A Study on the Explanation Scheme using Problem Solving Primitives

Gye Sung Lee

Dept. of Software, Dankook University, Korea
E-mail: gslee@dankook.ac.kr

Abstract

Knowledge based system includes tools for constructing, testing, validating and refining the system along with user interfaces. An important issue in the design of a complete knowledge based system is the ability to produce explanations. Explanations are not just a series of rules involved in reasoning track. More detailed and explicit form of explanations is required not only for reliable reasoning but also for maintainability of the knowledge based system. This requires the explanation mechanisms to extend from knowledge oriented analysis to task oriented explanations. The explicit modeling of problem solving structures is suggested for explanation generation as well as for efficient and effective reasoning. Unlike other explanation scheme such as feedback explanation, the detailed, smaller and explicit representation of problem solving constructs can provide the system with capability of quality explanation. As a key step to development for explanation scheme, the problem solving methods are broken down into a finer grained problem solving primitives. The system records all the steps with problem solving primitives and knowledge involved in the reasoning. These are used to validate the conclusion of the consultation through explanations. The system provides user interfaces and uses specific templates for generating explanation text.

Keywords: Explanation Generation, Problem Solving Primitives, Evidential Reasoning

1. Introduction

A generic architecture for knowledge based systems includes two primary components: the knowledge base and the reasoning mechanism [1,2]. The knowledge base captures the knowledge of human experts in a form defined by knowledge representation scheme. The inferencing engine searches the knowledge for deduction from given information. This simulates the problem solving operations of the expert. The interaction between the user and the system sometimes requires the system to explain its behavior to the user about system’s reasoning behind its conclusion. This helps the user understand how the system draws conclusion and what knowledge is involved in the reasoning process [3,4]. Reasoning proceeds in the inferencing engine by repeating the evidence collection and evidence combination. The user and the expert may interest in the reasoning process of what knowledge is involved and how the problem is solved.

Explanation facilities of the system are key to the successful maintenance of knowledge based system because explanations provide essential information when validity and consistency of the knowledge base is in question. Lack of knowledge or inconsistent knowledge can be found by tracing mechanism over the problem solving history. Explanations may facilitate to detect deficiency of the knowledge during the time of development and during actual use in the field [4]. Deficiencies include syntactic and semantic errors. Both
errors have to be detected and need to be corrected. Most common way to find the discrepancies in the knowledge base is running the cases and checking the results to see if they conform to the expectations of the user. If they differ, the user starts looking into the details of the run. In most systems, this type of debugging facility is not provided because it is not easy to present the details of reasoning mechanisms to the user. The feasible solution to generate the information needed for discrepancy detection is by generating explanations. It is known to be a difficult task to find semantic deficiencies. Running test cases is a good way for finding semantic deficiencies with the help of explanation tool. Through the examination over the generated explanations, expert can discover the deficiencies of the knowledge base and suggest different kinds of repair such as adding new knowledge, deleting knowledge, and modifying current knowledge [5,6].

To build a successful knowledge based system, efficient and effective inferencing engine should be designed and constructed along with complete and consistent knowledge base. In early knowledge based system, generic reasoning mechanisms are adopted regardless of the tasks. This limits the adaptability of the system because reasoning mechanism may differ depending on the type of application. The versatility of the system is also limited because it is hard to refine the knowledge base. To build an adaptable and versatile system, specific reasoning mechanism that can be abstracted by a problem solving method needs to fit to features of the specific problem tasks (e.g., classification, planning, diagnosis, etc.). Strong connection between the features of the problem and the problem solving methods may lead to the successful development of knowledge based systems [7].

Another issue for reliable performance is the use of explanations [3,4]. The system takes the complete solution for a problem and records all the steps as its trace of the run. These are used for explanation generation. Most of systems equipped with explanation scheme use these set of traces and simply provide the knowledge itself involved for the reasoning. This is called a feedback explanation scheme [8,9]. For better explanation, the system requires to keep track of both knowledge and problem solving strategies. Since the problem solving strategies follows the state of mind of human expert when they try to solve a problem, these should be modelled explicitly. These strategies called meta knowledge, or operational knowledge can be captured and utilized for explanation generation because these can mimic human cognition in problem solving [10-12]. In order to keep track of the steps of the run, the step has to be clearly and explicitly represented in a form of record structure: knowledge involved in the reasoning, problem solving methods applied to the current situation, and evidence used for reasoning. The explicit modeling of problem solving structures is the key to success for explainable reasoning. In our reasoning structure, this problem is handled with the provision of more detailed and explicitly represented problem solving primitives for smaller subtasks. These primitives are configured to explicitly express the overall control structure. This clear and modular definition of subtasks and explicit control structures benefit system designers and knowledge engineers in designing and implementing more effective explanation scheme.

The problem solving method framework has been presented in the earlier work for the design and implementation of efficient evidential reasoning. In this research, we will present another issue, explanation generation, based on the problem solving primitives. We will show that problem solving primitives can also be well utilized in the field of explanation generation. We will also show that explanation is composed of not only the trace of knowledge involved but also the human expert’s flow of steps which are explicitly expressed as problem solving strategies. It is also required for the system to provide a convenient interface for interaction between the user and the system. This interface makes it easy to trace back and forth the steps of reasoning and eventually help the user to understand system’s behavior on how and why the system goes through the process.

2. PROBLEM SOLVING PRIMITIVES

In the problem solving method (PSM) framework, the features of the application at hand needs to be identified and analyzed first. The appropriate problem solving method is then searched and selected for reasoning mechanism. The selection of the problem solving method is intended for system’s most effective
reasoning embedded in a form of operational knowledge. The PSM customized to the task is expected to perform well because it reflects an underlying human expert’s process of reasoning. Since the selected problem solving method bears important features of human expert problem solving behavior, the reasoning scheme performs as effective as intended for a specialized task.

Problem solving methods exploited for specific tasks (troubleshooting, scheduling, design, decision making, diagnosis, etc.) can also be used for acquiring knowledge as well as for reasoning [13, 14]. For example, for troubleshooting problem, a PSM called cover and differentiate can be designed to fit for the task. The use of this explicit PSM in this task implies that the system seeks domain knowledge in particular forms to achieve problem solving goals. The cover and differentiate PSM also implies that the system attempts to relate hypothetical causes of the problem and their symptoms in two steps (i) for given sets of symptoms the system needs to know what set of hypotheses can cover or explain these symptoms, and (ii) narrowing down the current set of hypothetical invokes a question of what other information is needed to further differentiate among them. This helps to focus on a small section of the search space. It thus leads to more effective and efficient reasoning. If something is found missing for further discrimination in a given problem solving state, the knowledge elicitation can be initiated to find more knowledge to achieve the purpose of discrimination.

The geological application that we choose as the domain in this research is a kind of classification problem. The features of the problem solving behavior in this domain are mainly to characterize and estimate hydrocarbon prospects. This type of feature makes the PSM fall in the category of characterization problem. It also contains goal oriented behavior of reasoning. Therefore, the most appropriate PSM would be the establish and select PSM. This PSM is composed of two intermediate PSMs, the establish PSM and the select PSM. If PSMs are further subdivided, they become smaller problem solving modules each of which is able to perform a subdivided smaller task, called problem solving constructs (PSCs). These form a hierarchy of the PSM. The problem solving module at the leaf node of the hierarchy is called problem solving primitives (PSPs). The PSP has a role in the process of the reasoning for the particular application and can be easily implemented. The reasoning process can be represented as a sequences of PSPs. Each step with a PSP corresponds to a state of problem solving process. These states can be used to form a state transition diagram which will be explicitly defined. This diagram defines the control structure of the entire reasoning. This is subsequently used to guide the system to perform reasoning for a task. The design framework using PSPs develops a scheme for explanation generation.

3. EXPLANATION SCHEME USING PROBLEM SOLVING PRIMITIVES

The primary approach to generating explanation is to adopt explicit problem solving methods which are tightly connected to the feature of the task. The features in a task framework are integrated into two components: kind of knowledge required, and the reasoning strategy used. These constitute basis for the task architecture and hence for explanation generation. The reasoning strategies should be represented as a set of smaller and more detailed problem solving constructs. Then the entire reasoning can be represented by those primitives and the entire reasoning can be presented to the user and expert. This enables the user and the expert to examine and understand how and why the reasoning proceeded. From the information extracted from the detailed trace of reasoning, the user and the expert can identify the cause of discrepancies. If the PSPs are defined coarsely, the explicability and thus comprehensibility would be low because of the level of the details [5]. To generate detailed explanation useful for knowledge base maintenance, the PSPs should be fine grained. This kind of explanation helps the user to understand how the system draws conclusions with a given set of evidence. On the other hand, if the reasoning mechanism is hard-wired, it is almost impossible to trace the reasoning and to reconfigure the reasoning structure [15]. This causes the configurability and expandability problem.

The problem solving context can be defined by the detailed PSPs and other extra information such as evidence, hypothesis, and rules involved in a certain stage of the reasoning. The degree of detail for representing the problem solving context is determined by the degree of detail in problem solving methods. The problem solving context with a PSP is defined by a problem solving state. The next state would be
A Study on the Explanation Scheme using Problem Solving Primitives

determined by the result of the current PSP’s action. If the PSP needs a query to ask for other information, evidence seek step is required and the system presents a query to the user. After the query, applications of this primitive to the current state of the system could lead to a new state. Therefore, a framework with problem solving primitives enables the entire problem solving to be explicitly represented as a state diagram. The state diagram provides the basis for more explicit and detailed explanations of problem solving behavior, which is then a key to performing better explanation generation [12].

The reasoning mechanism, as presented as in Figure 1, is defined by a group of PSPs. The node is one of two types: a circled node corresponds to a PSP and rectangular node is a state which is not associated with PSP. This is called as non-PSP associated state. These states are needed to control the reasoning process other than PSP initiated state transition. For example, the top-level goal node is used to set up goals (hypotheses) at the initial stage of the consultation. Similarly, the new top-level goal node is also a non-PSP associated state that choose a top-level goal among other goals. The user init state in the middle of the diagram is initiated when the user takes the control of the reasoning regardless of the system’s suggestion for next move. The user either alters the current goal or looks for another evidence that the user would think more relevant at the current stage. State transition takes place depending on the result of the PSP execution at the current state.

![Figure 1. State Transition of Problem Solving Primitives](image)

The establish PSM is composed of five sub-components which are called as problem solving constructs (PSCs): (1) get goal, (2) evidence seek (evid seek), (3) evidence cover (evid cover), (4) evidence combination (evid comb), and (5) exit check (exit chk). As a first step of the reasoning, goals are set up by the get goal PSC. These goals are named as primary goals. Intermediate goals are called as sub-goals. The goal is defined by a hierarchy of sub-goals. To achieve a top-level goal, sub-goals under it have to be achieved first. The evid seek PSC is to search for evidence to investigate the validity of the current goal. The evid seek PSC is composed of two PSPs: the pre-information seek PSP and the query/sub-goal PSP. The latter PSP asks a query to get evidence or it can set up a sub-goal. To validate the current goal, it sometimes needs to go down to the sub-goal and investigate the sub-goal. Once the sub-goal is confirmed, then the higher goal is investigated. If this is the case, the establish PSM is requested for achieving the sub-goal.
The reasoning system keeps looking for evidence to satisfy the conditions of the rule. When each condition is met, this is marked out (evid cover). When all the conditions are met, then the rule is executed either to confirm or disprove the set of current goals. This is done by the evid comb PSC. The Dempster-Shafer (DS) evidence combination scheme [16] is adopted for the internal computation for evidence combination. Two types of evidence combinations are attempted: regular and hierarchical DS combination. When the current sub-goal is achieved, the system returns back to its parent goal and continues reasoning until all the sub-goals are met. If all the primary goals are achieved, then the exit-chk PSP lets the system stop the reasoning. The exit-chk PSP also lets the system to stop reasoning when it runs out of all the evidence even though no conclusion is drawn.

After establishing the current set of goals, we need to select which goals are to further focus on. This process would be fulfilled by the select PSM. The select PSM is composed of three select PSCs: hypothesis selection, PSP selection, and evidence selection. The hypothesis selection PSP is used to choose one or more hypotheses for further investigation. A state is defined by a number of parameters such as the current set of hypotheses with degree of belief, rules associated with current hypotheses and known evidence. For PSP selection, the current set of hypotheses are checked. If the top-ranked hypothesis in terms of belief value is dominant among active hypotheses, the hypothesis is more likely than others at this point. So the system will focus on the hypothesis. The max-hyp PSP is selected to perform in this case. The dominant hypothesis is determined when the difference in belief values of two top-ranked hypotheses is greater than a certain predefined threshold.

If belief values of a group of hypotheses are greater than the rest of the hypotheses set, i.e., if they are stratified into two groups, then the filter PSP would be chosen for further investigation. This PSP will look for evidence that supports the highly believed group of hypotheses. The filter PSP split the hypotheses set into two group, one termed as positive group and the rest as negative group. The filter PSP looks for evidence to discriminate the two groups. The evidence is searched that strengthens the positive group. By the property of evidence combination, the belief in the negative hypotheses group will go down. Lastly, if there is no dominant hypothesis nor stratified group of hypotheses is found, then the diff PSP would be adopted for further investigation. The purpose of the diff PSP is to look for evidence that could differentiate among the set of hypotheses.

When the PSP selection is done, the control is passed to the evidence selection state where rules are examined to achieve the goal of the selected PSP, max_hyp, filter, or diff PSP. In order to find the best evidence from a set of rules, rules are preprocessed and transformed into an internal structure as a form of evidence and hypothesis network. This network provides a useful tool for easy and quick evidence search for a specific hypothesis. In general, since a set of evidences are retrieved from the network for a set of hypotheses, we need to find a method to distinguish them. The contribution measure is introduced to evaluate the impact of an evidence as shown in Eq. 1.

\[
\sum_{i} \frac{Bd (r_i)}{UH S_i} = \sum_{j} \frac{Bd (r_j)}{UH S_j}
\]  

(1)

The measure has two terms. The first term evaluates the degree of belief in a leading hypothesis or a set of positive hypotheses. \(Bd (r_i)\) is the degree of belief of rule \(i\) which is in support of hypothesis under investigation. This value is divided by the number of evidences (LHS\(_i\)) in the rule which are not instantiated. This implies that if there are more evidence to be instantiated for rule firing, it is not likely soon enough to support the hypothesis under consideration. These values are summed up for all the rules associated with the hypothesis of interest. The second term estimates the support for negative hypotheses. The best evidence for the selected PSP would be the one that maximizes this first term and minimizes the second term. For max-hyp PSP, the second term is ignored because it requires estimate for the leading hypothesis. The diff PSP and the filter PSP uses both terms for computing the measure but the diff PSP may not know positive and negative hypothesis until an evidence is chosen. Evidence is selected as a result of evidence selection PSC among un-instantiated evidences. If the evidence is associated with a query, the control goes to the evid collect state and
ask a question to get information for the evidence. When the evidence is associated with a sub-goal, the control goes to the new node goal state and starts over the new cycle with the new sub-goal. The overall reasoning process is shown in Figure 2. In the beginning, the top-level goals are listed up. The system gets all the known initial information from the user, which establishes a top-level goal. For the first goal, the select PSM is applied to select hypothesis, PSP, and then evidence in order. If the evidence is a query, then the establish PSM will be applied. If it is a sub-goal, then the select PSM will start.

Figure 2. Main Reasoning Control Structure

4. EXPLANATION GENERATION

Task based explanation scheme takes advantage of explicitly defined PSPs. The explanation by means of problem solving structure provides more useful information to the user. The reasoning control is also clearly and explicitly defined as a form of state diagram. The use of these structures in reasoning helps the system to generate more practical explanations for understanding the system’s behavior. This results from the fact that the problem solving strategies of human expert are explicitly contained in the problem solving primitives. When explanations are used for user’s understanding and for the purpose of knowledge refinement, one of the key tasks is to have the ability to trace case runs with detailed records of problem solving constructs. Details of how the problem was solved are the basis for deficiency detection as well as for aiding the user to understand the process. These include associated PSP, rules, evidence, and belief values that scale the support of the conclusion. The belief values enable the user to perform quantitative analysis. This information is collected in evidence combination stage during the reasoning process.

Explanations are constructed by using templates for each PSP and created in text form. Template is designed with predefined text and slots. The content needed for the slot is extracted from information recorded during the consultation. The reasoning for a goal is made of a cycle with two PSMs, establish and select, and their PSPs. Reasoning proceeds by going through the states. Once templates are completely designed, the filling the templates is needed for text generation. One cycle can form a chunk of explanation with a lot of detailed information and defined as a stage. The stage includes nine events:

1. Event: a list of events that took place in the current stage.
2. Goal: the current goal/sub-goal of the stage.
3. Leadhyp: the dominant set of hypotheses for the goal/sub-goal.
4. PSP: problem solving primitive selected by PSP selection rule.
5. EvidSeek: After evidence selection, list of evidence needed.
7. Rule: list of rules fired at this stage.
8. Hyp: list of hypotheses with new belief values as result of evidence combination
9. GoNext: depending the direction, the trace goes to next stage.

The explanation interface shown in Figure 3 is designed to fully display information mentioned above. Each event is mapped to a button. By clicking the button, the details of the current event are displayed. The backward button and GoNext button at the lower right part of the interface help the user to move back and forth the trace of the reasoning. Figure 3 shows a snapshot in the middle of the trace. The domain we chose is a geological domain, especially oil exploration. The goal is to identify an important geological characteristic, hydrocarbon play called facies [17]. We need data from a number of geological observations and data from indirect data sources. Facies is directly linked from another geological characteristic, depositional setting.

The stage of Figure 3 is right after control returns to parent goal, facies from a sub-goal, depositional setting. After evidence selection PSP, the selected evidence looks for information from the user via a query. When Query button is clicked, the detailed information about the query is created and displayed on the interface screen. The Goal button show the current goal of the stage, which is facies. To see the leading hypotheses of facies, the LeadHyp button will show the hypotheses with belief values. The user can evaluate the confidence of the result. The same information can be shown in multiple interface displays. The description of the query contains leading hypotheses and positive and negative hypotheses.

![Figure 3. Explanation Generation Interface](image)

The PSP button is used to see what is the result of the PSP selection, which is one of Max_hyp, Filter, and Differ PSP. From this, the user can conceive how the set of hypotheses are competing and their distribution in terms of belief values. As indicated in the fourth line of the display, the system selects a query expecting possible response that could achieve the goal of the PSP Differ. The evidence collected from the query would result in differentiating the current leading hypotheses for facies. The Rule button provides information about rules fired at the stage. Evidence used for rule firing can be found by clicking the EvidSeek button. As a result of rule firing, the belief values of the hypotheses are updated. The Hyp button is used to show the updated values. The Event button is used to display the trace of the events that took place at this stage of the run.

5. CONCLUSION

In this paper, we studied the scheme of explanation generation based on the problem solving primitives. The problem solving method paradigm is well presented in the earlier work for the design and implementation of efficient evidential reasoning. We have shown that it can also be well utilized in the field of explanation generation. The key applicable areas of explanation generation are the knowledge refinement and maintenance of reasoning system. Interaction is made possible for the user to trace the reasoning process by examining the explanations generated by the system. This is important not only for the user to get trust on the system but also
to understand system’s behavior on how and why the system draws the conclusion. Moreover, the validity of the conclusion can be further investigated with quantitative analysis. For more profound explanation, we added problem solving strategies in addition to knowledge-level traces. These strategies which are sometimes called meta knowledge play an important role in explanation generation because these mimic the flow of human expert’s process in problem solving. The key design consideration for explanation is that the reasoning should be explicitly described with degree of detail. Introducing a set of explicit and finer grained problem solving primitives is a major part in designing explanation scheme. We have shown that explanations are generated with level of detail. The user interface provides the user a convenient tool to trace back and forth over explanations. In the future, the research will be further expanded to different domains by introducing various types of problem solving methods.

REFERENCES