Knowledge Base Refinement Using Apprenticeship Learning Techniques

David C. Wilkins
Department of Computer Science
University of Illinois
Urbana, IL 61801

Abstract

This paper describes how apprenticeship learning techniques can be used to refine the knowledge base of an expert system for heuristic classification problems. The described method is an alternative to the long-standing practice of creating such knowledge bases via induction from examples. The form of apprenticeship learning discussed in this paper is a form of learning by watching, in which learning occurs by completing failed explanations of human problem-solving actions. An apprenticeship is the most powerful method that human experts use to refine their expertise in knowledge-intensive domains such as medicine; this motivates giving such capabilities to an expert system. A major accomplishment in this work is showing how an explicit representation of the strategy knowledge to solve a general problem class, such as diagnosis, can provide a basis for learning the knowledge that is specific to a particular domain, such as medicine.

1 Introduction

Traditional methods for semi-automatic refinement of the knowledge base of an expert system for heuristic classification problems [Clancey, 1985] have centered around induction over a case library of examples. Well-known sys tems that demonstrate this approach include ID3 [Quinlan, 1983], INDUCE [Michalski, 1983], SEEK [Politakis and Weiss, 1984; Ginsberg et al., 1985], and RL [Fu and Buchanan, 1985]. Over the last five years, we have been investigating a different approach to knowledge base refinement, called apprenticeship learning. This paper provides an overview of how the ODYSSEUS apprenticeship program improves an expert system by watching an expert.[1]

In induction from examples, a training instance consists of an unordered set of feature-value pairs for an entire diagnostic session and the correct diagnosis. In contrast, a training instance in apprenticeship learning is a single feature-value pair given within the context of a problem-solving session. This training instance is hence more fine-grained, can exploit the information implicit in the order in which the diagnostician collects information, and allows obtaining many training instances from a single diagnostic session. Our apprenticeship learning program attempts to construct an explanation of each training instance; an explanation failure occurs if none is found. The apprenticeship program then conjectures and tests modifications to the knowledge base that allow an explanation to be constructed. If an acceptable modification is found, the knowledge base is altered accordingly. This is a form of learning by completing failed explanations.

Apprenticeship learning involves the construction of explanations, but is different from explanation based learning as formulated in EBG [Mitchell et al., 1986] and EBL [DeJong, 1986]; it is also different from explanation based learning in LEAP [Mitchell et al., 1985], even though LEAP also focuses on the problem of improving a knowledge-based expert system. In EBG, EBL, and LEAP, the domain theory is capable of explaining a training instance and learning occurs by generalizing an explanation of the training instance. In contrast, in our apprenticeship research, a learning opportunity occurs when the domain theory, which is the domain knowledge base, is incapable of producing an explanation of a training instance. The domain theory is incomplete or erroneous, and all learning occurs by making an improvement to this domain theory.

2 Heracles Expert System Shell

ODYSSEUS is designed to improve any knowledge base crafted for HERACLES, an expert system shell that was created by removing the medical domain knowledge from the NEOMYCIN expert system [Clancey, 1984]. HERACLES uses a problem-solving method called *heuristic classification*, which is the process of selecting a solution out of a pre-enumerated solution set, using heuristic techniques [Clancey, 1985]. Our experiments used the NEOMYCIN medical knowledge base for diagnosis of neurological disorders. In a HERACLES-based system, there are three types of knowledge: domain knowledge, problem state knowledge, and strategy knowledge.[2]

Domain knowledge consists of Mycin-like rules and simple frame knowledge [Buchanan and Shortliffe, 1984]. An example of rule knowledge is `finding(photophobia, yes)` ⟶ `conclude(migraine-headache yes .5)`, meaning 'if the patient has photophobia, then conclude the patient has a migraine headache with a certainty factor of .5'. A typical example of frame knowledge is `subsumed-by(viral-meningitis meningitis)`, meaning 'hypothesis viral meningitis is subsumed by the hypothesis meningitis'.

Problem state knowledge is knowledge generated while running the expert system. For example, `rule-applied-(rule163)` says that Rule 163 has been applied during this

[1] ODYSSEUS can also improve an expert system by watching the expert system solve problems. This is another important form of apprenticeship learning, which is usually referred to as learning by doing, but is beyond the scope of this paper. The reader interested in further details is referred to [Wilkins, 1987].

[2] In this paper, the term *meta-level knowledge* refers to strategy knowledge; and the term *object-level knowledge* refers to domain and problem state knowledge.

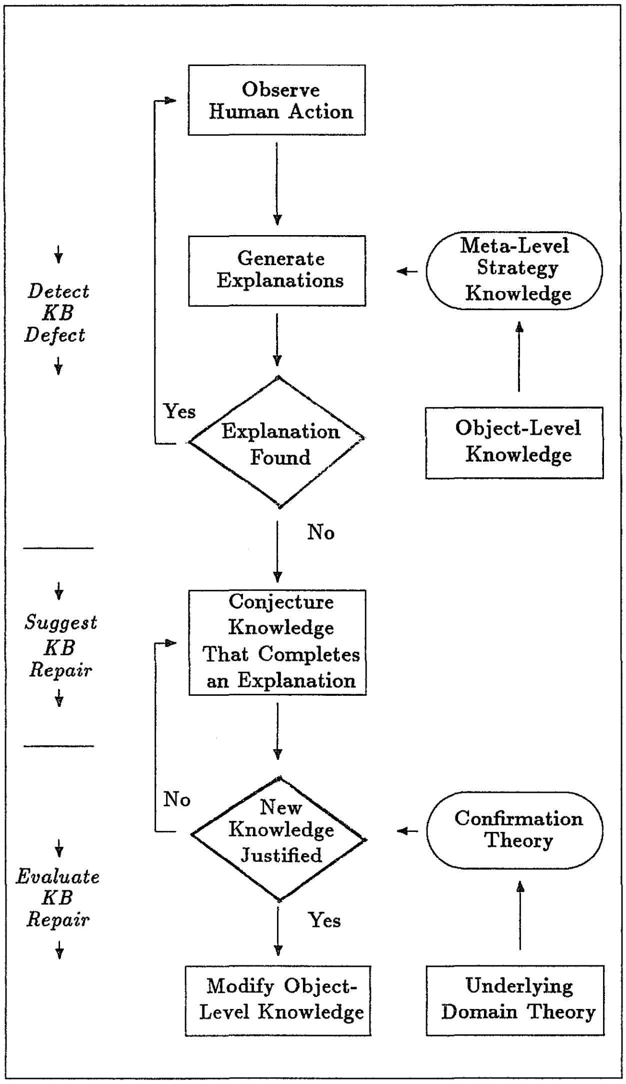


Figure 1: Overview of ODYSSEUS' method in a learning by watching apprentice situation. This paper describes techniques that permit automation of each of the three stages of learning shown on the left edge of the figure. An explanation is a proof that shows how the expert's action achieves a problem-solving goal.

consultation. Another example is `differential(migraine-headache tension-headache)`, which says that the expert system's active hypotheses are migraine headache and tension headache.

Strategy knowledge is contained in the HERACLES shell. The strategy knowledge approximates a cognitive model of heuristic classification problem solving. The different problem-solving strategies that can be employed during problem solving are explicitly represented. This facilitates using the model to follow the the line-of-reasoning of a human problem solver. The strategy knowledge determines what domain knowledge is relevant at any given time, and what additional information is needed to solve a particular problem case.

The strategy knowledge needs to access the domain and problem state knowledge. To achieve this, the domain and problem state knowledge is represented as tuples. Even rules are translated into tuples. For example, if Rule 160 is `finding(diplopia yes)` ∧ `finding(aphasia yes)` ⟶ `conclude(hemorrhage yes .5)`, it would be translated into the following four tuples: `evidence.for(diplopia hemorrhage rule160 .5)`, `evidence.for(aphasia hemorrhage rule160 .5)`, `antecedent(diplopia rule160)`, `antecedent(aphasia, rule160)`. Strategy metarules are quantified over the tuples. Figure 3 presents four strategy metarules in Horn clause form; the tuples in the body of the clause quantify over the domain and problem state knowledge. The rightmost metarule in Figure 3 encodes the strategy to find out about a symptom by finding out about a symptom that subsumes it. The metarule applies when the goal is to find out symptom P1, and there is a symptom P2 that is subsumed by P1, and P2 takes boolean values, and it is currently unknown, and P2 should be asked about instead of being derived from first principles. This is one of eight strategies in HERACLES for finding out the value of a symptom; this particular strategy of asking a more general question has the advantage of cognitive economy: a 'no' answer provides the answer to a potentially large number of questions, including the subsumed question.

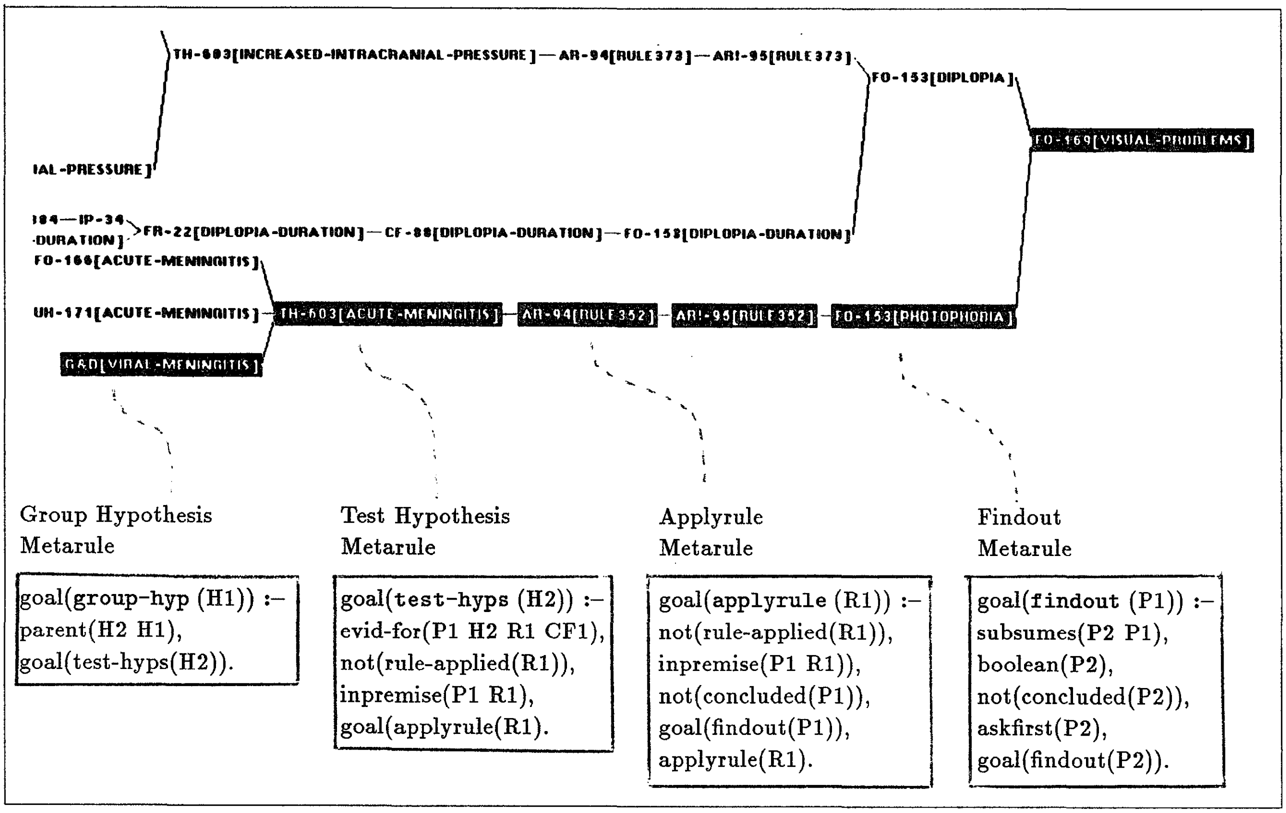
Patient's Complaint and Volunteered Information:
1. Alice Ecila, a 41 year old black female.
2. Chief complaint is a headache.

Physician's Data Requests:
3. Headache duration?
   focus=tension headache. 7 days.
4. Headache episodic?
   focus=tension headache. No.
5. Headache severity?
   focus=tension headache. 4 on 0-4 scale.
6. Visual problems?
   focus=subarachnoid hemorrhage. Yes.
7. Double vision?
   focus=subarachnoid hemorrhage, tumor. Yes.
8. Temperature?
   focus=infectious process. 98.7 Fahrenheit.

......

Physician's Final Diagnosis:
25. Migraine Headache.

Figure 2: An example of what the ODYSSEUS apprentice learner sees. The data requests in this problem-solving protocol were made by John Sotos, M.D. The physician also provides information on the focus of the data requests. The answers to the data requests were obtained from an actual patient file from the Stanford University Hospital, extracted by Edward Herskovits, M.D.



```
goal(group-hyp (H1)) :-
parent(H2 H1),
goal(test-hyps(H2)).
```

```
goal(test-hyps (H2)) :-
evid-for(P1 H2 R1 CF1),
not(rule-applied(R1)),
inpremise(P1 R1),
goal(applyrule(R1).
```

```
goal(applyrule (R1)) :-
not(rule-applied(R1)),
inpremise(P1 R1)),
not(concluded(P1)),
goal(findout(P1)),
applyrule(R1).
```

```
goal(findout (P1)) :-
subsumes(P2 P1),
boolean(P2),
not(concluded(P2)),
askfirst(P2),
goal(findout(P2)).
```

Figure 3: Learning by completing failed explanations. Each path in the graph is a failed explanation proof of attempts to connect a findout question about visual problems to a high-level problem solving goal; the nodes in the graph are metarules. The highlighted path is currently being examined during the second stage of learning, in which Odysseus tries to add knowledge to the domain knowledge base to complete the failed highlighted explanation. Four of the metarules in this highlighted path are illustrated in Horn clause form.

3 Odysseus' Apprenticeship Learning Method

The solution approach of the ODYSSEUS apprenticeship program in a learning by watching scenario is illustrated in Figure 1. As Figure 1 shows, the learning process involves three distinct steps: detect knowledge base (KB) deficiency, suggest KB repair, and evaluate KB repair. This section defines the concept of an explanation and then describes the three learning steps.

The main observable problem-solving activity in a diagnostic session is finding out features-values of the artifact to be diagnosed; we refer to this activity as asking *findout questions*. An *explanation* in ODYSSEUS is a proof that demonstrates how an expert's findout question is a logical consequence of the current problem state, the domain and strategy knowledge, and one of the current high-level strategy goals. An explanation is created by backchaining the meta-level strategy metarules; Figure 3 provides examples of these metarules represented in Horn clause form. The backchaining starts with the findout metarule, and continues until a metarule is reached whose head represents a high-level problem-solving goal. To backchain a metarule requires unification of the body of the Horn clause with domain and problem state knowledge. Examples of high-level goals are to test a hypothesis, to differentiate between several plausible hypotheses, to ask a clarifying question, and to ask a general question.

The first stage of learning involves the detection of a knowledge base deficiency. An expert's problem solving is observed and explanations are constructed for each of the observed problem-solving actions. An example will be used to illustrate our description of the three stages of learning, based on the NEOMYCIN knowledge base for diagnosing neurology problems. The input to ODYSSEUS is the problem-solving behavior of a physician, John Sotos, as shown in Figure 2. In our terminology, Dr. Sotos asks findout questions and concludes with a final diagnosis. For each of his actions, ODYSSEUS generates one or more explanations of his behavior.

When ODYSSEUS observes the expert asking a findout question, such as asking if the patient has visual problems, it finds all explanations for this action. When none can be found, an explanation failure occurs. This failure suggests that there is a difference between the knowledge of the expert and the expert system and it provides a learning opportunity. The knowledge difference may lie in any of the three types of knowledge that we have described: strategy knowledge, domain knowledge, or problem state knowledge. Currently, ODYSSEUS assumes that the cause of the explanation failure is that the domain knowledge is deficient. In the current example, no explanation can be found for findout question number 7, asking about visual problems, and an explanation failure occurs.

The second step of apprenticeship learning is to conjecture a knowledge base repair. A confirmation theory (which will be described in the discussion of the third stage of learning) can judge whether an arbitrary tuple of domain knowledge is erroneous, independently from the other knowledge in the knowledge base. A preprocessing stage allows the problem of erroneous knowledge to be corrected before the three stages of apprenticeship learning commences. The preprocessing stage also removes unsuitable knowledge. Knowledge is unsuitable if it is correct in isolation, but does not interact well with other knowledge in the knowledge base due to sociopathic interactions [Wilkins and Buchanan, 1986]. Hence, when a KB deficiency is detected during apprenticeship learning, we assume the problem is missing knowledge.

The search for the missing knowledge begins with the single fault assumption.[3] Conceptually, the missing knowledge could be eventually identified by adding a random domain knowledge tuple to the knowledge base and seeing whether an explanation of the expert's findout request can be constructed. How can a promising piece of such knowledge be effectively found? Our approach is to apply backward chaining to the findout question metarule, trying to construct a proof that explains why it was asked. When the proof fails, it is because a tuple of domain or problem state knowledge needed for the proof is not in the knowledge base. If the proof fails because of problem state knowledge, we look for a different proof of the findout question. If the proof fails because of a missing piece of domain knowledge, we temporarily add this tuple to the domain knowledge base. If the proof then goes through, the temporary piece of knowledge is our conjecture of how to refine the knowledge base.

Figure 3 illustrates the set of failed explanations that ODYSSEUS examines in connection with the unexplained action `findout(visual problems)` — the right most node of the graph. Each path in the graph is a potential explanation and each node in a path is a strategy metarule. The failed explanation that ODYSSEUS is examining is highlighted, and the associated metarules are shown below the graph. For a metarule to be used in a proof, its variables must be instantiated with domain or problem state tuples that are present in the knowledge base. In this example, the `evidence.for` tuple is responsible for the highlighted chain not forming a proof. It forms an acceptable proof if the tuple `evidence.for(photophobia acute.meningitis $rule $cf)` or `evidence.for(diplopia acute.meningitis $rule $cf)` is added to the knowledge base. During this step that generates repairs, neither the form of the left hand side of the rule (e.g., number of conjuncts) or the strength is known. In the step to evaluate repairs, the exact form of the rule is produced in the process of evaluation of the worth of the tuple.

The task of the third step of apprenticeship learning is to evaluate the proposed repair. To do this, we use a confirmation theory containing a decision procedure for each type of domain knowledge that tells us whether a given tuple is acceptable. There are different types of tuples in HERACLES' language. We only implemented a confirmation theory for three of the thirty-three different types of tuples in HERACLES's language: `evidence.for, clarifying.question,` and `ask.general.question` tuples.

`Evidence.for` tuples were generated in the visual problems example. In order to confirm the first candidate tuple, ODYSSEUS uses an induction system that generates and evaluates rules that have photophobia in their premise and acute meningitis in their conclusion. A rule is found that passes the rule 'goodness' measures, and is automatically added to the object-level knowledge base. All the tuples that are associated with the rule are also added to the knowledge base. This completes our example.

The confirmation theory also validates frame-like knowledge. An example of how this is accomplished will be described for clarify question tuples, such as `clarify.questions(headache-duration headache)`. This tuple means that if the physician discovers that the patient has a headache, she should always ask how long the headache has lasted. The confirmation theory must determine whether headache-duration is a good clarifying question for the 'headache' symptom. To achieve this, ODYSSEUS first checks to see if the question to be clarified is related to many hypotheses (the ODYSSEUS explanation generator allows it to determine this), and then tests whether the clarifying question can potentially eliminate a high percentage of these hypotheses. If these two criteria are met, then the clarify questions tuple is accepted.

4 Experimental Results

Our knowledge acquisition experiments centered on improving the knowledge base of the NEOMYCIN expert system for diagnosing neurology problems. The knowledge base of NEOMYCIN was constructed manually over a seven year period and had never been tested on a library of test cases. The NEOMYCIN vocabulary includes sixty diseases; our physician, Dr. John Sotos, determined that the existing data request vocabulary of 350 manifestations only allowed diagnosis of ten of these diseases. Another physician, Dr. Edward Herskovits, constructed a case library of 115 cases for these ten diseases from actual patient cases from the Stanford Medical Hospital, to be used for testing ODYSSEUS. The validation set consisted of 112 of these cases. The most recent version of NEOMYCIN, version 2.3, initially diagnosed 31% of these cases correctly.

For use as a training set, problem-solving protocols were collected of Dr. Sotos solving two cases, consisting of approximately thirty questions each. ODYSSEUS discovered ten pieces of knowledge by watching these two cases being solved; eight of these were domain rule knowledge. These eight pieces of information were added to the NEOMYCIN knowledge base of 152 rules, along with two pieces of frame knowledge that classified two symptoms as 'general ques-

[3] The missing knowledge is conceptually a single fault, but because of the way the knowledge is encoded, we can learn more than one tuple when we learn rule knowledge. For ease of presentation, this feature is not shown in the following examples.

tions'; these are questions that should be asked of every patient.

The set of 112 cases was rerun, and NEOMYCIN solved 44% of the cases correctly, a 42% improvement in performance. The performance of NEOMYCIN before and after learning is shown in Tables 1 and 2. All of this new knowledge was judged by Dr. Sotos as plausible medical knowledge, except for a domain rule linking aphasia to brain abscess. Importantly, the new knowledge was judged by our physicians to be of much higher quality than when straight induction was used to expand the knowledge base, without the use of explanation based learning.

The expected diagnostic performance that would be obtained by randomly guessing diagnoses is 10%, and the performance expected by always choosing the most common disease is 18%. NEOMYCIN initially diagnosed 31% of the cases correctly, which is 3.44 standard deviations better than always selecting the disease that is *a priori* the most likely. On a student-t test, this is significant at a t=.001 level of significance. Thus we can conclude that NEOMYCIN's initial diagnostic performance is significantly better than guessing.

After the apprenticeship learning session, NEOMYCIN correctly diagnosed 44% of the cases. Compared to NEOMYCIN's original performance, the performance of NEOMYCIN after improvement by ODYSSEUS is 2.86 standard deviations better. On a student-t test, this is significant for $t = .006$. One would expect the improved NEOMYCIN to perform better than the original NEOMYCIN in better than 99 out of 100 sample sets.

It is important to note that the improvement occurred despite the physician only diagnosing one of the two cases correctly. The physician correctly diagnosed a cluster headache case and misdiagnosed a bacterial meningitis case. As is evident from examining Tables 1 and 2, the improvement was over a wide range of cases. And the accuracy of diagnosing bacterial meningitis cases actually decreased. These counterintuitive results confirm our hypothesis that the power of our learning method derives from following the line of reasoning of a physician on individual findout questions, and is not sensitive to the final diagnosis as is the case when learning via induction from examples.

Disease	*Number Cases*	*True Positives*	*False Positives*	*False Negatives*
Brain Abscess	7	0	0	7
Bacterial Meningitis	16	16	47	0
Viral Meningitis	11	4	5	7
Fungal Meningitis	8	0	0	8
TB Meningitis	4	1	0	3
Cluster Headache	10	0	0	10
Tension Headache	9	9	20	0
Migraine Headache	10	1	1	9
Brain Tumor	16	0	0	16
Subarachnoid Hem.	21	4	0	17
None	0	0	4	0
Totals	112	35	77	77

Table 1: Performance of NEOMYCIN before apprenticeship learning. There were 112 cases used in the validation set to test NEOMYCIN's performance. A misdiagnosis produces a false positive and a false negative.

Disease	*Number Cases*	*True Positives*	*False Positives*	*False Negatives*
Brain Abscess	7	1	2	6
Bacterial Meningitis	16	12	31	4
Viral Meningitis	11	4	5	7
Fungal Meningitis	8	0	1	8
TB Meningitis	4	1	0	3
Cluster Headache	10	6	0	4
Tension Headache	9	9	11	0
Migraine Headache	10	2	0	8
Brain Tumor	16	5	3	11
Subarachnoid Hem.	21	9	1	12
None	0	0	9	0
Totals	112	49	63	63

Table 2: Performance of NEOMYCIN after apprenticeship learning. This shows the results of NEOMYCIN after a learning by watching session using ODYSSEUS that involved watching a physician solve two medical cases.

5 Conclusions

Apprenticeship is the most effective means for human problem solvers to learn domain-specific problem-solving knowledge in knowledge-intensive domains. This observation provides motivation to give apprenticeship learning abilities to knowledge-based expert systems. The paradigmatic example of an apprenticeship period is medical training. Our research investigated apprenticeship in a medical domain.

The described research illustrates how an explicit representation of the strategy knowledge for a general problem class, such as diagnosis, provides a basis for learning the domain-level knowledge that is specific to a particular domain, such as medicine, in an apprenticeship setting. Our approach uses a given body of strategy knowledge that is assumed to be complete and correct, and the goal is to learn domain-specific knowledge. This contrasts with learning programs such as LEX and LP where the domain-specific knowledge (e.g., integration formulas) is completely given at the start, and the goal is to learn strategy knowledge (e.g., preconditions of operators) [Mitchell, et al., 1983]. Two sources of power of the ODYSSEUS approach are the method of completing failed explanations and the use of a confirmation theory to evaluate domain-knowledge changes.

Our approach is also in contrast to the traditional induction from examples method of refining a knowledge base for an expert system for heuristic classification problems. With respect to learning certain types of heuristic rule knowledge, induction over examples plays a significant role in our work. In these cases, an apprenticeship approach can be viewed as a new method of biasing selection of which knowledge is learned by induction.

An apprenticeship learning approach, such as described in this paper, is perhaps the best possible bias for automatic creation of large 'use-independent' knowledge bases for expert systems. We desire to create knowledge bases that will support the multifaceted dimensions of expertise exhibited by some human experts, dimensions such as diagnosis, design, teaching, learning, explanation, and critiquing the behavior of another expert.

6 Acknowledgments

Many people have greatly contributed to the evolution of the ideas presented in this paper. I would especially like to thank Bruce Buchanan, Bill Clancey, Tom Dietterich, Haym Hirsh, John Holland, John Laird, Pat Langley, Bob Lindsay, John McDermott, Ryszard Michalski, Roy Rada, Tom Mitchell, Paul Rosenbloom, Ted Shortliffe, Paul Scott, Devika Subramanian, Marianne Winslett, the members of the Grail learning group, and the Guidon tutoring group. Marianne Winslett provided invaluable comments on draft versions of this paper. Larry Rendell also provided helpful comments. This work would have not been possible without the help of physicians Eddy Herskovits, Kurt Kapsner, and John Sotos. This research was principally supported by NSF grant MCS-83-12148, and ONR grants N00014-79C-0302 and N00014-88K0124.

References

[Buchanan and Shortliffe, 1984] Buchanan, B. G. and Shortliffe, E. H. (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, Mass.: Addison-Wesley.

[Clancey, 1984] Clancey, W. J. (1984). NEOMYCIN: reconfiguring a rule-based system with application to teaching. In Clancey, W. J. and Shortliffe, E. H., editors, *Readings in Medical Artificial Intelligence*, chapter 15, pages 361–381, Reading, Mass.: Addison-Wesley.

[Clancey, 1985] Clancey, W. J. (1985). Heuristic classification. *Artificial Intelligence*, 27:289–350.

[DeJong, 1986] DeJong, G. (1986). An approach to learning from observation. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning, Volume II*, chapter 19, pages 571–590, Los Altos: Morgan Kaufmann.

[Fu and Buchanan, 1985] Fu, L. and Buchanan, B. G. (1985). *Inductive knowledge acquisition for rule based expert systems.* Knowledge Systems Laboratory Report KSL-85-42, Stanford University, Stanford, CA.

[Ginsberg et al., 1985] Ginsberg, A., Weiss, S., and Politakis, P. (1985). SEEK2: a generalized approach to automatic knowledge base refinement. In *Proceedings of the 1985 IJCAI*, pages 367–374, Los Angeles, CA.

[Michalski, 1983] Michalski, R. S. (1983). A theory and methodology of inductive inference. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, chapter 4, pages 83–134, Palo Alto: Tioga Press.

[Mitchell et al., 1983] Mitchell, T., Utgoff, P. E., and Banerji, R. S. (1983). Learning by experimentation: acquiring and refining problem-solving heuristics. In Michalski, T. M., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 163–190, Palo Alto: Tioga Press.

[Mitchell et al., 1986] Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. (1986). Explanation-based generalization: a unifying view. *Machine Learning*, 1(1):47–80.

[Mitchell et al., 1985] Mitchell, T. M., Mahadevan, S., and Steinberg, L. I. (1985). LEAP: a learning apprentice for VLSI design. In *Proceedings of the 1985 IJCAI*, pages 573–580, Los Angeles, CA.

[Politakis and Weiss, 1984] Politakis, P. and Weiss, S. M. (1984). Using empirical analysis to refine expert system knowledge bases. *Artificial Intelligence*, 22(1):23–48.

[Quinlan, 1983] Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning*, chapter 15, pages 463–482, Palo Alto: Tioga Press.

[Smith et al., 1985] Smith, R. G., Winston, H. A., Mitchell, T. M., and Buchanan, B. G. (1985). Representation and use of explicit justifications for knowledge base refinement. In *Proceedings of the 1985 IJCAI*, pages 673–680, Los Angeles, CA.

[Wilkins, 1987] Wilkins, D. C. (1987). *Apprenticeship Learning Techniques For Knowledge Based Systems.* PhD thesis, University of Michigan. Also, Knowledge Systems Lab Report KSL-88-14, Dept. of Computer Science, Stanford University, 1988, 153pp.

[Wilkins and Buchanan, 1986] Wilkins, D. C. and Buchanan, B. G. (1986). On debugging rule sets when reasoning under uncertainty. In *Proceedings of the 1986 National Conference on Artificial Intelligence*, pages 448–454, Philadelphia, PA.