Establishing the Coherence of an Explanation to Improve Refinement of an Incomplete Knowledge Base

Young-Tack Park and **David C. Wilkins**
Computer Science Department
405 North Mathews Ave.
University of Illinois
Urbana, Illinois 61801

Abstract

The power of knowledge acquisition systems that employ failure-driven learning derives from two main sources: an effective global credit assignment process that determines when to acquire new knowledge by watching an expert's behavior, and an efficient local credit assignment process that determines what new knowledge will be created for completing a failed explanation of an expert's action. Because an input (e.g., observed action) to a failure-driven learning system can generate multiple explanations, a learning opportunity to extend the incomplete domain theory can go unobserved. This paper describes a failure-driven learning with a context analysis mechanism as a method to constrain explanations and thereby increase the number of learning opportunities. Experimentation using a synthetic expert system as the observed expert shows that the use of context analysis increases the number of learning opportunities by about 47%, and increases the overall amount of improvement to the expert system by around 10%.

Introduction

Knowledge acquisition is the major bottleneck in the development of expert systems. One promising method to overcome this difficulty is exemplified by the Learning Apprentice System [Mitchell *et al.*, 1985] and other apprenticeship learning programs, which assimilate new problem solving knowledge by observing and analyzing a human expert's actions [Kodratoff and Tecuci, 1987] [Wilkins, 1988a].

Apprenticeship learning involves first recognizing a failure to interpret an expert's actions and then acquiring new knowledge to recover from the failure. The detection of a failure can be done either by outside human experts or by the apprenticeship learning system itself. When outside human experts are used, the humans point out the the expert system's failure and provide the apprenticeship learning system with a learning opportunity [Mitchell *et al.*, 1985] [Kodratoff and Tecuci, 1987]. Hence, in such systems, the global credit assignment as defined in [Dietterich and Buchanan, 1981] is done by a human and is not addressed in the learning process.

In contrast, when an apprenticeship learning system recognizes a failure by watching a human expert's problem solving steps, it must employ a global credit assignment process to determine when to learn. For example, the ODYSSEUS apprenticeship learning program [Wilkins, 1988a] for the HERACLES classification shell [Clancey, 1987] watches an expert and tries to explain the expert's observed actions. An explanation in ODYSSEUS is created by backward chaining the meta-level strategy rules. When ODYSSEUS fails to explain an action, it assumes that relevant facts are missing from its knowledge base.

Apprenticeship learning systems use diverse means to acquire new knowledge from failures [Mitchell *et al.*, 1985; Kodratoff and Tecuci, 1987; Wilkins, 1988a]. The most common approach is to construct an explanation of the failure. ODYSSEUS suggests new knowledge that can complete a failed meta-rule chain when this knowledge is added to the knowledge base.

Therefore, the power of knowledge acquisition systems that employ an apprenticeship learning method derives from two main sources: an effective global credit assignment process that determines when to acquire new knowledge by watching an expert's behavior, and an efficient local credit assignment process that determines what new knowledge will be created for completing a failed explanation of an expert's action.

Because a human's observed action can be explained in many different ways, a learning opportunity can go unnoticed. If the learning system does not filter out implausible explanations, its performance may be seriously hampered. In order to profit fully from watching an expert, the learning system must be able to reason about explanations as well as generate them.

In this paper we present an effective mechanism to avoid masking of learning opportunities by multiple explanations in apprenticeship learning system. The method reasons about the generated explanations and has been demonstrated to improve the performance of the learning system. We also offer an efficient method to suggest missing knowledge to recover from failures through a repair module that employs top-down and bottom-up approaches to infer new knowledge. The repair program reasons about failures, suggests new knowledge based on induction over solved cases, and verifies the suggested knowledge based on the context.

Explanation Generation

When a human expert takes an incomprehensible action and fails to do what a learning system expects him or her to do, failure driven learning attempts to understand the action by generating plausible explanations of the expert's behavior. Understanding is a process that explains the incomprehensible action in a context-sensitive way. Hence, it is important to represent explanations and contexts declaratively.

Experts usually employ problem solving strategy knowledge that is obtained through experience. Therefore, strategy knowledge is an important part of explaining an observed action. We implemented the MINERVA expert system shell [Park *et al.*, 1989] that uses explicit and flexible strategy knowledge. MINERVA is a rule-based expert system shell that is a Prolog reimplementation and enhancement of HERACLES expert system shell [Clancey, 1987]. The enhancement was guided by experience gained from the ODYSSEUS [Wilkins, 1988b] apprenticeship learning program for HERACLES. In MINERVA, the representation of the meta-level strategy knowledge is more explicit, declarative, and modular. This strategy knowledge is a Horn clause. The following is an example:

```
goal(clarify_finding(Finding1)) :-
            new_datum(Finding1),
            not value(Finding1, no),
            clarified_by(Finding1, Finding2),
            not concluded(Finding2),
            goal(findout(Finding2)).
```

The head of this clause consists of name and argument of this strategy knowledge source. The body consists of a sequence of premises followed by a subgoal. A strategy knowledge source is activated only by a new change of problem state and invokes a subgoal that changes problem states. When the premises of an activated knowledge source are satisfied, the subgoal is inserted into the agenda. MINERVA employs explicit schedule knowledge to determine which subgoal to perform next. Subgoals are of two types: actions and tasks. An action refers to a subgoal that tries to find out a symptom. A task refers to a subgoal that invokes another task. For example, a subgoal *apply_rule(Rule)* represents a task that fires a domain rule. It triggers a sequence of tasks that find the values of antecedents of the domain rule, checks if the values satisfy conditions, and evaluates the domain rule. The modular representation and the opportunistic control of strategy knowledge source is suitable both for flexible problem solving and efficient knowledge acquisition.

We view an expert's action as a coproduction of strategy knowledge, domain knowledge, and problem solving state. Finding relevant explanations of an expert's action hinges on the notion of differential modelling, whereby the expert's behavior is compared to the expert system's choices for the same circumstances.

The modular knowledge sources predict all reasonable actions that an expert might take based on the new problem solving states. The problem solving states can be acquired from the expert or by inferring from the expert's problem solving steps. Predicted actions and tasks are justified by the explanations that describe why those are expected by the expert system. And, we will say that apprentice understands an observed action if it can construct an explanation of the same expected action or the relevant task.

Failure Detection

If an apprenticeship learning system fails to generate an explanation of an expert's action, this suggests that the knowledge base is missing a piece of information that the expert knows. This kind of explanation failure is categorized as the *expectation failure* type.

While the detection of the expectation failure is important, it is also important to be able to recognize and learn from situations where explanations are not coherent with a problem solving context. When failure driven learning encounters a situation where all the generated explanations fail to be consistent with the problem solving context, it discards them and tries to understand the action by finding a new explanation that is coherent with the context. This kind of explanation failure is categorized as the *context failure* type.

For example, suppose a human physician asks a question on *headache_duration*. Perhaps she wants to clarify a finding *headache*, because *headache_duration* is a more specific fact. Or, she might intend to differentiate two diseases; the symptom of one disease hypothesis is long *headache_duration* and the other has short duration. Hence, a learning system cannot identify which explanation is a plausible intention of the question without using a bias. However, if the physician asked many questions on the detailed facts of *headache* just before, her intention is more likely to be clarification of *headache*. In contrast, if she asked questions on both diseases, she is probably trying to differentiate between the hypotheses. We need meta level reasoning about generated explanations to judge the quality of these multiple explanations.

Our model of knowledge acquisition includes the detection of both types of failures and the generation of explanations for the failures (see Figure 1). The global and local credit assignment processes are guided by reasoning about strategy knowledge and context analysis.

The scope of this work is limited by the assumption that the strategy knowledge of the expert system is complete, consistent, and correct. Thus, when a discrepancy arises between what the learning system expects and what the human expert apparently does, the learning system is justified in assuming that strategy knowledge is right and domain knowledge is wrong.

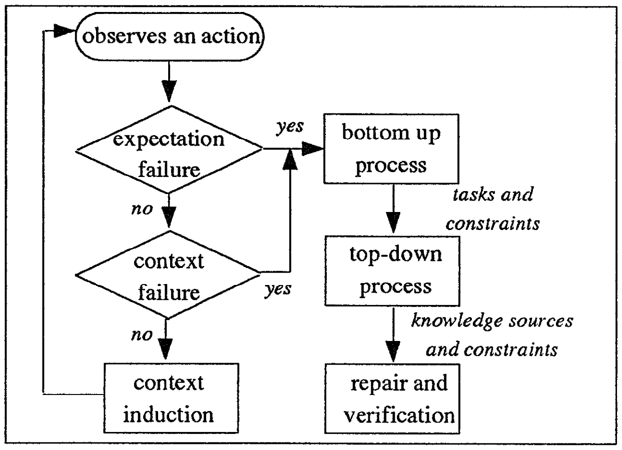

Figure 1: A knowledge acquisition cycle

Using expectations to generate explanations

We use an expect-explain method to understand observed actions (see Figure 1). As the learning program watches the human expert's actions, it builds a problem solving model which contains all the given facts, derived facts, and hypotheses. Whenever the learning program observes an action, it updates the problem solving model and predicts the next possible actions based on the problem solving model and its strategic knowledge (see section).

Expectation failure detection

In order for the learning system to explain an expert's action, it tries to connect the action to one of its expected actions. If the observed action is one of those predicted, then the learning system understands the action. The justification of the expected action is a candidate explanation for the expert's action.

An observed action may be related to an expected task. Suppose the system expects a task that executes a domain rule. If an observed action is one of the antecedents of the domain rule, the explainer assumes the task's explanation as a candidate explanation of the observed action. In order to find the path between the observed action and the task, the learning program runs meta rules backward. A declarative and explicit representation of meta rules makes it easy for the learning program to run the system backward.

When the learning system fails to find a candidate explanation for the observed action, this suggests that the knowledge base is missing the piece of information that prompted the expert to take the observed action. This *expectation failure* triggers the repair module to refine the incomplete knowledge base by finding missing knowledge that will allow a coherent explanation of the expert's actions.

Context analysis

An explanation is relevant to an observed action if it coheres with the problem solving context as well as addresses the action. To understand an action by finding an explanation, it is important to recognize whether an explanation is relevant to the context as well as the observed action. Even if a single goal may prompt the human expert to ask a question, when we try to find reverse mappings from the question to possible explanations, there will be many possible answers. To single out the correct explanation among them requires reasoning about the context where the action is taken.

In general, experts tend to take an action which is coherent with their problem solving context. For example, physicians usually ask follow-up questions on current hypotheses or symptoms. When human apprentices encounter difficulties to understand an expert's actions, they have an urge to find an explanation based on the human expert's problem solving context.

Constrain context failure

The learning program observes a human expert's action one at a time and generates a set of explanations of the action. An explanation consists of a strategy knowledge source name, such as *clarify_finding*, and a focus of attention such as *headache* (see a knowledge source example in section).

The context analysis program maps an observed action on the explanation plane (see Figure 2). The explanation plane has two axes. The strategy axis represents all the predefined strategy knowledge sources. The focus axis represents human expert's actions, derived facts, and hypotheses. An explanation on this plane has a pointer to a set of actions that are explained by the explanation. The context analysis program also keeps track of observed actions until a context failure occurs.

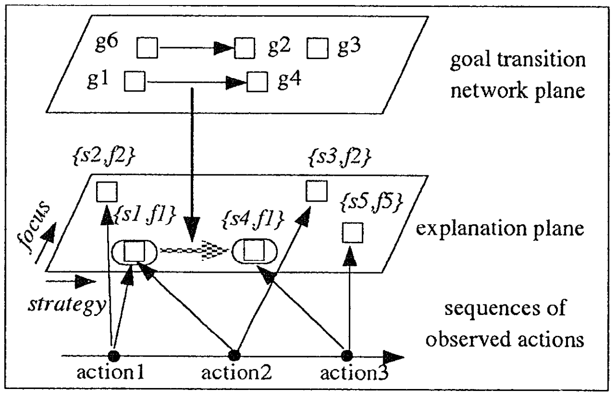

Figure 2: A context transition guided by a goal transition network

Since an action can have multiple explanations, the mapping may be one-to-one or one-to-many. An explanation on the explanation plane becomes a sub-context that explains a subset of observed actions. When the context analysis program inputs the set of explanations of the first action, it creates as many sub-contexts as explanations in the set. As the context analysis program inputs a succession of explanation sets, it creates many sub-contexts that explain a subset of observed actions. Since the context analysis program keeps track of observed actions, it can find a sub-context that explains all the actions. Such a sub-context is considered as the context of the observed actions.

However, this simple method to build a context may produce many redundant context failures even if the human expert changes his or her goal slightly. It is necessary for the context analysis program to understand such change and not to consider it as a context failure. We use a goal transition network to identify a natural goal transition from a context failure (see Figure 2).

There are at least three factors which can be justified as rational bases for people to make causal descriptions: contiguity, similarity, and statistical [Anderson, 1987]. Our context analysis program currently considers contiguity and similarity factors to constrain redundant context failures. By the similarity factor we mean similar contexts. Contexts are similar if their strategy axes are the same and their focus axes can be grouped by a known relation. For example, suppose `context-1` and `context-2` consist of (*clarify_finding*, *surgery*) and (*clarify_finding*, *neurosurgery*), respectively. Since *surgery* and *neurosurgery* is defined by a relation *more_specific*, a transition from `context-1` to `context-2` is considered as a natural one. This transition describes a human expert's natural stream of diagnosis that attempts to find out more detailed facts progressively.

By the contiguity factor we mean contiguous transition. A transition from `context-1` to `context-2` is contiguous if their focus axes are the same and the transition between strategy is declared to be plausible. For example, (*clarify_finding, X*) and (*process_finding, X*) [1] are different contexts. When a human expert starts to ask a question that is explained by (*process_finding, X*), after he or she asked questions related to (*clarify_finding, X*), the observed action can not be explained by the current context (*clarify_finding, X*). Hence, the simple context analysis program considers the explanation (*process_finding, X*) as a context failure and tries to repair the failure. However, the transition from (*clarify_finding, X*) to (*process_finding, X*) is a stream of the diagnostic reasoning sequence. This transition describes a human expert's reasoning approach that predicts hypotheses related to a symptom after he or she find out more detailed facts of the symptom.

A transition on the goal transition network plane represents such allowable transitions to guide context transitions. In the current system, the goal transition network is explicitly hand-coded.

[1] The first seeks more specific information and the second applies domain rules related to X.

In Figure 2, `action1` and `action2` are described by a context {*s1,f1*}. Because no transitions are known from {*s2,f2*} to {*s3,f2*} and {*s1, f1*} to {*s3,f2*} in the goal transition network, only {*s1,f1*} becomes the context when `action2` is observed. Then, the context analysis program projects explanations of `action3` for context {*s1,f1*}. If the context {*s1,f1*} were unable to cover any explanation of `action3`, the transition model would guide the induction program in considering the transition from {*s1,f1*} to {*s4,f1*} as a natural one.

Suppose projected explanations of an `action-n` are too far from the current context on the explanation plane and no paths in the goal transition model can explain them. Then the context analysis program considers two possible cases: the expert may have changed his or her goal or the system may be missing a piece of information. When the context analysis program encounters such a situation where no explanation coheres with the context, it saves the current context, restarts the context analysis by posting sub-contexts of `action-n`, and waits to see the next actions. If some of the sub-contexts explain subsequent actions, the context analysis program assumes the expert changed his or her goal. However, if the subsequent actions are explained by the saved context instead of the sub-contexts of `action-n`, the explanations of `action-n` are not coherent with the expert's problem solving behavior. The context analysis program recognizes the explanations of the `action-n` as a context failure. If a failure driven learning system does not employ this context analysis, multiple explanations may mask a learning opportunity in this situation.

Failure Recovery and Constraints

When the global credit assignment program encounters an explanation failure, it triggers a repair module. The repair module employs top-down and bottom-up approaches to infer new knowledge. While the top-down interpretation fires a set of knowledge sources to predict actions and tasks, the bottom-up interpretation accepts observed actions and finds tasks related to the actions by running the meta rules backward.

We employ a combined method (see Figure 1) that first runs meta rules backwards [Wilkins, 1988b] and then runs strategy knowledge sources forward. Both searches will suggest plausible paths which connect the observed action and the strategy knowledge sources, and also post constraints to predicates in the paths. These constraints and the use of the meta-level strategy knowledge enable the repair module to reduce search space.

Construction of explanation and constraints

When the learning system detects expectation failures and context failures, it invokes a bottom-up repair module that drives meta rules backward from an observed action to tasks (see Figure 1). Since in medical di-

Disease	Learning opportunities	Method-1		Method-2		
		expectation failure	multiple masking	expectation failure	context failure	multiple masking
Bacterial Meningitis	40	11	29	11	8	21
Brain Abscess	3	0	3	0	1	2
Cluster Headache	16	16	0	16	0	0
Fungal Meningitis	0	0	0	0	0	0
Migraine	5	0	5	0	2	3
Myco-TB Meningitis	6	6	0	6	0	0
Primary Brain Tumor	1	0	1	0	1	0
Subarach Hemorrhage	34	18	16	18	13	3
Tension Headache	3	0	3	0	3	0
Viral Meningitis	26	15	11	15	3	8
Totals	134	66	68	66	31	37

Table 1: Comparison of failure detection with and without context analysis. *Method-2* and *Method-1* represent the failure detection method with and without context analysis, respectively.

agnosis, the backward search may suffer from combinatorial explosion, the bottom-up repair module runs the meta rules backward from an observed action to tasks that are used as subgoals of strategy knowledge sources. Hence the backward search is not deep and the number of paths from the action to tasks is small. The bottom-up repair module generates constraints of the tasks as well as finds tasks that are related to the observed action. For example, suppose the human expert asks about *seizures* and the learning system fails to find explanations of the action. The bottom-up module runs meta-rules backward from the observed action to a task such as *apply_rule(domain_rule1)* and the constraint that *domain_rule1* must have *seizures* in the antecedent. The tasks and constraints of the argument generated by the bottom-up module are used to determine the subgoals of strategy knowledge sources.

The top-down repair module is run after the bottom-up module and determines the heads of strategy knowledge sources based on the context and sub-contexts. This assumes the human expert diagnoses a case in a context sensitive way. The top-down module instantiates a set of strategy knowledge sources using subgoals, constraints, and instantiated heads generated both repair modules. A strategy knowledge source in MINERVA is designed to be instantiated by the bindings of head and subgoal. Suppose the current context is (*explore_hypothesis*, *primary_brain_tumor*), then the top-down repair module instantiates *explore_hypothesis* strategy knowledge source and adds a constraint that *domain_rule1* must conclude *primary_brain_tumor*. In this example, *domain_rule1* has one antecedent *seizures* and must conclude *primary_brain_tumor*.

The constraints are used to remove an instantiated knowledge source from the candidate explanations of the unexplained action. The top-down repair module first removes an instantiate knowledge source whose constraints of a premise are contradictory.[2] Suppose an instantiated knowledge source has a constraint that *domain_rule1* must have an antecedent *value(headache,yes)*. However, if the patient is known *not* to have a *headache*, the knowledge source contains a contradictory condition and is removed from the candidate set.

[2]MINERVA uses a maintenance system to maintain the dependency structure of facts and hypotheses.

If a premise of an instantiated knowledge source is not contradictory and fails to be satisfied, it may be the missing knowledge that is responsible for the failed explanation. Suppose there is no rule that has the desired symptom in the antecedent and the desired disease in the consequent; this means the knowledge base is missing the domain rule. The repair module identifies that a domain rule is missing and induces the domain rule which has the desired disease in the consequent and has the symptom in the antecedent, using the case library. Moreover, the repair module produces new domain rules which have more specific antecedents and also conclude the disease. If this domain rule generates a coherent explanation and its belief value calculated by Pearl's method [Pearl, 1986] over a library of 112 solved cases exceeds the threshold, it will be added to the knowledge base.

Experimental Results

We have completed two experiments using the knowledge refinement method described in this paper. The first experiment tests the performance of the failure detection program. The second experiment tests the repair module which generates new domain knowledge based on the failure, and also tests the diagnostic accuracy of the MINERVA expert system shell [Park *et al.*, 1989] after learning has taken place. We used a collection of 112 solved medical cases that were obtained from records at Stanford Medical Hospital.

The synthetic agent method [Wilkins, 1988a] was used in experiments: MINERVA with complete domain knowledge is used as a synthetic agent. We created ten incomplete knowledge bases. An incomplete knowledge base is missing all the domain rules which conclude a

	Incomplete KB	Refined KB	
Disease	Perf.1	Perf.2	Perf.3
Bacterial Meningitis	50	59	61
Brain Abscess	54	54	57
Cluster Headache	55	58	58
Fungal Meningitis	53	53	53
Migraine	57	57	58
Myco-TB Meningitis	57	61	61
Primary Brain Tumor	60	60	77
Subarach Hemorrhage	53	59	60
Tension Headache	53	53	60
Viral Meningitis	60	62	67
All diseases		64	75

Table 2: The performance of MINERVA after failure driven learning and context analysis. Each experiment is done over a library of 112 solved medical cases.

specific disease (see Table 1). In Table 1, *learning opportunity* means that a missing rule is a part of the synthetic agent's explanation. *Multiple masking* represents a case where multiple explanations mask an explanation failure. When the failure detection method without context analysis is applied, it can detect 66 out of 134 missing domain rule applications (49.2% accuracy). When the failure detection method with context analysis is used, it can detect 97 out of 134 missing domain rule applications (72.4%accuracy). Hence, the proposed method increased the number of learning opportunities by 47%.

In Table 2, under performance 1 column, the performance of MINERVA with missing knowledge is reported. Each experiment is done over a library of 112 solved cases with an incomplete knowledge base that is missing all the rules that conclude one designated disease. Under performance 2 column, the improved performance after failure driven learning without context analysis is presented. Each incomplete knowledge base is refined by the failure driven learning method. And each improved knowledge base is used to diagnose the 112 case library. Under performance 3 column, further improved performance after failure driven learning and context analysis is reported. After the system learns new knowledge for each disease, it adds all the knowledge to the existing knowledge base. The results of experiments with these new knowledge bases are shown in the last rows of performance 2 and performance 3 columns.

Conclusion

We have presented an effective and efficient knowledge acquisition method that employs failure driven learning. The failure detection process has been demonstrated to pinpoint the context failure as well as the expectation failure. It enables the learning program to avoid masking of learning opportunities due to the existence of multiple explanations, thereby improving the performance of the knowledge acquisition process. Our failure repair approach generates explanations and constraints using a bottom-up search followed by a top-down instantiation. A subject for future work is to develop a method which can reduce the number of redundant context changes by building a more robust goal transition model.

Acknowledgments

We would especially like to thank Marianne Winslett for providing invaluable comments on this paper. Chip Mayse also provided helpful comments.

We would like to express our deep gratitude to Ziad Najem, Lawrence Chachere, Kok-Wah Tan for their major role in the design and implementation of the MINERVA shell. This research was supported by ONR grant N00014-88K0124.

References

[Anderson, 1987] J.R. Anderson. Causal analysis and inductive learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, Urvine,CA, 1987.

[Clancey, 1987] W.J. Clancey. Acquiring, representing, and evaluating a competence model of diagnostic strategy. In *Contributions to the Nature of Expertise.* Lawerce Erlbaum Press, 1987.

[Dietterich and Buchanan, 1981] T.G. Dietterich and B.G. Buchanan. The role of the critic in learning systems. Technical Report STAN-CS-81-891, Stanford University,CA., 1981.

[Kodratoff and Tecuci, 1987] Y. Kodratoff and T. Tecuci. What is an explanation in disciple? In *Proceedings of the Fourth International Machine Learning Workshop*, pages 160–166, Irvine,CA, 1987.

[Mitchell *et al.*, 1985] T.M. Mitchell, S. Mahadevan, and L.I. Steinberg. Leap: a learning apprentice for vlsi design. In *Proceedings of the National Conference on AI*, pages 573–580, Los Angeles,CA, August 1985.

[Park *et al.*, 1989] Y.T. Park, K.W. Tan, and David Wilkins. *MINERVA: A Knowledge Based System with Declarative Representation and Flexible Control.* Working Paper UIUC-KBS-89-01, Dept. of Computer Science, University of Illinois, Urbana, IL, 1989.

[Pearl, 1986] J. Pearl. On evidential reasoning in a hierarchy of hypotheses. *Artificial Intelligence*, 28:9–15, 1986.

[Wilkins, 1988a] D.C. Wilkins. Apprenticeship learning techniques for knowledge based systems. Technical Report STAN-CS-88-1242, Stanford University,CA., 1988.

[Wilkins, 1988b] D.C. Wilkins. Knowledge base refinement using apprenticeship learning techniques. In *Proceedings of the National Conference on AI*, pages 646–651, St. Paul, MN, August 1988.