Mixed-Initiative Development of Knowledge Bases

Tecuci G., Boicu M., Wright K., and Lee S.W.

Learning Agents Laboratory, Department of Computer Science, MSN 4A5, George Mason University, Fairfax, VA 22030
{tecuci, mboicu, kwright, swlee}@gmu.edu

Abstract

Ample experimental evidence shows that manual solutions to the problem of building knowledge bases are highly inefficient, while fully automated, machine learning based, approaches have not yet led to practical solutions. This makes this problem an ideal test bed for developing mixed-initiative methods. In this paper we present such a mixed-initiative approach where a subject matter expert teaches his or her expertise to a learning agent. We present several mixed-initiative methods used during the various phases of knowledge base development, such as, cooperative problem solving, rule learning, rule refinement and exception handling, discussing solutions to the control and communication issues that allow the realization of several types of synergism between the subject matter expert and the learning agent. We also provide experimental evidence of the feasibility of the proposed approach.

Introduction

The development of knowledge bases is one of the big challenges of Artificial Intelligence, the proposed solutions covering the full spectrum of approaches, from manual ones (where a human builds the knowledge base) to the fully automated ones (where a knowledge base is learned by a system).

An even more challenging version of this problem, addressed by DARPA's and AFOSR's High Performance Knowledge Base Program (Cohen et al., 1998), is the development of a knowledge base directly by a subject matter expert that has limited knowledge engineering experience and receives limited support from a knowledge engineer. We think that, as formulated, the solution to this problem can only come from a mixed-initiative approach where a subject matter expert cooperates with a learning agent to build the knowledge base.

From an abstract point of view, solving this problem means building a machine representation of the model of the real world that exists only in the mind of the domain expert. However, the domain expert is not a knowledge engineer and cannot be expected to be able to appropriately formalize his or her domain model. On the other hand, the learning agent does not know what is to be formalized and has to get this information from the expert. There are two main issues that need to be considered in any attempted solution: a control issue and a communication issue. The control issue deals with the distribution of tasks between the expert and the agent. There are some tasks that are more difficult for the expert than for the agent, and there are other tasks that are more difficult to be performed by the agent.

Representative examples of tasks that are significantly harder for the domain expert are the definition of general problem solving knowledge (that correctly characterizes specific examples), the verification of knowledge base consistency and reduction of any inconsistencies, and the reorganization of the knowledge base to improve the problem solving efficiency of the knowledge base.

Representative examples of tasks that are significantly harder for the learning agent are the problem of new terms, the credit/blame assignment problem, and the definition of the learner's representation language, background knowledge, input examples, and bias.

Therefore, a mixed-initiative approach to knowledge base development should be based on a control structure where each participant does what it can do best, and receives help from the other party.

The second issue is the communication one: how to define a protocol to exchange information between the expert and the agent in a form that one can easily create and the other can easily understand? For instance, it is clearly easier for the domain expert to understand a sentence in agent's language than to create the sentence. Therefore, when the agent needs some information from the expert, rather than asking the expert to provide it, the agent can formulate the question and possible answers and ask the expert to indicate the correct one. Or, when the agent cannot hypothesize the correct answer, it could ask the expert to only provide a hint and will use this hint to hypothesize possible answers.

For several years we have been working on developing a mixed-initiative approach to knowledge base development. In this approach, called Disciple, the subject matter expert teaches the agent how to perform domain-specific tasks in a way that resembles how the expert would teach an apprentice, by giving the agent examples and explanations as well as by supervising and correcting its behavior (Tecuci, 1998). From this interaction with the expert the agent builds and improves its knowledge base. The current version of the Disciple approach is implemented in the Disciple Learning Agent Shell (Disciple-LAS). The Disciple shell and methodology are presented in (Tecuci et al., 1999). Therefore, in this paper, we will only address the mixed-initiative aspects of Disciple.

The central idea of the Disciple approach is to facilitate the process of building the knowledge base through the use of the synergism at three different levels.

At the highest level, there is the synergism between the expert and the agent in solving a problem in cooperation. The agent has plausible reasoning capabilities that allow it to distinguish between the following types of problem solving situations:

- Situations where the agent is able to generate routine solutions (that are known to be correct).
- Situations where the agent can only attempt innovative solutions through the use of plausible reasoning. These solutions may or may not be correct and have to be checked by the domain expert.
- Situations that require creative solutions that are beyond agent's reasoning capabilities. These solutions need to be provided by the domain expert.

The agent's ability to recognize the current situation guides the interactions with the domain expert and leads to a cooperative problem solving process where the agent solves the more routine parts of the problem and the expert solves the more creative ones. In the process, the agent learns from the expert. The problem solving situations that were innovative for the agent become routine, and those that were creative become first innovative and later routine ones.

At the second level there is the synergism between expert's teaching of the agent and the agent's learning from the expert (Tecuci and Kodratoff, 1995). For instance, the expert may select a representative example to teach the agent how to solve a new problem (in a creative problem solving situation), may provide explanations of the solution, and may answer agent's questions. The agent, on the other hand, will learn a general problem solving rule that would be difficult to be defined by the expert, and will consistently integrate it into its knowledge base.

Finally, at the deepest level, there is the synergism between the different learning methods employed by the agent (Michalski and Tecuci, 1994). By integrating complementary learning methods (such as inductive learning from examples, explanation-based learning, learning by analogy, learning by experimentation) in a dynamic way, the Disciple agent is able to learn from the human expert in situations in which no single strategy learning method would be sufficient.

The Disciple-LAS mixed-initiative approach has been applied in the HPKB program to develop the knowledge base of a workaround agent. This agent has to determine the best plan of actions for bypassing or reconstituting a damage to an infrastructure, such as a damaged bridge or a cratered road. The plan has to indicate a minimal and expected time of completion, the resources needed, and the transportation capacity of the reconstructed bridge or road. We use this application to discuss the various mixed-initiative methods of Disciple. We first describe the structure of the knowledge base to be developed. Then we present the mixed-initiative methods used during cooperative problem solving, rule learning, rule refinement and exception handling. We provide some experimental results and we conclude the paper with a discussion of some of the weaknesses of our approach and of future research directions.

The Structure of the Knowledge Base

The knowledge base of Disciple-LAS consists of an ontology and a set of problem solving rules. The ontology defines the concepts from the application domain (Gruber, 1993). It consists of hierarchical descriptions of objects, features and tasks, all represented as frames, according to the OKBC knowledge model (Chaudhri et al., 1998).

The problem solving approach of the Disciple agent is task reduction, where a task to be accomplished by the agent is successively reduced to simpler tasks until the initial task is reduced to a set of elementary tasks that can be immediately performed. Therefore the problem solving rules from the agent's knowledge base are task reduction rules expressed in terms of the concepts from the ontology. These rules are learned by the agent from specific examples of task reductions provided by the domain expert and from the explanations of these reductions.

An example of task reduction has the following form:

TR: If the task to accomplish is T_1
then accomplish the tasks T_{11}, … , T_{1n}

A task may be reduced to one simpler task, or to a (partially ordered) set of tasks. An example of task reduction is presented in Figure 1. It states that in order to workaround the destroyed bridge at site100, one has to use a bridge equipment of type avlb-eq and to reduce the size of the gap.

```
IF the task to accomplish is
     Workaround-unmined-destroyed-bridge-with-fixed-bridge
          at-location    site100
          for-gap        site103
          by-unit        unit91010
THEN accomplish the task
     Use-fixed-bridge-with-gap-reduction-over-gap
          at-location    site100
          for-gap        site103
          by-unit        unit91010
          with-br-eq     avlb-eq
```

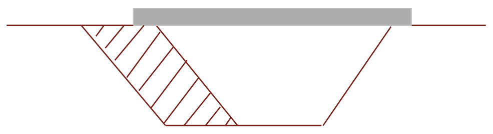


Figure 1: An example of task reduction.

An explanation is an expression of objects and features that indicates why a task reduction is correct (or why it is incorrect). It corresponds to the justification given by a domain expert to a specific task reduction:

the task reduction TR is correct because E

For example, an explanation of the task reduction from Figure 1 is the one from Figure 2. The first two explanation pieces justify why one needs to use gap reduction. The length of the site103 gap is 25 m and the avlb-eq allows building a bridge of type avlb70 that can only span gaps up to 17 m. Therefore, the gap is too large to install avlb70 directly. However, any gap that is not longer than 26 m can be reduced to a 17 m gap on which one can install an avlb70 bridge. The next two explanation pieces show that an avlb70 bridge is strong enough to sustain the vehicles of unit91010. Indeed, the maximum load class of the wheeled vehicles of unit91010 is 20 tones and avlb70 can sustain vehicles with a load of up to 70 tones. Similarly, the avlb70 bridge can sustain the tracked vehicles of unit91010.

```
site103  with-length    25m,
avlb-eq  can-build  avlb70  max-gap    17m  <  25m

site103  with-length    25m,
avlb-eq  can-build  avlb70  max-reducible-gap  26m  =  25m

unit91010  max-wheeled-mlc   20t,
avlb-eq  can-build  avlb70  mlc-rating  70t   =  20t

unit91010  max-tracked-mlc   40t,
avlb-eq  can-build  avlb70  mlc-rating  70t   =  40t
```

Figure 2: Explanation of the task reduction in Figure 1.

The rule learned from a task reduction TR and an explanation E has the following form:

If the task to accomplish is T_{1g} and
E_g holds ; plausible upper bound condition
E_s holds ; plausible lower bound condition
then accomplish the tasks $T_{11g}, \dots , T_{1ng}$

This is a partially learned IF-THEN rule that has two conditions, a plausible upper bound (PUB) condition E_g which, as an approximation, is more general than the exact condition E_h, and a plausible lower bound (PLB) condition E_s which, as an approximation, is less general than E_h. During rule refinement, the two conditions will both converge toward the single condition E_h. We will refer to such a rule as a plausible version space rule, or PVS rule.

In addition to the rule's condition that needs to hold in order for the rule to be applicable, the rule may also have several "except-when" conditions that should not hold, in order for the rule to be applicable. An except-when condition is a generalization of the explanation of why a negative example of a rule does not represent a correct task reduction. Finally, the rule may also have "except-for" conditions (that specify negative exceptions of the rule) and "for" conditions (that specify positive exceptions).

The plausible version space rule learned from the example in Figure 1 and the explanation in Figure 2 is shown in Figure 3.

Cooperative Problem Solving

The main control of the process of building the knowledge base is provided by the Cooperative Problem Solver, as indicated in Figure 4.

IF ***the task to accomplish is***

Workaround-unmined-destroyed-bridge-with-fixed-bridge
at-location ?o1
for-gap ?o2
by-unit ?o3

Plausible upper bound	***Plausible lower bound***
?o1 is bridge	?o1 is site100
?o2 is cross-section has-length ?n4	?o2 is site103 has-length ?n4
?o3 is military-unit max-tracked-mlc ?n3 max-wheeled-mlc ?n2	?o3 is unit91010 max-tracked-mlc ?n3 max-wheeled-mlc ?n2
?o4 is avlb-eq can-build ?o5 max-reducible-gap ?n5 max-gap ?n6	?o4 is avlb-eq can-build ?o5 max-reducible-gap ?n5 max-gap ?n6
?o5 is avlb70 mlc-rating ?n1	?o5 is avlb70 mlc-rating ?n1
?n1 is-in [0.0 150.0]	?n1 is-in [70 70]
?n2 is-in [0.0 150.0] = ?n1	?n2 is-in [25 25] = ?n1
?n3 is-in [0.0 150.0] = ?n1	?n3 is-in [63 63] = ?n1
?n4 is-in [0 +infinite)	?n4 is-in [25.0 25.0]
?n5 is-in [0.0 100.0] = ?n4	?n5 is-in [26 26] = ?n4
?n6 is-in [0 1000] < ?n4	?n6 is-in [17 17] < ?n4

THEN ***accomplish the task***
?t1 Use-fix-br-with-gap-reduction-over-gap
at-location ?o1
for-gap ?o2
by-unit ?o3
with-br-eq ?o4

Figure 3: An example of a task reduction rule.

The expert formulates a problem to solve (or, in agent terminology, a task to accomplish) and Disciple tries to reduce it by applying the task reduction rules. A plausible version space rule is used to generate task reductions with different degrees of plausibility, depending on which of its conditions are satisfied. If the PLB condition is satisfied, then the reduction is very likely to be correct. If PLB is not satisfied, but PUB is satisfied, then the solution is considered only plausible. The same rule could also be applied for tasks that are considered similar to the task reduced by it. In such a case the reductions are considered even less plausible.

The expert has to analyze agent's solution, deciding to accept it or to reject it. In both cases the rule refiner is invoked to either generalize or to specialize the rule that generated the solution. The rule refinement process is discussed in more details in a subsequent section.

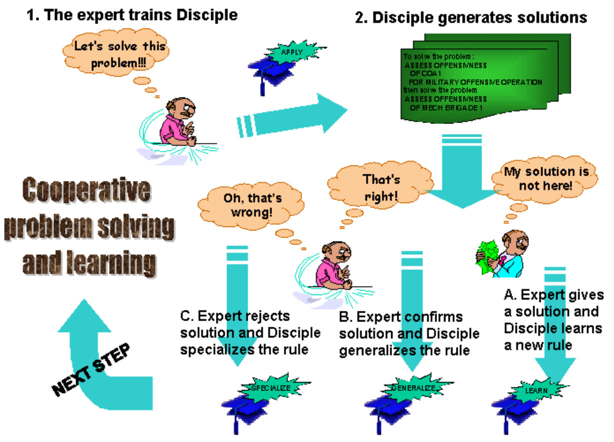

1. The expert trains Disciple
Let's solve this problem!!!
APPLY
2. Disciple generates solutions
To solve the problem : ASSESS OFFENSIVNESS OF COA 1 FOR MILITARY OFFENSIVE OPERATION then solve the problem ASSESS OFFENSIVNESS OF MECH BRIGADE 1
Cooperative problem solving and learning
Oh, that's wrong!
That's right!
My solution is not here!
C. Expert rejects solution and Disciple specializes the rule
B. Expert confirms solution and Disciple generalizes the rule
A. Expert gives a solution and Disciple learns a new rule
NEXT STEP
SPECIALIZE
GENERALIZE
LEARN

Figure 4: Cooperative problem solving and learning.

If the agent was not able to propose any solution, then the expert has to provide one. In this case the rule learner is invoked to learn a new plausible version space rule. The rule learning process is described in the next section.

From a mixed-initiative point of view, the important aspect here is that the expert solves the credit and the blame assignment problem, and the agent updates the rule to correspondingly assign credit or blame to it.

The problem solver can also be invoked in an autonomous mode. In this case it receives an initial problem and attempts to solve it completely, through successive reductions. If the final solution is correct, then all the applied rules receive the corresponding credit. However, if the expert disagrees with the final solution, then he or she has to investigate each individual reduction that led to the final solution, and has to identify the reduction to blame.

The workaround agent receives as input the description of a damage and of a military unit that has to workaround it, and generates a workaround plan for that unit. For instance, the input might be the destroyed bridge at site100 (see Figure 1) and unit91010 has to workaround it. The best plan found by the workaround agent is presented in Figure 5. It consists of installing an avlb bridge over the river gap. It is estimated that this will take a minimum of 11h:4m:58s, the expected duration being 14h:25m:56s. Unit91010 will need the help of unit202 that has an avlb equipment and of unit201 that has a bulldozer. After the bridge is installed, it will allow a traffic of 135.13 vehicles/h. The plan consists of 12 elementary actions. Unit91010 has to obtain operational control of unit202 that has the avlb equipment.

Workaround summary:

Initial task:
Workaround-damage
    for-damage damage200
    by-interdicted-unit unit91010

Engineering action: install avlb
    min-duration 11h:4m:58s
    expected-duration 14h:25m:56s
    resources required: (avlb-unit202 bulldozer-unit201)
    link capacity after reconstruction: 135.13 vehicles/hr

Detailed plan:

S1 Obtain-operational-control-from-corps

of-unit	unit202
by-unit	unit91010
min-duration	4h:0m:0s
expected-duration	6h:0m:0s
time-constraints:	none

S2 Move-unit

for-unit	unit202
from-location	site0
to-location	site100
min-duration	1h:8m:14s
expected-duration	1h:8m:14s
time-constraints:	after S1

S3 Report-obtained-equipment

for-eq-set	avlb-unit202
min-duration	0s
expected-duration	0s
time-constraints:	after S2

S4 Obtain-operational-control-from-corps

of-unit	unit201
by-unit	unit91010
min-duration	4h:0m:0s
expected-duration	6h:0m:0s
time-constraints:	none

S5 Move-unit

for-unit	unit201
from-location	site0
to-location	site100
min-duration	1h:8m:14s
expected-duration	1h:8m:14s
time-constraints:	after S4

S6 Report-obtained-equipment

for-eq-set	bulldozer-unit201
min-duration	0s
expected-duration	0s
time-constraints:	after S5

S7 Narrow-gap-by-filling-with-bank

for-gap	site103
for-br-design	avlb70
min-duration	5h:19m:44s
expected-duration	6h:7m:42s
resources-required	bulldozer-unit201
time-constraints:	after S6

S8 Emplace-avlb

for-br-design	avlb70
min-duration	5m:0s
expected-duration	10m:0s
resources-required	avlb-unit202
time-constraints:	after S3, S7

S9 Report-emplaced-fixed-bridge

for-mil-bridge	fixed-military-bridge-eq
min-duration	0s
expected-duration	0s
time-constraints:	after S8

S10 Move-equipment-over-unstabilized-mil-bridge

for-eq-set	bulldozer-unit201
for-br-design	avlb70
min-duration	2m:0s
expected-duration	10m:0s
resources-required	avlb-unit202
time-constraints:	after S9

S11 Minor-bank-preparation

of-bank	site105
min-duration	30m:0s
expected-duration	50m:0s
resources-required	bulldozer-unit201
time-constraints:	after S10

S12 Restore-traffic-link

for-unit	unit91010
for-link	avlb70
link-capacity	135.13 vehicles/h
min-duration	0s
expected-duration	0s
time-constraints:	after S11

Figure 5: A generated workaround plan.

Then this unit has to come to the site of the destroyed bridge. Also, unit91010 has to obtain operational control of unit201 that has a bulldozer. This unit will have to move to the site of the destroyed bridge and then to narrow the river gap from 25m to 17m. These actions can take place in parallel with the actions of bringing unit202 to the bridge site. Then the avlb bridge is emplaced, the bulldozer moves over the bridge and clears the other side of the river to restore the traffic. This plan was generated by successively reducing the initial "Workaround-damage" task to simpler subtasks.

Rule Learning

Each time the expert will provide the reduction of the current task, the agent will attempt to learn a task reduction rule which is a generalization of this example of problem solving episode, as shown in Figure 6.

To learn the rule, the agent will first try to find an explanation of why the example reduction is correct. Then the example and the explanation are automatically generalized to a plausible version space rule.

The expert may provide an explanation to the agent of why the example is correct. However, we have experimentally found that it is difficult for the expert to manually define such explanations. The reasons are the following ones:

- The expert would need to provide an explanation that the agent can "understand". That is, the explanation has to take into account the knowledge level of the agent in the same way in which one would give a different explanation of the same phenomenon to an elementary school student than to a high school student. This means that the expert would need to have a good understanding of agent's knowledge.

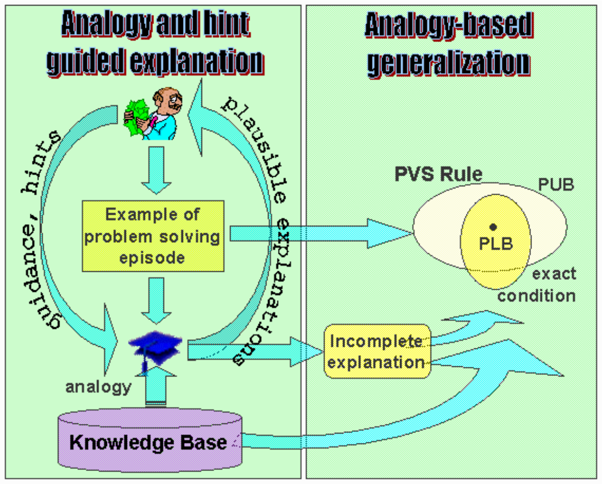
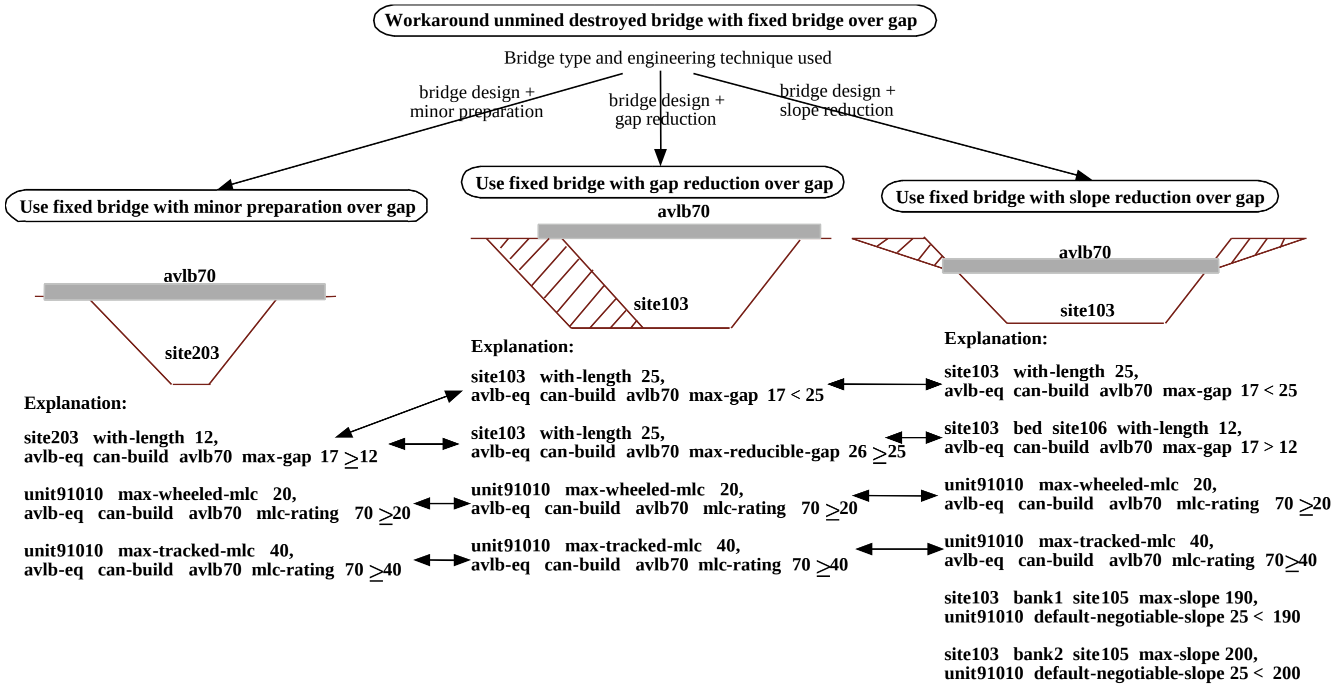


Figure 6: The rule learning method.

- The expert would need to be able to express the explanation in the language of the agent. That is, the expert would need to know not only the syntax of the correct sentences but also the names of the various concepts.

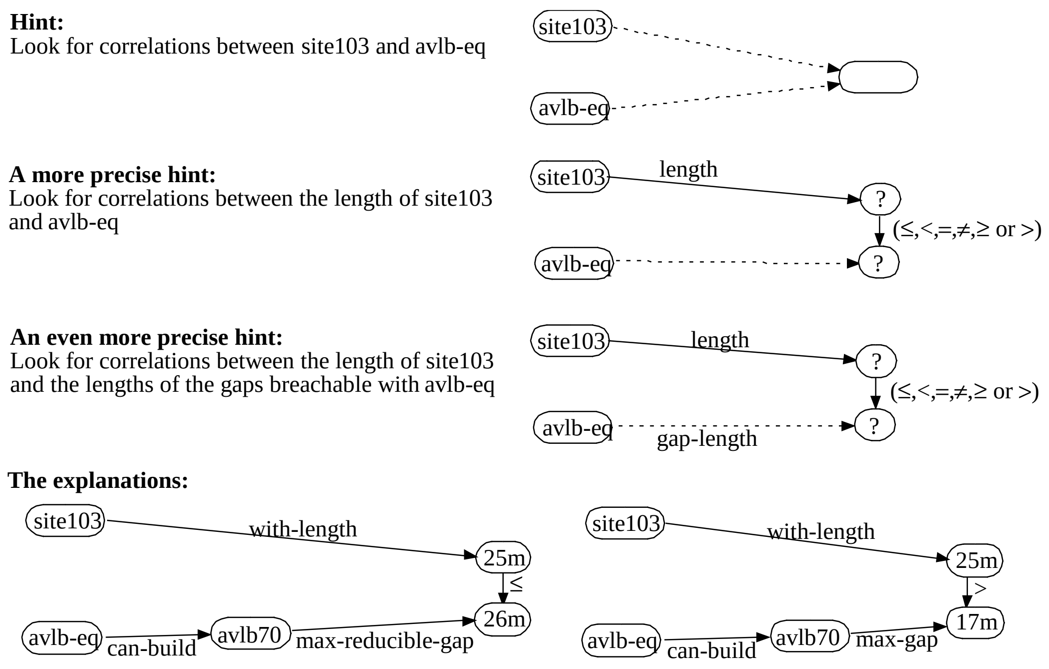
Because it is more difficult for the expert to provide explanations, we have adopted a different approach. The agent will attempt various strategies to propose plausible explanations from which the user will choose the correct ones. The strategies are based on an ordered set of heuristics for analogical reasoning which allow the agent to propose explanations ordered by their plausibility.

For instance, to understand why a task reduction is correct, the agent will consider the rules that reduce the



Workaround unmined destroyed bridge with fixed bridge over gap
Bridge type and engineering technique used
bridge design + minor preparation
bridge design + gap reduction
bridge design + slope reduction
Use fixed bridge with minor preparation over gap
Use fixed bridge with gap reduction over gap
Use fixed bridge with slope reduction over gap
avlb70
site203
avlb70
site103
avlb70
site103
Explanation:
site203 with-length 12,
avlb-eq can-build avlb70 max-gap 17 ≥12
unit91010 max-wheeled-mlc 20,
avlb-eq can-build avlb70 mlc-rating 70 ≥20
unit91010 max-tracked-mlc 40,
avlb-eq can-build avlb70 mlc-rating 70 ≥40
Explanation:
site103 with-length 25,
avlb-eq can-build avlb70 max-gap 17 < 25
site103 with-length 25,
avlb-eq can-build avlb70 max-reducible-gap 26 ≥25
unit91010 max-wheeled-mlc 20,
avlb-eq can-build avlb70 mlc-rating 70 ≥20
unit91010 max-tracked-mlc 40,
avlb-eq can-build avlb70 mlc-rating 70 ≥40
Explanation:
site103 with-length 25,
avlb-eq can-build avlb70 max-gap 17 < 25
site103 bed site106 with-length 12,
avlb-eq can-build avlb70 max-gap 17 > 12
unit91010 max-wheeled-mlc 20,
avlb-eq can-build avlb70 mlc-rating 70 ≥20
unit91010 max-tracked-mlc 40,
avlb-eq can-build avlb70 mlc-rating 70≥40
site103 bank1 site105 max-slope 190,
unit91010 default-negotiable-slope 25 < 190
site103 bank2 site105 max-slope 200,
unit91010 default-negotiable-slope 25 < 200

Figure 7: Generation of explanations by analogy with other reductions of the same task.



Hint:
Look for correlations between site103 and avlb-eq

A more precise hint:
Look for correlations between the length of site103 and avlb-eq

An even more precise hint:
Look for correlations between the length of site103 and the lengths of the gaps breachable with avlb-eq

The explanations:

Figure 8: Guiding the agent to generate explanations.

same task into different subtasks, and will use the explanations corresponding to these rules to propose similar explanations for the current reduction. This heuristic is based on the observation that the explanations of the alternative reductions of a task tend to have similar structures. The same factors are considered, but the relationships between them are different. Figure 7, for instance, shows three different reductions of the task to workaround a damaged bridge using a fixed military bridge. The leftmost reduction consists in installing the fixed bridge with minor preparations, the center reduction consists in using gap reduction, and the right most reduction consists in using slope reduction. In a particular situation, the decision of which of these reductions to perform depends upon the specific relationships between the dimensions of the bridge and the dimensions of the river gap. Below each reduction in Figure 7 there are the explanations corresponding to it. The similar explanations are connected by bi-directional arrows. As one can see, if the agent has already learned a rule corresponding to any of these reductions, then learning the rules corresponding to the other reductions is much simpler because the agent can propose explanations by analogy with those of the learned rule.

This above strategy works well when the agent has already learned rules similar with the rule being currently learned. In the situations when this is not true the agent has to acquire the explanations from the expert. However, even in such cases, the expert need not provide explanations, but only hints that may have various degrees of detail. Let us consider, for instance, the task reduction in Figure 1. The expert can give the agent a very general hint, such as, "Look for correlations between site103 (the river gap) and avlb-eq." This is expressed by simply pointing to the objects site103 and avlb-eq. A more specific hint would be "Look for correlations between the length of site103 and avlb-eq". An even more specific hint would be "Look for correlations between the length of site103 and the lengths of the gaps breachable with avlb-eq".

Such hints will guide the agent in looking for explanations that have a certain pattern, as indicated in Figure 8. Among the plausible explanations proposed by the agent will also be the correct explanations shown at the bottom of Figure 8. Notice that when matching the pattern corresponding to a hint to the explanation in the agent's ontology, the agent uses the generalization hierarchy of the features. For instance, "length" in the hint matches "with-length" in the explanations because "with-length" is a subclass of "length". Also, "gap-length" will matches both "max-reducible-gap" and "max-gap".

The goal of this process is to allow the expert to provide hints or incomplete explanations, which are easier to express, rather than detailed explanations.

Once the explanation is found, the example and the explanation are automatically generalized by Disciple into a quite complex plausible version space rule (see Figure 3). This generalization process takes into account the various constraints that different pieces of knowledge should satisfy. Let us consider the two explanations from the bottom of Figure 8. In these explanations, avlb70 appears as the value of the feature can-build, and has the features max-reducible-gap and max-gap. Therefore, the generalization of avlb70 should be a concept that is at most as general as the intersection between the range of the feature can-build, the domain of the feature max-reducible-gap and the domain of the feature max-gap. It would be very difficult for a domain expert to resolve these kind of constraints, while for the agent this is very easy.

Rule Refinement

The explanation based on which a rule is learned is generally incomplete. This may be due to the fact that agent's knowledge is incomplete and therefore the explanations proposed by the agent will be incomplete. This may also be due to the fact that the explanations given by human experts are generally incomplete, important aspects being left out because they are assumed to be known to the other party.

Because the explanation of the example is incomplete, the rule learned by the Disciple agent will also be incomplete and will need to be refined. Figure 9 represents the basic approach to rule refinement. The agent uses its plausible upper bound to generate an example and asks the user to characterize it as correct or as incorrect. If the example is accepted, then it is a positive example of the rule and the plausible lower bound condition is automatically generalized to cover it.

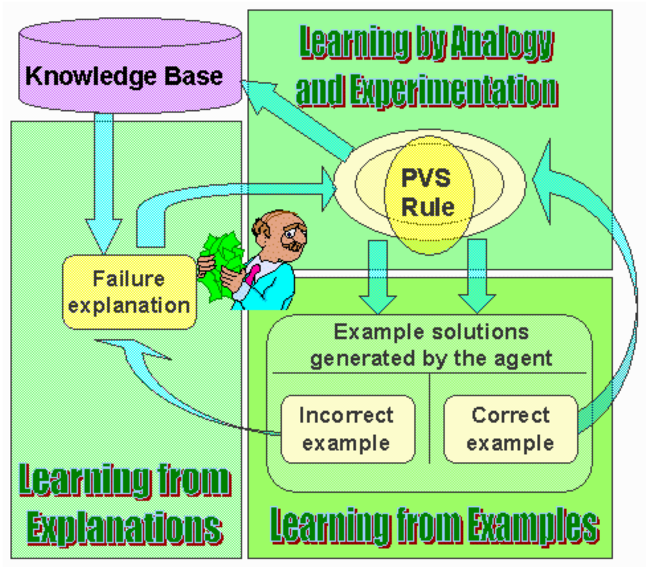

Knowledge Base
Learning by Analogy and Experimentation
PVS Rule
Failure explanation
Example solutions generated by the agent
Incorrect example
Correct example
Learning from Explanations
Learning from Examples

Figure 9: The rule refinement method.

However, if the example is rejected, then it is a negative example of the rule. In this case the agent will try to find an explanation of why the example problem solving episode is wrong, by applying the strategies presented in the previous section. If an explanation is found, then the rule is specialized with a generalization of the explanation. The important aspect here is that the expert is helping the agent to understand what is wrong with the negative example and the agent uses this explanation to automatically refine the rule, rather than having the expert or the knowledge engineer to manually modify the rule.

Another important aspect is that a user always analyzes specific examples, and is the agent that manipulates the rule to make it consistent with these examples.

During rule refinement, the agent may use various strategies to select the examples to be analyzed by the expert. On the one hand one would like to learn a rule from as few examples as possible, to minimize the interaction with the expert. On the other hand, however, one would like to generate enough examples to assure a certain degree of verification of the learned rule.

To minimize the number of examples to be generated the agent will explore the structure of the plausible version space rule. For instance, an example covered by the plausible upper bound condition and not covered by the plausible lower bound condition will always lead to rule refinement, whether the example turns out to be negative or positive. On the contrary, an example that is covered by the plausible lower bound condition will lead to a refinement of the rule only if it turns out to be negative.

To increase the degree of verification of the rule, the agent will explore the structure of the ontology and of the previous examples. The idea is to cover the example space of the rule as uniformly as possible.

Exception Handling

An important aspect of Disciple is that the ontology is incomplete and partially incorrect and is itself evolving during knowledge acquisition and learning. This distinguishes Disciple from most of the other learning agents that make the less realistic assumption that the representation language for learning is completely defined before any learning could take place. This may cause these learning agents to face the *"problem of new terms"* (i.e., the need to extend the representation language with new terms when it cannot express the concept to be learned (Dietterich et al., 1982)). Without a human expert in the learning loop, this problem is very difficult, and has not yet received a satisfactory solution.

In Disciple, a consequence of the incompleteness of the ontology is that the result of rule refinement may be a plausible version space rule with exceptions. There are several methods that a Disciple agent can use to extend and improve the ontology in order to reduce the number of rule exceptions.

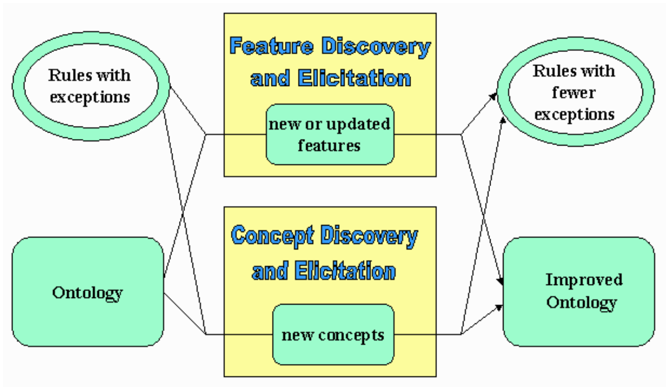

Rules with exceptions
Feature Discovery and Elicitation
new or updated features
Rules with fewer exceptions
Concept Discovery and Elicitation
new concepts
Ontology
Improved Ontology

Figure 10: Exception handling methods.

As shown in Figure 10, the agent could attempt to discover or elicit from the expert new object features or even new concepts that discriminate between the rule's examples and some or all of the rule's exceptions.

As a result of applying these methods, the rules will have fewer (if any) exceptions and the ontology will be more complete and more correct. As in the case of rule learning, the agent will use analogical reasoning to hypothesize new object features the will reduce the number of exceptions, and will ask the user to confirm them.

Consider, for instance, the following situation: some of the objects from the positive examples of a rule have a certain feature F while the corresponding objects from the other positive examples and from some of the negative exceptions do not have this feature. It may be the case that this feature is missing from the objects belonging to the positive examples because these objects are incompletely defined. Therefore, Disciple will simply ask the domain expert specific questions such as :

- *Does the object O has the feature F?*

If the answer is yes, then not only is the description of object O extended, but the rule is also refined with the feature F, and some negative exceptions are removed.

Other types of questions that Disciple will ask the expert, in order to remove exceptions, are:

- *Which feature makes the objects O_1 and O_2 similar and different from O_3 ?*
- *Which feature distinguishes between the objects O_1 and O_2 ?*
- *Could you think of a concept that covers the objects O_1 and O_2 without covering the object O_3 ?*

While some of these questions are clearly more difficult than others, they are all very specific. We therefore claim that it is easier for a domain expert to answer them than it is for a system to create natural terms that will remove the exceptions of the rules.

As one can see, this is yet another example where the agent analyzes the complex structure of the rules and of the knowledge base to suggest potential fixes, and then asks precise questions to the expert, in order to identify the correct fixes.

Experimental Results and Claims

As mentioned before, the Disciple-LAS and methodology have been used and evaluated within the HPKB program by developing the workaround agent and its knowledge base. The evaluation was performed by Alphatech and took place over a two week period in June 1998. It consisted of two phases, each comprising a test and a re-test. In the first phase, the agent was tested on 20 problems that were similar with those used for its development. Then the solutions were provided and the developers had one week to improve the knowledge base, which was tested again on the same problems. In the second phase, the agent was tested on five new problems, partially or completely out of the scope of the agent. Then again the correct solutions were provided, the knowledge base was improved and the agent was tested again on the same five problems and five new ones. Solutions were scored along five equally weighted dimensions: (1) generation of workaround solutions for all the viable options, (2) correctness of the overall time estimate for each workaround solution, (3) correctness of each solution step, (4) correctness of temporal constraints among these steps, and (5) appropriateness of engineering resources used. Scores were assigned by comparing the agent's answers with those of Alphatech's expert.

We entered the evaluation with a workaround agent the knowledge base of which was covering only about 40% of the workaround domain (11841 binary predicates). During the evaluation period we continued to extend the knowledge base to cover more of the initially specified domain, in addition to the developments required by the modification phase. At the end of the two weeks of evaluation, the knowledge base of our agent grew to cover about 80% of the domain (20324 binary predicates). This corresponds to a rate of knowledge acquisition of approximately 787 binary predicates/day. This result supports the claim that the Disciple mixed-initiative approach enables rapid acquisition of relevant problem solving knowledge from subject matter experts.

With respect to the quality of the generated solutions, within its scope, the Disciple agent performed at the level of the human expert. This result supports a second claim that the acquired problem solving knowledge is of a good enough quality to assure a high degree of correctness of the solutions generated by the agent.

Finally, our workaround generator had also a very good performance, being able to generate a solution in about 0.3 seconds, on a medium power PC. This supports a third claim that the acquired problem solving knowledge assures a high performance of the problem solver.

Based on the evaluation results, the agent developed with Disciple-LAS was selected to be further extended and was integrated into a larger system that supports air campaign planning. The integrated system was further selected to be demonstrated at EFX'98, the Air Force's annual show case of promising new technologies.

Conclusion and Future Research

Ample experimental evidence shows that manual solutions to the problem of building a knowledge base are highly inefficient and error-prone, requiring a long and intensive joint effort by a subject matter expert and a knowledge engineer. Also, the fully automated, machine learning based approaches are still to demonstrate their ability to create complex knowledge bases for real world domains. For these reasons, this problem provides an ideal test bed for developing mixed-initiative methods. In this paper we have presented such a mixed-initiative approach to the problem of building a knowledge base, and we have provided some experimental evidence of its good qualities.

However, there are also several weaknesses of our approach that we plan to address in the future. For instance, we need to develop more powerful and natural methods for hint specification by the expert. The current types of allowable hints do not constrain enough the search for explanations. Moreover, some of them are not very intuitive for the expert. Also, more powerful methods for analogical reasoning may be developed, to enable the agent to make more use of its background knowledge and to provide more assistance to the expert.

We have shown various mixed-initiative methods for cooperative problem solving, rule learning, rule refinement and exception handling. There are, however, several other processes in our knowledge base building methodology that do not yet employ mixed-initiative methods. For instance, the initial conceptual modeling of the domain is performed by the subject matter expert and the knowledge engineer and is not supported by the agent. We therefore plan to develop a modeling tool that will use abstract descriptions of tasks and objects in a scenario similar to that used in teaching the agent.

The building of the initial ontology, including the importing of concepts and features from previously developed ontologies, such as CYC (Lenat, 1995), LOOM (MacGregor, 1991) or Ontolingua (Farquhar et al, 1996), is also a manual process, unassisted by the Disciple agent. Because the initial conceptual modeling of the domain identifies many necessary concepts and their relations, it is possible to develop mixed-initiative methods that use these concepts to explore an external ontology in order to identify similar concepts and to point them to the user. This would relieve the expert and the knowledge engineer from the tedious task of manually browsing large and unfamiliar ontologies, in search of reusable concepts.

Acknowledgments. Michael Bowman, Dorin Marcu and Andrei Zaharescu have contributed to the Disciple-LAS project. This research is supported by the AFOSR grant F49620-97-1-0188, as part of the DARPA's High Performance Knowledge Bases Program.

References

Chaudhri, V. K., Farquhar, A., Fikes, R., Park, P. D., and Rice, J. P. 1998. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proc. AAAI-98*, pp. 600 – 607, Menlo Park, CA: AAAI Press.

Cohen P., Schrag R., Jones E., Pease A., Lin A., Starr B., Gunning D., and Burke M. 1998. The DARPA High-Performance Knowledge Bases Project, *AI Magazine*, 19(4),25-49.

Dietterich, T. G., London, R. L., Clarkson, K., and Dromey, G. 1982. Learning and inductive inference. Chapter XIV in Cohen, P. R., and Feigenbaum, E. A., *The Handbook of Artificial Intelligence, Vol. III*, 323-512, Los Altos, CA: William Kaufmann.

Farquhar, A., Fikes, R., and Rice, J. 1996. The Ontolingua Server: a Tool for Collaborative Ontology Construction. In *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, Alberta, Canada.

Gruber, T. R. 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, **5**(2):199-220.

Lenat, D. B. 1995. CYC: A Large-scale investment in knowledge infrastructure *Comm of the ACM* 38(11):33-38.

MacGregor R. 1991. The Evolving Technology of Classification-Based Knowledge Representation Systems. In Sowa, J. ed. *Principles of Semantic Networks: Explorations in the Representations of Knowledge*, pp. 385-400. San Francisco, CA: Morgan Kaufmann.

Mahadevan, S., Mitchell, T., Mostow, J., Steinberg, L., and Tadepalli, P. 1993. An Apprentice Based Approach to Knowledge Acquisition, *Artificial Intelligence*, 64(1):1-52.

Michalski, R. S. and Tecuci, G. (editors), 1994. *Machine Learning: A Multistrategy Approach Volume 4*, Morgan Kaufmann Publishers, San Mateo, CA.

Tecuci, G. 1998. *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies.* London, England: Academic Press.

Tecuci, G. and Kodratoff, Y. (editors), 1995. *Machine Learning and Knowledge Acquisition: Integrated Approaches*, Academic Press.

Tecuci, G., Boicu, M., Wright, K., Lee, S.W., Marcu, D. and Bowman, M. 1999. An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents. To appear in *Proc. AAAI-99,* July 18-22, Orlando, Florida, Menlo Park, CA:AAAI Press.