is the use of operational repertory grids at a meta level to control the acquisition,

¹In general, whenever the statement "A is about A" is true, with A being a concept such as "knowledge", "technique", "rule", or "language", then A is 'meta' in nature and is labelled 'meta-A' when referenced.

definition, and use of problem-solving techniques.

The remainder of this paper will provide a detailed description of the background, theory, practice, and implications of the proposed KA capability. Necessary background material will be presented which provides a basis for a subsequent theoretical description of the proposed modifications. Once the theory for the modifications is established, an example of the application of such a system is presented which is followed by a discussion of the various ramifications of the system. A conclusion will summarize the preceding material and discuss future work.

2 BACKGROUND

2.1 Historical and General Knowledge Acquisition

Knowledge acquisition of the type which interests computer science and which particularly interests artificial intelligence has been studied for quite some time. The intertwining nature of knowledge acquisition, learning, and inferencing means that study of any one of the three involves study of the other two. Since the earliest times, the nature of thought and learning has been studied. More recently, psychologists and cognitive scientists have been interested in these issues. Many computerized systems have a basis in psychological theories about the nature of cognition. Descartes and Hobbes were two individuals which had a major influence on later research in cognitive science [Stillings et al 1989]. In the modern day, Kelly [Kelly 1955] was instrumental in defining the theory of personal constructs which has been used often in recent KA systems.

Knowledge acquisition as it became known within computer science was practiced earlier on a manual basis and later on an automated basis. Manual knowledge acquisition came in several flavors with protocol analysis, interviews, and observation being common acquisition activities [Belkin, Brooks, and Daniels 1987]. Protocol analysis involves inspecting verbal records of experts describing their own thought processes as they solve a typical problem. Interviews are a process where a knowledge engineer either informally or formally debriefs an expert. Observation is an acquisitional process where an observer determines useful information by observing an expert complete a typical task. All of these processes are labor intensive and error-prone. Automated knowledge acquisition offers a significant amount of relief to these problems.

2.2 Previous Automated Approaches

The previous automated approaches to knowledge acquisition were of two types [Gruber 1988]. One type was based around machine learning where the knowledge acquisition consisted of a set of examples being provided to a program which is capable of generalizing these examples to extract pertinent knowledge about the phenomenon in question. Such types of systems tend to work better in relatively well-understood areas like planning than in difficult areas like natural language understanding [Rich and Knight 1991]. Neural network learning is one type of such knowledge acquisition. The knowledge acquired by a neural network is not symbolic in nature and is not useful as a basis for further inferencing. The other type of knowledge acquisition is interactive knowledge acquisition where an interactive program aids in the process of extracting

knowledge from a knowledge source. This type of knowledge acquisition appears to have the potential to provide the powerful, general form of knowledge acquisition that future systems will require. Systems of this type are predominantly designed to work best within the confines of a specific domain.

The restricted flexibility of many of these knowledge acquisition systems manifests itself in two ways: in a constrained representation of the acquired knowledge and in a necessarily limited application area for any single knowledge acquisition system. The knowledge representation used is often tailored to be useful for one specific type of problem and one problem solving method. MOLE, SALT, KNACK, TKAW, and OPAL are all systems which intimately associate knowledge with the role that knowledge plays [Kornell 1988]. This is a patently effective way to construct a knowledge based system that will serve a specific purpose. Considered as a whole, the knowledge based system performs a certain role just as each atomic piece of knowledge is assumed to serve a particular (albeit limited) role. The necessarily limited application area of a knowledge acquisition system typically stems from the constrained nature of the domains (e.g. cancer therapy, electrical-mechanical design, elevator design), possible problems (e.g. diagnosis, repair, planning), and possible solution methods (e.g. heuristic classification, propose-and-refine, propose-and-apply).

Repertory grid systems are the least constrained in terms of the domain that can be represented within a system. On the other hand, they are strongly restricted in terms of the problem type and the solution method. Problem types may be classified as analytic, synthetic, and an analytic-synthetic mixture. Analytic problems where it is possible to pre-enumerate the solutions which are possible are the only problem type for which repertory grids have traditionally been considered to be useful. Repertory grids are also strongly tied to the heuristic classification problem solving method which is most useful in the solution of analytic problems. It is possible to cast a synthetic problem in terms of heuristic classification [Gaines 1994, Bradshaw, Boose, Covington, and Russo 1987] but the problem must be restated to fit the heuristic classification terminology.

Meta-level systems are useful for generalizing the capabilities of a given non meta-level system. PROTEGE-II [Eriksson, Shahar, Tu, Puerta, and Musen 1994], TERESIAS [Barr and Feigenbaum 1986], ASK [Gruber 1989], and SOAR [Rich and Knight 1991] are all meta-level systems (PROTEGE-II, TERESIAS, and ASK are KA systems although SOAR is not) which improve their performance, their ability to solve general problems more efficiently, through the advent of higher-level knowledge about their own internal problem solving behaviors. Despite the improved ability displayed by these systems, they still suffer from inherent limitations in regard to their constrained knowledge representation and limited application area. TERESIAS, ASK, and SOAR all assume a rule system contains their internal knowledge. PROTEGE-II assumes a small, fixed number of defined problem solving methods.

2.3 Personal Constructs and Repertory Grids

The personal construct theory of Kelly as instantiated by repertory grids is useful for solving analytic problems by means of applying heuristic classification. Simplistically, heuristic classification is the process of determining the answer to the question "What is this?" Heuristic classification by repertory grids is based on a classification provided by a human expert or "question-answerer." The final result of expertise transfer from the repertory grid process is an automated "question-

answerer.” In a typical scenario of this problem-solving method, the question-answerer is given some (or perhaps all) information about the characteristics of the object in question. The question-answerer must then correctly classify the object. Medical diagnosis is a typical type of problem that repertory grids are used to solve.

The technique repertory grids use to solve analytic problems using heuristic classification is to acquire the constructs of the human knowledge source. Every human is assumed to unwittingly (or wittingly) act as a scientist trying to understand the world. Every human is assumed to be continually attempting to predict objects and events in their life. A “personal construct” is a characteristic of an object in the world that allows a human to make predictive judgements. A repertory grid system will query the human knowledge source using natural language text queries about the objects that the knowledge source perceives and about the constructs that the knowledge source uses to characterize the objects.

Repertory grids are more flexible than other knowledge acquisition methods because of the natural language form of constructs. A single repertory grid is used to define a class of objects. The labels for the columns of the grid are specific examples of that class. These examples are usually referred to as ‘elements’. For instance, if the user was describing a familiar class of songs then the labels of the columns would be the names of known, typical songs. The labels for the rows are the constructs. A repertory grid for an individual’s known songs might have constructs which looked like: “jazz/rock”, “slow/fast”, or “pretty/loud.” Figure 1 is a grid which describes programming languages. In this grid, numeric values in the grid are used to indicate the degree to which poles are associated with elements. In the column for ADA, the ‘5’ in the first row indicates that ADA is numeric.

Note that a construct has two sides or poles, is in a natural language form, and is not necessarily boolean. “Pretty” is not the same as “not loud.” “Pretty” and “loud” are the two sides of a characteristic that the knowledge source uses to distinguish between songs.

A very common feature of repertory grid systems is triadic elicitation. Triadic elicitation is a process by which trios of elements are used to help with the acquisition process. Typically, to start building a grid, the user supplies three candidate elements. The user is prompted to pick two elements which are the same and to name a pole which tells why these two elements are the same. If the three elements from Figure 1 were ADA, LISP, and PROLOG and the two similar elements were LISP and PROLOG, then they would be the same because they are both “symbolic.” The other pole applies to ADA and the user supplies this pole when queried as to why ADA is different from LISP and PROLOG. Different triads are used to elicit the rest of the constructs. This method helps fill in a solitary grid but does nothing to help with the process of filling in multiple grids.

The more powerful forms of repertory grid systems such as AQUINAS [Boose and Bradshaw 1987] and KSSn [Gaines 1993] provide sophisticated ways to manipulate the acquired knowledge. AQUINAS provides a number of ways to fill in individual grid values and to fill in grids. An appropriate type of grid value can be selected: nominal, ordinal, interval, or ratio. A grid value can also be selected from a probability distribution. The Analytic Hierarchy Process (AHP) can be used to aid the process of assigning values [Saaty 1980]. Laddering can be used to determine higher and lower level traits. An associated grid at one level can be used to fill in a grid at another level. Clustering can be used to derive higher level grids. Implication analysis can be used to search for redundancy. AQUINAS supports the reconciliation of knowledge from multiple

experts. KSSn has a different approach to personal constructs. KSSn acquires a grid in the form of a semantic network specialized to conform with Kelly's precepts about constructs. The visual language which describes this semantic network and the associated editor aids in the acquisition of the target knowledge into a frame system. Implication analysis can now be performed using the frame system's natural propensity for term subsumption.

Repertory grids are flexible in that the natural language form of the construct does not constrain the information which can be contained in any particular construct. This is in contrast to the constrained form into which information must be coerced in other knowledge acquisition systems. The grid is in a particular form, but the topic of the grid is not constrained. The grid could be about music, chairs, high level programming languages, or dance steps.

The inflexibility that repertory grids do display involves the problem solving method which they are intended to implement. Typically, a completed repertory grid is used to generate a rule set which performs the heuristic classification function. Thus, the information which had been captured flexibly is subsequently applied inflexibly.

2.4 Meta Level Systems

Meta-level (higher level) systems are often improved versions of previously existing non-meta systems that deal with the subject matter of the non-meta systems in a more abstract fashion. The addition of the meta reasoning level enhances the utility of the previous system. The addition of meta reasoning might increase the generality of the previous system. PROTEGE is a system which generalizes the function of the OPAL KA system. OPAL was useful for acquiring cancer therapy plans. PROTEGE allowed the ability to generate OPAL-like systems useful for performing knowledge acquisition for episodic skeletal-plan refinement (a problem-solving method useful for cancer therapy). PROTEGE-II which is a later version of PROTEGE has an even greater level of generality [Eriksson, Shahar, Tu, Puerta, and Musen 1994]. Problem solving methods other than episodic skeletal-plan refinement are supported.

Generality is not the only advantage of a meta-level system. Meta reasoning can also improve the efficiency of reasoning and knowledge acquisition. SOAR is a rule-based system which emulates human problem solving [Rich and Knight 1991]. By reasoning about which rules are likely to be useful, the efficiency of the system is improved. This type of efficiency improvement could make an unsuccessful system become successful. ASK is a meta level system which concentrates on strategic knowledge [Gruber 1989]. Strategic knowledge is knowledge about "what should be done next." This meta level operational data is used to allow the direct acquisition of the control structure for the reasoning system. TEREISIAS works with meta-level descriptions of rules [Barr and Feigenbaum 1986]. This allows TEREISIAS to help the knowledge source test the current rule set. In essence, the knowledge source is helping to debug the current knowledge structures. This follows good software engineering practice in that the testing of a system is part of the design of the system.

3 PROPOSED MODIFICATIONS

Laddering is an accepted method of acquiring new constructs from a user by asking questions about an existing construct. The proposed modifications use (re-apply) laddering-style questions to acquire new constructs. One reason that the (re)application of questions is apt is because such questions already implicitly exist at many places within the present process for acquiring repertory grids. Implicitly, questions exist internally in grids, externally between grids, during triadic elicitation, and at other miscellaneous locations. The following sections will describe the circumstances in which questions currently appear and what modifications are necessary to use those questions more systematically.

3.1 Internal Grid Questions

One place that such a question exists is internally within every grid. Every repertory grid contains information about entities and the attributes which define those entities. Implicitly, every grid answers the question “what constructs do the elements have?” which is asked about each element. It is much easier to answer an explicitly stated question than a hidden question. Indeed, the user is asked a similar question to create the constructs and to link them to elements. Despite the fact that the constructs are elicited via questions, the constructs are not primarily thought of as answers to questions but rather as simply being the constructs which describe the elements. It would be consonant with personal construct theory for constructs to be the result of answering questions. A personal scientist is in the business of answering questions and the constructs of a grid are answers to questions.

An alternate but very similar perspective would consider each repertory grid as a unit to be an answer to the question “what constructs?” Each grid as presently defined has an implicit, default meaning. The meaning of every repertory grid is “what constructs?” because every grid indicates what constructs define the elements. If each repertory grid in a repertory grid system were a relation in a relational database system then the name of each relation would be “what attributes.” Just as relational database systems have found it useful to have each relation have a specific meaning, it would also be useful for the implicit meaning of each repertory grid to not only be explicit but possibly be different.

Any question has a form. In all likelihood, any given question inquires about who, what, where, when, why, or how. Every present repertory grid is always about *what*. Such a flat hierarchy of meaning for grids is less useful than having each grid have a more appropriate designation. The availability of additional specialized types of repertory grids would be beneficial. The proposed alternate types of grids in addition to the most general type of *what* are *who*, *where*, and *when*. The present form of laddering already partially covers *why* and *how*.

Before, when the constructs which were to be used to define a class were elicited, the constructs were not constrained. The constructs could be about any mixture of *who*, *where*, and *when*. It was considered sufficient for the constructs to somehow apply to the element. Now, when a class is being defined, constructs which specifically deal with people, locations, and time are extracted separately. There could still be an unconstrained *what* grid to be elicited but it should contain constructs which do not strictly pertain to just *who* or *where* or *when* information. Similarly

| | | | |
|---|---|---|--|
| e | n | d | wants your money/ does not want your money |
| n | e | e | goes to houses/ doesn't go to houses |
| n | e | e | available 24 hrs/ 9-5 |
| n | n | e | shoot criminals/ doesn't shoot |
| n | e | n | fight fires/ no fire fighting |
| n | e | e | go to emergencies/ not emergency personnel |

tax collector fire fighters police

Figure 2: Repertory Grid with Intermixed Who, When, Where Information.

to AQUINAS, it is logical to assume that hierarchies of constructs will exist. Overlaid on the most general *what* hierarchy will be the *who, where*, and *when* hierarchies. These additional three categories are not purported to be the only possible three such categories, they are simply three categories which provide generally applicable and useful knowledge.

For example, suppose the class in question was “public servants.” The normal *what* grid might look something like Figure 2. The ‘e’ in the first row in the tax collector column indicates that the emergent (e) pole applies to the tax collector element, that the tax collector wants your money. The ‘n’ in the first row in the fire fighters column indicates that the non-emergent (n) pole applies to the fire fighters element, that fire fighters do not want your money. The ‘d’ in the first row in the police column indicates that the construct does not (d) apply to the element. This grid contains a conglomeration of time, location, personnel, and general constructs. If the knowledge about the constructs which define an element were segregated by topic, the user might find it easier to remember constructs of a particular type. This in turn might lead to a better characterization of a particular class or element. For example, assume that the constructs which were describing a set of elements were restricted to being temporal in nature. If the present set of constructs were Monday, Tuesday, Thursday, Friday, Saturday, and Sunday then the user might immediately see that Wednesday should also be a construct. If the temporal aspect of constructs were intermixed haphazardly with other constructs emphasizing other aspects then it might not be so clear that Wednesday must also be included.

These additional hierarchies might allow a more efficient solution of the heuristic classification process. The number of questions that a user must answer and the speed of the system are an issue. The execution time of the system is proportional to the number of levels in the hierarchies. The overlaid new hierarchies potentially could create a shorter path through the grid structure.

3.2 Questions that Link Grids

Laddering is normally used to link together grids by acquiring generalized (more abstract) and specialized (less abstract) versions of constructs that already exist. The needed semantic modification is to allow repertory grids to be linked together in a manner similar to the way semantic networks allows concepts to be linked together. In addition to laddering questions which acquire *why* and *how* information; *who*, *what*, *where*, and *when* information will be acquired. The availability of additional questions would be just as useful in this context as it was internally to grids. By treating additional questions as “tracks” (instead of as ladders which only go up and down) which lead to additional information, the richness of the knowledge base is increased. Instead of just two possible

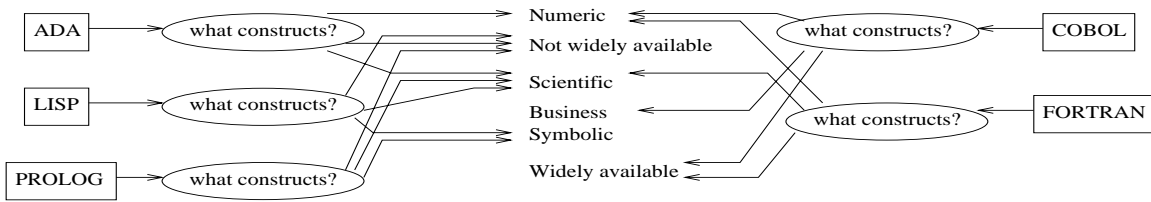


Figure 3: Track Representation of the AQUINAS Repertory Grid.

| | | | | | |
|-----|------|--------|-------|---------|--|
| e | n | n | n | n | Used primarily by government programmers/ Not used primarily by government programmers |
| n | e | e | n | n | Used primarily by artificial intelligence researchers/ Not used primarily by artificial intelligence researchers |
| n | n | n | e | n | Used primarily by commercial programmers/ Not used primarily by commercial programmers |
| n | n | n | n | e | Used primarily by scientists and engineers/ Not used primarily by commercial programmers |
| ADA | LISP | PROLOG | COBOL | FORTRAN | |

Figure 4: Track Grid for the Question "who uses?"

tracks, i.e. "why preferred?" and "how characterized?", there are now as many tracks as might be useful. It is not suggested that tracks proliferate uncontrollably, but rather that tracks be available when useful. In retrospect, it can be seen that the additional questions suggested in the preceding section are also tracks. The difference is that those tracks apply internally to grids while these would link grids. Notationally, the grid from AQUINAS in Figure 1 might be described in terms of tracks by Figure 3.

Figure 3 takes the liberty of treating a rating of 1 and 2 as being identical. Each element in the original grid is the object of inquiry when the user is asked "what constructs?" Since the same question is being asked of each element albeit with varying answers, the meaning of the grid itself can be considered to be the answer to the question "what constructs?" This is the essence of the first suggested semantic modification with the proviso that questions other than "what constructs?" should also be allowable. For instance, simply "when?" or "when existed?" or "where born?" might be the questions. Each of these questions is a track as a track applies to the meaning of a single grid.

There are three ways that a question might link a source grid to a destination grid. One way is that the track question might be about the constructs of the source grid. The second way is that the track question might be about the elements of the grid. The final way is that the track question might be about element/construct pairs.

A track question which applies to the elements of Figure 1 is "who uses?" This question is particularly apt because one of the most important characteristics of high level languages is who uses the language. Figure 4 shows a possible resulting repertory grid.

A conceptual graph representation of the relationship of the Figure 1 grid to the Figure 4 grid appears in Figure 5. A conceptual graph is a form of a semantic network which has a semantics based on predicate logic, an ability to organize information in a standardized and therefore comparable format, and relatively few built-in, unavoidable restrictions on its ability to represent knowledge [Sowa 1984].

An alternate notation that emphasizes the difference between internal track questions and external

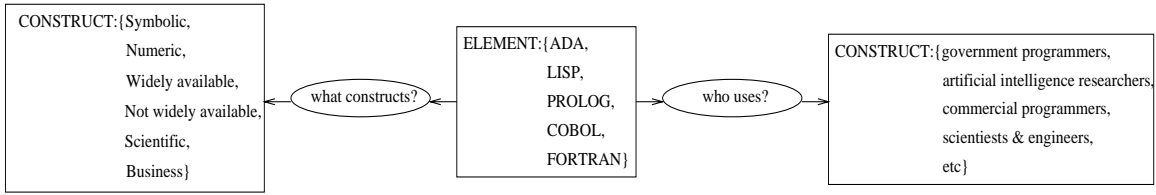


Figure 5: Conceptual Graph of the Relation between the Grids of Figure 1 and Figure 4.

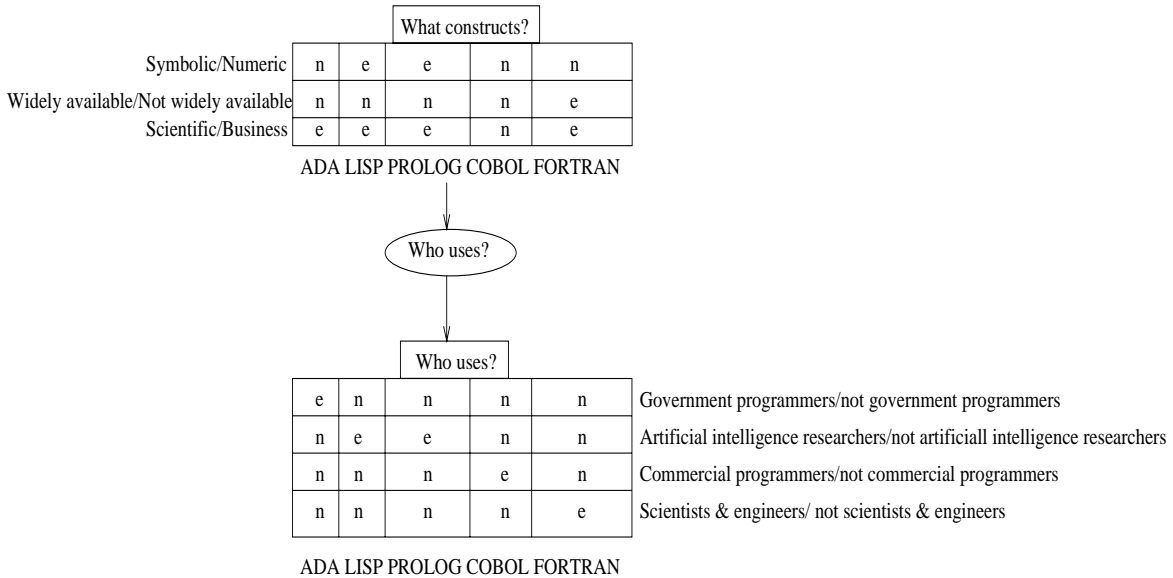


Figure 6: Alternate Representation of the Relation between the Grids of Figure 1 and Figure 4.

track questions appears in Figure 6.

Another example where the track question applies to elements with specific constructs is shown in Figure 7. The track question in this case is “why true” and this question is not being asked about just the constructs or simply the elements but rather is being asked about elements associated with specific constructs. This example grid is not fully populated because the full grid would be too large and a grid fragment is sufficient to illustrate the technique.

3.3 Other Questions and Making Grids Operational

This subsection will show how other questions that occur during the present repertory grid process can be considered track questions and how these track questions relate to the process of making repertory grids operational. Prior to that, it will be shown that tracked repertory grids can be used to capture an operational description of a problem procedure. A very simple example of this type of knowledge acquisition can take as a subject the process of averaging two numbers. The track for the initial grid should be “what actions?” instead of “what constructs?” or just “what?” Figure 8 shows the initial grid.

This grid is somewhat unusual compared to repertory grids tailored for heuristic classification. There is only one element and only one pole of each construct applies to an element. The assumed

| Why true? | | | | | | | |
|-----------|---|---|---|---|---|---|---|
| e | d | d | d | d | d | d | optimized for numeric applications/ not optimized for numeric applications |
| d | e | d | d | d | d | d | scientists and engineers need/ not needed by scientists and engineers |
| d | d | e | d | d | d | d | science involves numbers/ science does not involve numbers |
| d | d | d | e | d | d | d | business needed a business language/ business did not need a business language |
| d | d | d | d | e | d | d | demand for AI applications not widespread/ AI needed |
| d | d | d | d | d | e | e | AI applications typically operate on symbols/AI apps do not typically operat on simbols |

FORTRAN numeric COBOL business LISP symbolic
 FORTRAN widely available PROLOG not widely available
 FORTRAN scientific PROLOG symbolic

Figure 7: Repertory Grid for the Track Question "why true?" as it Applies to Figure 1.

| What actions? | | |
|---------------|--|---|
| e | | acquire one number/do not acquire one number |
| e | | acquire the other number/ do not acquire the other number |
| e | | add the two numbers/ do not add the two numbers |
| e | | divide by two/ do not divide by two |

averaging two numbers

Figure 8: Tracked Repertory Grid for the Capture of Operational Constructs.

intent of grids in this proposal are not just to provide for the execution of heuristic classification but make operationalization possible. Just as it was not convenient in AQUINAS to only allow bi-polar constructs [Boose and Bradshaw 1987], it is not convenient in this proposed system to assume that well-formed grids have multiple elements or that every pole must apply to an element. It is even tempting to allow the possibility of a single pole construct! However, it is convenient to assume a default non-emergent pole which is the boolean negation of the emergent pole. In the succeeding figures, whenever the non-emergent pole is not specified, such a default pole exists.

There seem to be at least two different ways to proceed with making this repertory grid operational. One way would be to use additional tracks as needed to extract additional useful information and to assign high level language operations directly to the constructs. Figure 9 illustrates one useful, additional track. This track is "when occurs?" This provides for the ordering of the actions necessary to average two numbers.

This grid shows that sequential operations can be acquired. Recall that sequence, alternation, and iteration are all the types of operations that are required to be supported to have power equivalent to a high level language. Alternation and iteration can also be acquired by tracked grids. Figure 10 demonstrates alternation.

Unfortunately, use of high level language constructs to provide the components used in operational tracked grids leaves dangling two issues. One issue is how the other questions present in the current repertory grid process can be integrated into this proposal for tracked grids. The other issue is how can a methodology be used that will control the acquisition of multiple grids: unlike the triadic

| | | | | |
|---|---|---|---|--|
| d | e | e | e | after acquire one number/ not after acquire one number |
| e | d | n | n | before acquire the other number/ not before acquire the other number |
| n | d | e | e | after acquire the other number/ not after acquire the other number |
| e | e | d | n | before add the two numbers/ not before add the two numbers |
| n | n | d | e | after add the two numbers/not after add the two numbers |
| e | e | e | d | before divide by two/ not before divide by two |

acquire one number acquire the other number add the two numbers divide by two

Figure 9: Tracked Repertory Grid for the Capture of Operational Sequence.

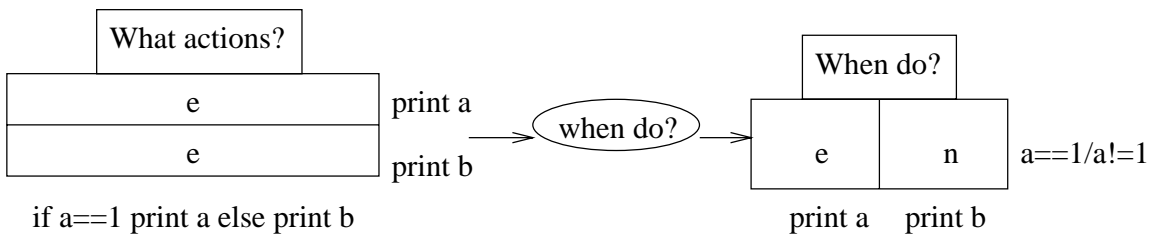


Figure 10: Tracked Repertory Grid Example of Alternation.

elicitation methodology which only controls questions during the process of building a single grid. These issues can be resolved by not using high level language statements to provide the operational definition of constructs but rather to re-use a familiar paradigm for specifying the operations: to use calls to repertory grid procedures as operation definitions.

The creation of a repertory grid is an operation that uses input from an external entity to create a data structure that is a repertory grid. The execution of a procedure which creates a repertory grid necessarily involves input and assignment. The procedure potentially involves iteration. Figure 11 shows this architecture.

Consider the construct “acquire one number” from Figure 8. This is an input operation. A repertory grid creation procedure is suitable for performing this input. In fact, the track question “what number?” specifies just such a repertory grid creation procedure. The grid that would be created would be a 1 by 1 repertory grid. Therefore, the operationalization of “acquire one number” can be achieved by prompting the user to supply the track question which produces the desired result. Figure 12 shows the complete operational definition of Figure 8.

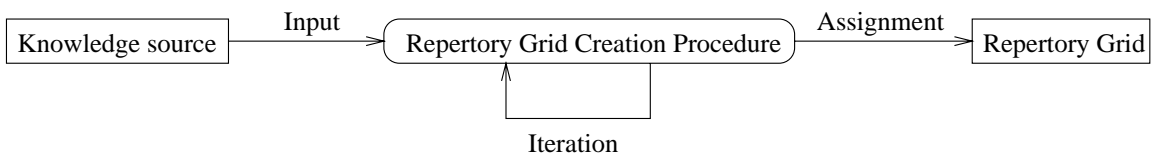


Figure 11: Repertory Grid Creation.

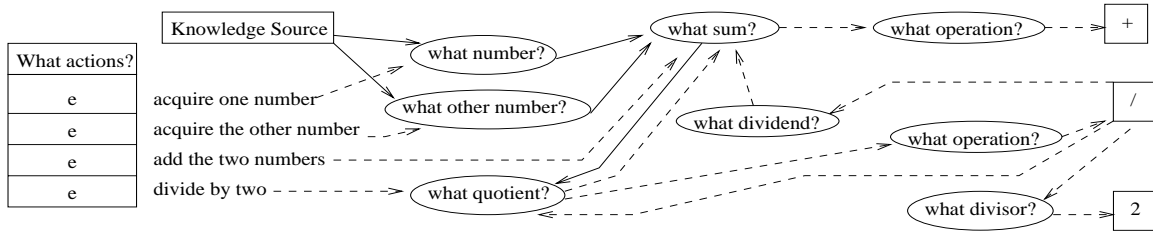


Figure 12: Operational Definition of Figure 8.

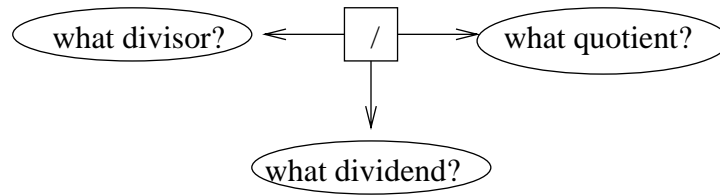


Figure 13: Meta Grid for Averaging Example.

Additional track questions were needed to actually specify the meaning of “what sum” and “what quotient” since the proposed system is *not* a natural language processor. The system does recognize a minimal set of operators which includes addition and division. The user would need to manually ask the question “what operation” but once the system sees the construct + or / it will know what actual math function to execute. In the case of /, the system will know to ask the track questions which identify the divisor and dividend. The method the system will use to know to do this is the same method that will integrate the other questions currently used by the current repertory grid process.

The control of the querying of the user will derive from a set of repertory grids that are functioning at a meta level. These grids are ‘meta’ because they are grids that contain information about grids. For the simple averaging example, there will be only one such meta grid. This grid in Figure 13 is entered by the individual filling the role of the meta knowledge engineer (KE). This grid is used in a pattern-directed fashion. That is, whenever the system sees a / the system will inquire about dividend, divisor, and quotient. A meta-grid may also simply be a grid which has been fully or partially operationalized. In the case of partial operationalization, it is the task of the non-meta KE to complete the operationalization task.

Figure 11 shows the architecture for the proposed system. The meta-KE is responsible for creating whatever meta-grids needed to control the lower level acquisition process. Whoever is filling the role of the domain expert and/or the non-meta-KE is responsible for the rest of the task.

There are three functions that meta-grids can perform. One such function occurs when a meta-grid specifies a number of track questions to be asked to acquire initial elements and constructs. In the current form of repertory grids, the user is queried as to the subject, i.e. “what subject” which is followed by a request for the user to enter a list of elements or by the start of the triadic elicitation method. To strictly mimic the behavior of the current method, a meta-grid would specify an initial track question of “what subject?”. The second function that meta-grids will provide is a specification of the process for filling in a single grid. When triadic elicitation is mimicked, the track questions “how same” and “how different” will be asked of element pairs from the present

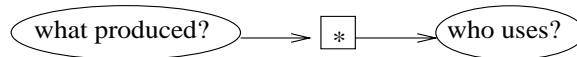


Figure 14: Meta Grid for Controlling New Grid Creation.

triad. The third function is to control the creation of new grids by the way of track questions based on previous grids. The pattern directed meta-grid used to trigger track questions about division in the previous simple example is just such a grid. Figure 14 is a more sophisticated example. In this case, whenever a construct has been created as the result of the question “what produced?” this construct is the subject of the question “who uses?”. These meta functions allow a unified methodology to be applied to the creation of networks of grids and to the process of creating a single grid.

The typical usage of this system will be by a domain expert who will act as the non-meta KE. At system start time, the domain expert will have available a library of configurations of meta-grids. The default set of meta-grids will provide the methodology needed to guide inquiry about *who*, *what*, *where*, *when*, and *why*. Another set of meta-grids available within the library will implement basic repertory grid functionality (including triadic elicitation). There will be a small set of predefined operations including math operations. Although it is not assumed that the end result of the proposed system is always an operational set of grids, there will be a set of meta-grids which will guide the process of making operational the grids provided by the domain expert. These meta-grids will ask “what grids are operational?” and “what questions lead to operational grids?”. For instance, if the base grid were about “what actions?” then the user could indicate that this grid is operational. Since it may be necessary to decompose constructs in this grid, the question “how do?” applied to this grid would create operational derived grids. By acquiring from the user which grids are operational and by making sure that no construct in these grids fail to attain operational form, the system can know that the operationalization process is complete. A construct can attain operational form three ways. The construct can have its equivalent track question defined (which is directly executable). The construct can be defined by a chain of track questions with the last question in the chain being a predefined operation. The construct can be operationally linked to another grid (e.g. via “how do?”) which in turn has attained operational form.

4 DETAILED EXAMPLE

The Sisyphus I room assignment problem is a good problem to use to show how the proposed technique would work. This older Sisyphus problem is synthetic in nature and the proposed technique can solve it *naturally* using repertory grids. It is not necessary to recast the problem description into the form of heuristic classification. The meta-level grids that describe the problem will be shown first. Subsequently, the non-meta level grids will be shown.

4.1 Meta-level

The meta-level description of a synthetic problem is shown in Figure 15 [Bradshaw, Boose, Covington, and Russo 1987]. This model is very general in nature but it does

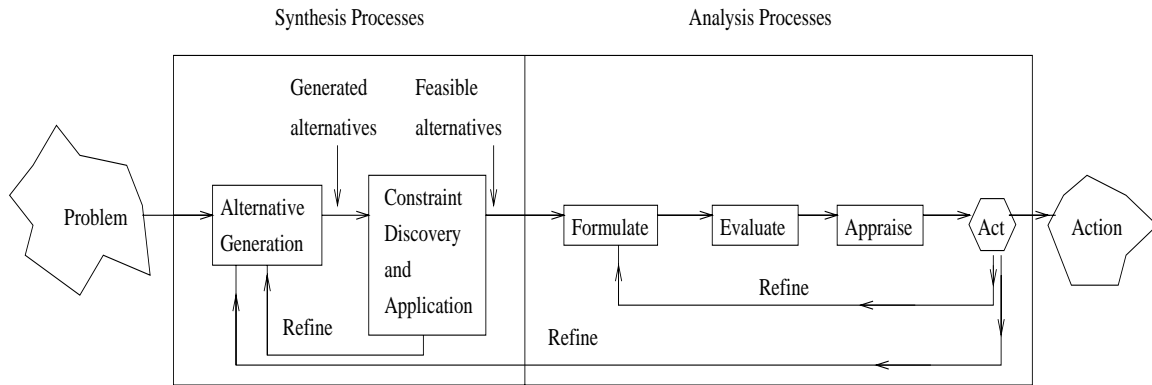


Figure 15: Synthetic Problem Description.

| What? | |
|-------|--|
| e | cannot enumerate solutions/can enumerate solutions |
| e | involves constraints/ does not involve constraints |
| e | sometimes involves backtracking/ does not involve backtracking |
| e | not analytic/analytic |

synthetic problem

Figure 16: First Grid for a Meta-level Synthetic Problem Methodology (a *what* grid).

show specific steps in the process of solving synthetic problems.

The methodology for synthetic problems would be acquired from the meta-KE. The meta-KE might not know that the lower level synthetic problem is going to be the Sisyphus I problem. Subsequently, the non-meta KE would further specialize the problem solving behavior to fit the specific problem. The assumption for this example is that the meta-level knowledge reservoir is empty at the start of the example. There is no previously existing acquisition strategy available. In a more realistic setting, the system would have a library of pre-defined methodologies and a procedure similar to that used by PROTEGE-II [Eriksson, Shahar, Tu, Puerta, and Musen 1994], DDUCKS [Bradshaw and Boose 1993] or FAD [Bradshaw, Covington, Russo, and Boose 1989] for selecting an appropriate strategy can be used.

The task of the meta-KE would be to capture the knowledge about synthetic problems contained in this figure. The final result of the overall acquisition process should be operational and it should contain whatever strategic knowledge that is useful. Figure 16 would be the first grid.

The track which leads to an operational definition is “how do?” Note that in this case that *how* is not being used for generalization or specialization but rather for operationalization. Figure 17 is the grid that the meta-KE would enter.

Comparison of this grid with the original synthetic problem description shows that not every entity in the original problem description has a corresponding construct in the grid or has a textually

| How do? | |
|---------|---|
| e | acquire problem/ do not acquire problem |
| e | generate alternative/ do not generate alternative |
| e | check constraints/ do not check constraints |
| e | formulate/ do not formulate |
| e | evaluate for optimality/ do not evaluate |
| e | appraise/ no appraisal |
| e | display result/ no output |

synthetic problem

Figure 17: Grid for the Track “how do?”

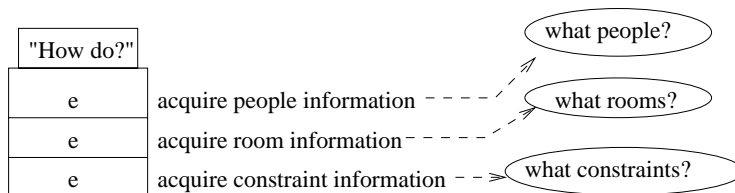


Figure 18: Operational Form of “Acquire Problem.”

identical counterpart in the grid. Some restructuring is inevitable and desirable. The expert or KE must describe knowledge in domain specific terms which are familiar.

At this point, the problem description is as operational as it is possible for it to be at the meta level but it is not fully operational. The precise meanings of constructs such as “acquire problem” and “check constraints” must be defined by the non-meta KE. For the Sisyphus problem, “acquire problem” would be the straightforward input of people, rooms, and constraints. The construct “check constraints” would consist of the operational description of checking for violated constraints.

4.2 Non Meta-level

The non-meta KE knows that the precise problem is the Sisyphus I room assignment problem. Figure 18 shows the operational grid for the track question “how do?” as asked about the construct “acquire problem.”

There are many ways that an alternative might be generated but assume that the grid for “how do?” as applied to “generate alternatives” is as in Figure 19.

Figure 20 shows an intermediate operationalization of “assign head of group.”

The construct “assume rooms are unassigned” in Figure 19 will eventually take the form of an iteration which initializes a repertory grid that contains room assignments. The construct “find large, central, unassigned room” will decompose into two iterations. One iteration will find unassigned rooms within the repertory grid that stores room assignments. The other iteration will pick out large central rooms from the repertory grid data structure which defines rooms. Subsequently, the results of the two iterations will be merged to produce the large central unassigned room. A repeated

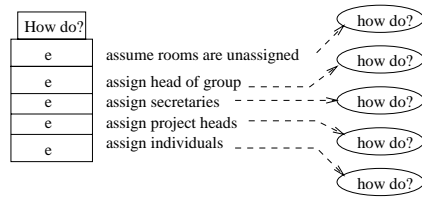


Figure 19: Operational Form of “Generate Alternatives.”

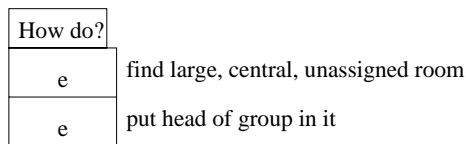


Figure 20: Operational Form of “assign head of group.”

process of decomposition and operationalization applied to the other constructs from Figure 17 will complete the process of producing a system to solve the Sisyphus I problem.

5 DISCUSSION

There are a number of issues which have not been covered in the preceding analysis or covered only lightly. These issues are:

1. Track questions form a hierarchy.
2. Tracked grids include implication grids.
3. Ontological uses of tracked grids.
4. Standard knowledge representation of tracked grids.

Potential track questions are not really limited but the track questions that exist at a particular time could be considered to form a hierarchy. One possible hierarchy is shown in Figure 21. Solid lines indicate concatenations and dotted lines indicate is-a relations. Boxes group potential responses to a concatenation. For instance the solid line from "where" to the box containing "exists", "located", and "created" indicates three different track questions. The dotted line from "what" to "who" indicates that "who" is a "what."

Implication grids are a newer form of personal construct system where implications between constructs are explicitly indicated in the grid. The fact that implication grids can be easily stated within the paradigm proposed by this paper means that the paper's scope has greater generality.

In addition to generalizing the ability of knowledge acquisition to be used to perform synthetic problems and to capture strategic knowledge, the modifications proposed in this paper should allow for a more controlled acquisition of ontological knowledge. The same characteristics that allow

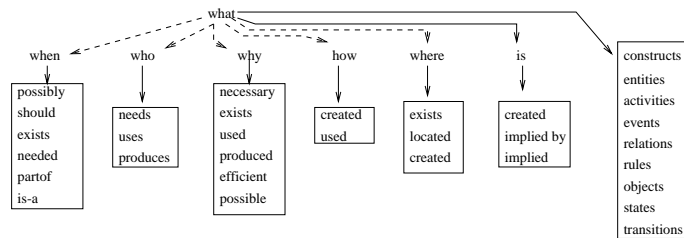


Figure 21: Track Question Hierarchy.

tracked grids to be used to establish a multiple grid methodology will allow tracked grids to be used to establish a methodology for acquiring ontological knowledge.

There is a reasonable process for converting constructs to a standard knowledge representation (conceptual graphs) [Wolf and Delugach 1996]. If this process is applied to tracked grid constructs, then the knowledge contained within a tracked grid system would be accessible to other systems.

6 CONCLUSION

The system described by this paper has a number of abilities that previous knowledge acquisition systems have achieved with difficulty or not achieved at all. The acquisition of synthetic problems is one such ability. Another ability is the ability to capture operational and strategic knowledge. A third ability is the ability to apply a methodology to the acquisition of multiple grids. The fourth is the ability to acquire problem solving methods using the natural language interface of repertory grids. The simultaneous attainment of these three abilities would be quite useful. The additional work that needs to be done is to implement and validate the proposed system.

References

- [Barr and Feigenbaum 1986] Barr, A., and Feigenbaum, E.A. (1986). *The Handbook of Artificial Intelligence*, Vol. 2, Addison-Wesley, Reading, Massachusetts.
- [Belkin, Brooks, and Daniels 1987] Belkin, N.J., Brooks, H.M., and Daniels, P.J. (1987). "Knowledge Elicitation using Discourse Analysis" *International Journal of Man-Machine Studies*, Vol. 27(2), pp. 127-144.
- [Boose and Bradshaw 1987] Boose, J.H., and 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*, Vol. 26, pp. 3-28, 1987.
- [Bradshaw, Boose, Covington, and Russo 1987] Bradshaw, J.M., Boose, J.H., Covington, S.P., and Russo, P.J. (1987). "How to Do With Grids What People Say You Can't" *Knowledge Acquisition: An International Journal of Knowledge Acquisition for Knowledge-Based Systems*, Vol. 1(4).

- [Bradshaw, Covington, Russo, and Boose 1989] Bradshaw,J.M., Covington,S.P, Russo,P.J., and Boose,J.H. (1989). “*Aquinas, Axotl, Folie à Deux: Doing Knowledge Acquisition for Decision Analysis Problems*” *Proceedings of the Fourth AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.
- [Bradshaw and Boose 1993] Bradshaw,J.M., and Boose,J.H. (1993). “Mediating Representations for Knowledge Acquisition”, *Proceedings of the AAAI '92 Knowledge Acquisition: From Science to Techniques to Tools Workshop*, Anaheim, CA, July, 1991.
- [Eriksson, Shahar, Tu, Puerta, and Musen 1994] Eriksson,H., Shahar,Y., Tu,S.W., Puerta,A.R., and Musen,M.A. (1994). “Task modeling with reusable problem-solving methods”, *Artificial Intelligence*, Vol. 79, pp. 293-326, 1995.
- [Gaines 1993] Gaines,B.R., and Shaw,M.L.G. (1993). “Knowledge Acquisition Tools Based on Person Construct Psychology”, *Knowledge Engineering Review*, Vol. 8(1), pp. 49-85, 1993.
- [Gaines 1994] Gaines,B.R. (1994). “A Situated Classification Solution of a Resource Allocation Task Represented in a Visual Language”, *International Journal of Human-Computer Studies*, Vol. 40(2), pp. 243, 1994.
- [Gruber 1988] Gruber,T.R. (1988). “Acquiring Strategic Knowledge From Experts”, *International Journal of Man-Machine Studies*, Vol. 29(5), pp. 579-597, 1988.
- [Gruber 1989] Gruber,T.R. (1989). *The Acquisition of Strategic Knowledge*, 1989, Harcourt Brace Jovanovich, Boston.
- [Kornell 1988] Kornell,J. (1988). “Formal thought and narrative thought in knowledge acquisition”, pp. 35, in *Knowledge Acquisition for Knowledge-based Systems*, B. Gaines and J. Boose, eds., 1988, Academic Press, San Diego, California.
- [Kelly 1955] Kelly,G.A. (1955). *The Psychology of Personal Constructs*, 1955, W. W. Norton, NY, NY.
- [Rich and Knight 1991] Rich,E., and Knight,K. (1991). *Artificial Intelligence*, 1991, McGraw-Hill NY, NY.
- [Saaty 1980] Saaty,T.L. (1980). *The Analytic Hierarchy Process*, McGraw-Hill, 1980, New York.
- [Sowa 1984] Sowa,J.F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*, 1984, Addison-Wesley, Reading, Massachusetts.
- [Stillings et al 1989] Stillings,N.A., Feinstein,M.H., Garfield,J.L., Rissland,E.L. (1989). David A. Rosenbaum, Steven E. Weisler and Lynne Baker-Ward, (Eds.) *Cognitive Science: An Introduction*, MIT Press, Cambridge, Massachusetts.
- [Wolf and Delugach 1996] Wolf,R.P., and Delugach,H.S. (1996). “Knowledge Acquisition via the Integration of Repertory Grids and Conceptual Graphs”, *Proceedings of the International Conference on Conceptual Structures*, University of New South Wales, Sydney, Australia, August, 19-23 (accepted).