Autonomously Acquiring Declarative and Procedural Domain Knowledge for ICAT Systems

Vincent J. Kovarik Jr.
Software Productivity Solutions
122 4th Avenue
Indialantic, FL 32903
vjk@sps.com

Abstract

The construction of Intelligent Computer Aided Training (ICAT) systems is critically dependent on the ability to define and encode knowledge. This knowledge engineering effort can be broadly divided into two categories domain knowledge and expert or task knowledge. Domain knowledge refers to the physical environment or system with which the expert interacts. Expert knowledge consists of the set of procedures and heuristics employed by the expert in performing their task. Both these areas are a significant bottleneck in the acquisition of knowledge for ICAT systems. This paper presents a research project in the area of autonomous knowledge acquisition using a passive observation concept. The system observes an expert and then generalizes the observations into production rules representing the domain expert's knowledge.

INTRODUCTION

Knowledge acquisition remains a bottleneck in the construction of expert systems. There have been a number of projects which have sought to automate the process of knowledge acquisition but have typically focused on the acquisition of knowledge as machine learning. From early systems such as AM [4] to more recent projects such as Cyc [6] investigating the construction of new knowledge has been based primarily on reasoning and discovery within the system.

Acquisition of knowledge for an ICAT system has a somewhat different perspective. Rather than covering a wide range of concepts, domain knowledge acquisition is concerned primarily with the capture and codification of the set of heuristics required for expert performance in that domain. While this does lead to system brittleness as problems are encountered which are on the fringe of the knowledge base or require commonsense reasoning based on real-world experiences or knowledge, it remains a proven and viable approach for special-purpose expert systems in the diagnosis, control, and procedural task oriented domains.

The focus of this effort has been on the capture and codification of knowledge relating to procedural and diagnostic tasks. These domains provide systems which are typically physical and, therefore, can be *instrumented* and the expert performs some *action* as part of performing their tasks. These two concepts, instrumentation and actions are key to enabling *i*Shadow to autonomously acquire knowledge.

Knowledge Acquisition Bottleneck

The knowledge acquisition process is a human intensive effort representing a serious impediment in the development of knowledge bases for expert systems and ICAT systems [5]. The knowledge engineer must work with the expert for extended periods of time. The domain of the expert must be learned by the knowledge engineer such that the knowledge engineer may encode the necessary domain information. Effective transfer of the acquired knowledge is a critical component [1] but continues to require significant human involvement.

The current methodology for acquiring knowledge requires the dedication of an individual to gather the knowledge from the expert and codify it into a set of rules or other form which can be interpreted by a machine. This involves conversations and interviews with the expert, observing the expert perform the task, eliciting

additional information or rationale for the behaviors observed, and then encoding the collected interviews, observations, rationale, and behaviors into a knowledge representation language.

Clearly this continues to be a significant bottleneck in constructing knowledge bases. It presents a significant problem for ICAT systems in NASA and the military because it is difficult to obtain uninterrupted time for the domain experts during which the knowledge engineer can extract the knowledge base. Yet, with downsizing of the military, limited government funds, and increased workload, more efficient methods must be found for extracting and encoding the knowledge applied by experts in performing their tasks.

Alternative Approaches

Other approaches to knowledge acquisition focus primarily on discovery [6] or pattern recognition in sets of data (i.e. database mining) [3]. These approaches rely on a closed world in the former and a broad base of facts in the latter. *i*Shadow follows the same general tactics as discovery-based learning in that it generalizes empirical observations to ascertain general domain rules. This approach relies on inductive reasoning and the related explanation-based reasoning.

Knowledge generation from databases typically applies abductive reasoning in attempting to develop knowledge which fits a particular set of data.

KNOWLEDGE CLASSIFICATION

Expert knowledge can be broadly classified into one of three categories:

1. Declarative
2. Process
3. Meta-Knowledge

These three categories reflect the domain in which the expert performs their task, the actual task actions, and the strategic knowledge applied to the performance of the task.

Declarative Knowledge

Declarative knowledge forms the basis of the domain in which the knowledge acquisition must be performed. This portion of the *i*Shadow knowledge base is developed using the object and relation editors defined within the *i*Shadow system. It is not necessary to build a "high-fidelity" model of the domain which is capable of simulation. Rather, a basic definition of the different classes of object in the domain and then the creation of instances of those classes within *i*Shadow that map to the external system.

The key is that *i*Shadow uses the states and relationships of the various objects in the domain to build a rule base. Representation of complex interactions and simulation is not needed to develop a set of rules.

Process Knowledge

Process knowledge is the set of observations collected by iShadow as it observes the expert performing their task. The process knowledge is divided into two categories, monitoring actions and state-change actions. Monitoring actions are those actions performed by the expert which does not alter the state of the domain or domain simulation. Examples of monitoring actions would be inspecting a gauge, observing the state of a switch, or querying the value of an ohm meter.

State-change actions are those actions performed by the expert which alter the state of the domain simulation or the domain itself. These might be opening or closing a valve, turning a rheostat, or flipping a switch on or off.

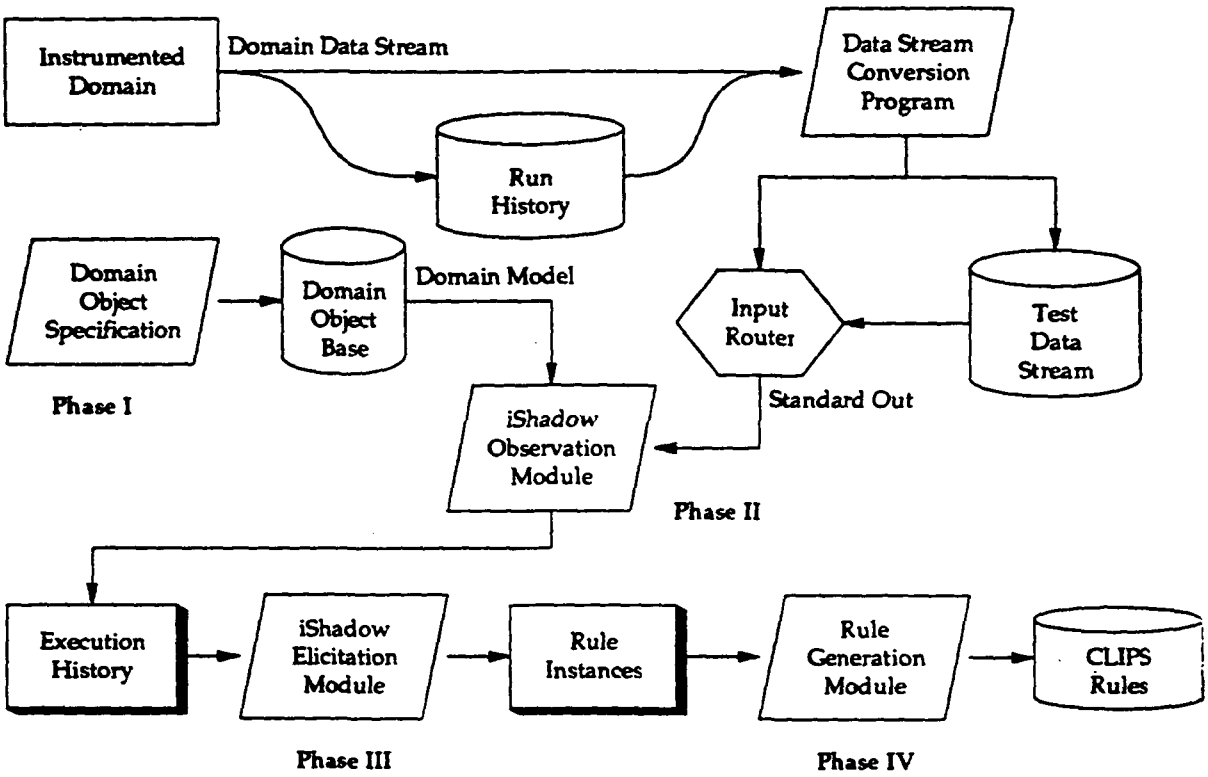

Figure 1: *i*Shadow Operational Flow

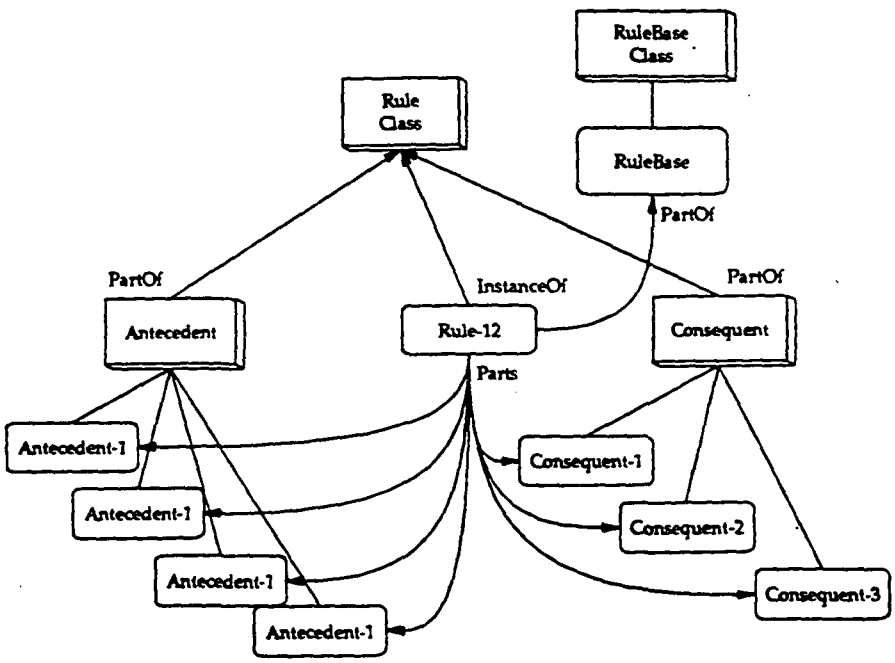

Figure 2: Organization of Rule Tuples in Object SimBioSys

Meta-Knowledge

Meta-knowledge refers to the set of heuristics applied by the expert which lead them to perform actions in a particular order. While *i*Shadow does not capture this class of knowledge automatically, it does elicit explanations and rationale from the expert regarding their selection of actions to perform

PASSIVE KNOWLEDGE ACQUISITION

The focus of this work has been in the development of a knowledge acquisition approach based on passive observation. Passive acquisition was chosen as the acquisition paradigm to emulate the observation process performed by a human knowledge engineer. The system is called *i*Shadow for an intelligent Shadow of the expert's actions.

Since the projects initiation, other efforts, such as [2], have also investigated the automated learning of diagnostic knowledge. These have focused on the automated acquisition of knowledge for a expert system as opposed to an ICAT system.

Project Goals

The goals of this effort have been refined as the project progressed and a better understanding of the problem and feasibility of the approach became more clear. At first our goal was to capture all classifications of knowledge but it soon became apparent that certain tasks were not easily captured in a passive scenario. Tasks such as solving differential equations, or psycho-motor tasks, such as controlling the remote manipulator arm of the Shuttle were difficult or impossible to instrument.

The focus of this effort has homed in on the generation of initial knowledge bases. The goal is to obtain approximately 75% of the initial knowledge base required for system operation. Coupled with the above coverage goal is the goal to achieve this initial set of knowledge in approximately 25% of the time it would take to build the knowledge base with a knowledge engineer.

Prerequisites to Passive Acquisition

The *i*Shadow approach, as mentioned previosly is not amenable to all types of knowledge acquisition. Knowledge domains which are highly cognitive in nature and those requiring psycho-motor skills, for example, are not suitable to this approach. The domain must consist of a combination of physical environment that can either be instrumented or simulated and a set of actions performed by the expert which can be observed through the system instrumentation

While these requisites limit the range of domains to iShadow can be applied, they also represent a significant portion of the types of domains and tasks for which ICAT systems are developed.

Four Phase Process

The automated acquisition process followed by iShadow is divided into four distinct phases, as illustrated in Figure 1. These are:

1. Domain Object Specification
2. Expert Observation
3. Elicitation and Rule Construction
4. CLIPS Rule Generation

Phase I: Domain Object Specification

In the first phase, a specification of the objects which comprise the expert's domain is developed. These objects are defined using the iShadow domain class editor. These classes define the generic set of entities and their states for the domain. Individual instances are then created to mirror the particular configuration of the system. Each configuration is called a model instance within *i*Shadow. Multiple models may be defined for a given domain.

Phase II: Expert Observation

In the second phase *i*Shadow connects to the instrumented system with which the expert interacts. As the expert performs activities in solving the domain problem, these are captured by the instrumentation and sent to the observation component. The activities captured are logged by *i*Shadow and form a temporal chain of events performed by the expert. Multiple observations may be captured for a single model.

Phase III: Elicitation and Rule Construction

After a set of observations have been performed, *i*Shadow then performs an elicitation and rule construction process. The elicitation consists of playing back the various sets of observations and querying the expert for rationale regarding their decision process, why certain actions were performed before others, and any strategy explanations they may have. These comments help to describe the meta-knowledge regarding the domain and the task ordering performed by the expert. *i*Shadow does not perform any generation of rules based on these elicitations but does embed them within the generated rules as comments. Thus enabling the knowledge engineer to add such strategic knowledge to the generated knowledge base.

The rule construction process begins by segregating the observed actions into sets of tuple pairs. These tuples consist of expert actions which represent the components of a rule. Figure 2 illustrates the organization and relationships between the various rule components.

The first element of the pair consists of observation tuples. Each observation action is encoded into an observation tuple. A set of these tuples is constructed until a state-change action is encountered. This set of observation tuples becomes the initial primitive set of antecedents for a rule.

iShadow then processes the state-change actions, collecting them into a set of action tuples until an observation action is encountered. The current set of action tuples is then paired with the current set of observation tuples to form a basic rule. This process continues until the end of the action list is encountered. If the end of the action list is encountered during a observation tuple set then iShadow disregards the current set of observation tuples being collected and stops processing. The basis for disregarding the observation set is that the previous tuple set had, in fact, been the end of the problem set and the last set of observations with no associated action were simply verification by the expert that the diagnosis and last action were indeed correct.

iShadow the adds to each set of observation tuples tests for the observed state of the objects references in the observation tuples and the current state of the objects referenced in the action tuples. This ensures that the rule will reflect the full set of states for all objects referenced within the rule.

The next step in the generation process takes the set of rule tuples and builds a symbol table consisting of all specific object references within the tuples. For example, the expert may have inspected gauge-1 and closed valve-2. While this level of specificity is appropriate for the selected model, the goal is to develop rules appropriate for the domain in general and not a specific model instantiation. Thus a symbol table is built and a single variable name is generated for all unique object names in the tuples. These variables names are then inserted into the tuples, replacing the specific object references.

Once the specific names have been replaced by variables, *i*Shadow then compares each rule in the generated rule base to each of the other rules. The process identifies those rules which are duplicates, have common antecedents or consequents. Duplicate rules are removed from the set. Rules with common antecedents but different consequents are merged and a notice to the developer is generated identifying the merged rule and the two original forms. Rules with different antecedents and common consequents are flagged for further inspection by

the developer as they represent some potentially serious gap in the knowledge base. Typically these last occurrences indicate some aspect of the model or domain has not been adequately captured and/or represented. For example, there may be some connectivity relationship which was not specified in the original model which reflects a distinct set of conditions.

Phase IV: CLIPS Rule Generation

The forth and final phase of the acquisition process is the generation of the CLIPS rule base from the internal rule tuples. This component has been intentionally designed as a modular, back-end process to allow for the generation of other rule syntaxes.

Implementation

*i*Shadow has been implemented on a SUN SPARC Station. It runs under either OpenWindows 3.0 or X Windows with OSF/Motif 1.1 or later. The system core was developed using SimBioSys, a object-oriented programming and knowledge representation system developed at SPS.

The current version uses the serial port of the SUN SPARC to provide an input channel for the observations generated via the instrumentation of the expert's domain. The link to the input port is purely a software link and the user can write any program required to provide input to iShadow.

To establish a communication link to the instrumentation, the user selects the Communication option in the Acquisition menu of the main window. This brings up a dialog box in which the user specifies any UNIX command which writes to standard output. The command is then executed in a background child process by *i*Shadow which also sets up a read pipe between the background child process and the main process. This enables *i*Shadow to simply read the output of the process.

CONCLUSIONS

The initial conclusions, based on observations of *i*Shadow in operation on simple domains is that *i*Shadow can generate a set of rules based solely on observations of the expert performing their tasks. A critical issue is the focus on procedural tasks, particularly in the diagnostic and troubleshooting area for physical systems and domains in the computer software realm. These domains provide opportunity for instrumentation which is critical to the success of the acquisition process.

More work is needed to refine the generation process. Initial work has been completed in the identification of duplicate rules, merge rules, and conflicting rules. More data needs to be gathered which can be used to determine the efficiency with which iShadow

Initial experiments have also been performed to test the feasibility of using iShadow to perform dynamic analysis of a rule base in action. Because the domain objects, expert actions, inference engine components and internal rule representations are all defined within SimBioSys, SimBioSys can control the execution of the rule set. This allows SimBioSys to perform single step execution of the rule base. Between each step, the conflict set and working memory can be examined and analyzed. This provides a significant capability not available in other expert system shells. It, in effect, provides a introspection of the behavior of the rule set at run time.

REFERENCES

[1] Gaines, B.R. and Shaw, M. L. G., *Eliciting Knowledge and Transferring It Effectively to a Knowledge-Based System,* IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 1, p. 4-14, February 1993.

[2] Giordana, A. et al, *ENIGMA: A System That Learns Diagnostic Knowledge,* IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 1, p. 15-28, February 1993.

[3] Han, J., Cai, Y., and Cercone, N., *Data-Driven Discovery of Quantitative Rules in Relational Databases,* IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 1, p. 4-14, February 1993.

[4] Lenat, D.B. and Guha, R.V., *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., 1990.

[5] Lenat, D. B., Prakash, M., and Shepherd, M., *CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks*, AI Magazine, 6, 4, Winter, 1985.

[6] Lenat, D. B., AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search, Rep. No. STAN-CS-76-570, Computer Science Dept., Stanford University.