Case-Based Planning to Learn

J. William Murdock, Gordon Shippey, and Ashwin Ram

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Abstract. Learning can be viewed as a problem of planning a series of modifications to memory. We adopt this view of learning and propose the applicability of the case-based planning methodology to the task of planning to learn. We argue that relatively simple, fine-grained primitive inferential operators are needed to support flexible planning. We show that it is possible to obtain the benefits of case-based reasoning within a planning to learn framework.

1 Problem

A recent view of learning is one in which learning is modeled as an active, deliberative, goal-driven process involving the explicit identification and pursuit of learning goals [12, 13]. In this view, modifications to memory are treated as planning operations and learning is done by constructing a plan over these operations. The work that has been done in this area has largely focussed on search-based planning algorithms such as non-linear planning; examples of this sort of work include [1, 3, 10, 14]. Furthermore, past work in this area has generally made use of large, complex learning algorithms as the operators for the planning algorithm; for example, Meta-AQUA [1] has a library of strategies such as explanation-based generalization which it combines to form learning plans. We argue, however, that a system capable of a truly broad and flexible range of learning needs to be able to construct its own learning algorithms from more basic components.

One approach to such basic components can be found in [5] which describes a taxonomy of *knowledge transmutations* which can be thought of as basic operations over knowledge elements. An example of one of these transmutations is *generalize*. The generalize transmutation specifies that given a fact about a knowledge element one can infer that information about a more general element which that element is an instance of. Our implementation uses a set of operations which is roughly (but not exactly) a subset of those defined by Michalski; see [6] for further discussion of knowledge planning using transmutation operators. In this paper we are concerned with the algorithms that are used to plan using such operators.

Search-based methods for planning are powerful and broadly applicable. However, using very low level primitive operations such as Michalski's knowledge transmutations means that the plans themselves are substantially longer than

those formed from larger building blocks and thus may be intractable in sufficiently complex domains. Furthermore, because these primitive building blocks are so general, it may not be possible to completely identify whether an operator is guaranteed to produce a correct result. Case-based planning provides an potential solution to these problems.

Traditional case-based planning programs such as [2] have focused on planning in domains of physical action rather than in mental domains. More recent work [4, 8, 9] which has addressed mental domains has focused largely on the specific issue of using a "meta-level" CBR process to develop new adaptation strategies on top of more traditional CBR. In contrast, we are interested in how case-based planning can be extended into the mental domain of learning. Of the many advantages which case-based planning provides, the two which are most relevant to this research are:

- In complex domains, case-based planning may be considerably faster than methods which do searching.
- In domains for which no complete and correct theory is available, case-based planning may provide plausible plans on the basis of similarity to past examples while search may not be able to provide any meaningful insight.

The combination of case-based reasoning and planning to learn raises a variety of challenging issues; such a synthesis involves complex issues relating to both representation and processing. We propose a case-based method for planning to learn which provides the above benefits.

2 Example

As an example, consider a relatively novice poker[1] player. This player knows the rules to a small variety of different poker games but is not particularly familiar with the key strategic differences between these games. However, there is one specific distinction between two specific games of which the novice player is aware: the fact that the game acepots is strongly positional[2] and the fact that baseball is weakly positional. The problem that the player now wants to solve is whether some other poker variant is strongly or weakly positional.

[1] The example developed in this and succeeding sections involves a specific analysis of some particular poker games. The rationale behind this analysis and the specific details of the different games are not particularly important; all of the information which our system has about these games is illustrated in figure 1. For a relatively complete treatment of poker in general, see [15].

[2] By strongly positional, we mean that the relative value of a hand is strongly affected by the position which the individual holding the hand is seated with respect to the dealer because of the additional knowledge that players have by seeing what other players have already bet before deciding on their own bet. Again, the precise nature of this characteristic is not important to this example; what is important is that knowing the positionality is important to effective play but is not directly stated in the game rules.

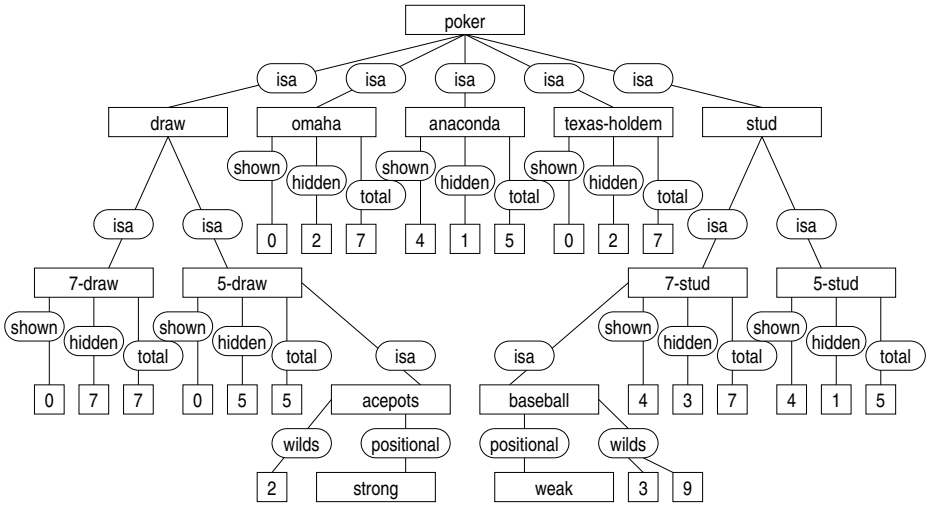

Fig. 1. This figure portrays a sample knowledge base in the domain of poker variants. Rectangular boxes indicate nodes in a semantic network. Rounded boxes indicate relationships between these nodes.

Figure 1 illustrates the sample knowledge base described in this example. A group of variants of poker are described along with a few specific characteristics of each variant.

If the player in this example had a deep understanding of what aspects of the rules of these two games affect whether the game is strongly positional then this player might be able to form generalizations for what constitutes a strongly positional poker game using a method such as EBG [7]. Similarly, if the player had a sufficiently large and diverse set of games for which this feature was known, then it would be possible to induce a decision tree mapping observable feature sets to the two positional classifications using a technique such as ID3 [11]. Lastly, a true expert at poker would probably already have in memory a precise normative characterization of what constitutes a strongly positional game and thus would be able to directly infer this feature from a description of the rules. The player in this example, however, has access to none of this information and thus cannot use any of these methods. In our example, the player does, however, have access to the following argument that texas-holdem is strongly positional:

- acepots is strong and acepots is a kind of 5-draw thus 5-draw is strong
- 5-draw is strong and 5-draw is a kind of draw thus draw is strong
- draw is strong and 7-draw is a kind of draw thus 7-draw is strong
- 7-draw is strong and 7-draw is very similar to texas-holdem in that both games involve a total of 7 cards *and* both games have each player showing none of their own cards thus texas-holdem is strong

This argument can be viewed as a plan for learning that texas-holdem is

strongly positional. The case memory portrayed in figure 1 is the initial state for this *learning plan*. The goal is that the positionality of texas-holdem is known. A plan like this one could be provided by an external observer or even developed slowly over time through experience with the individual games described in the argument. Once this learning plan was available, the further analysis of additional games could be done using case-based reasoning. An example would be the goal of determining the positionality of anaconda. This example would involve abstracting out the specific objects used in the initial plan and replacing them with alternate objects from the knowledge base. In particular, the adapted learning plan would be:

- baseball is weak and baseball is a kind of 7-stud thus 7-stud is weak
- 7-stud is weak and the 7-stud is a kind of stud thus stud is weak
- stud is weak and 5-stud is a kind of stud thus 5-stud is weak
- 5-stud is weak and 5-stud is very similar to anaconda in that both games involve a total of 5 cards *and* both games have each player showing 4 of their cards thus anaconda is weak

Notice that this example requires a variety of different sorts of learning such as generalization, specialization, and analogy. Furthermore, in order to arrive at the desired conclusion, these different varieties of learning need to be combined dynamically (i.e. the particular pattern of interactions is specific to this problem and thus no single fixed strategy will solve this problem). We argue that case-based reasoning is the mechanism for accomplishing this dynamic combination of learning actions.

3 Representation

There are several general types of knowledge which must be represented in order to develop any sort of planning system. States correspond to arbitrary situations in the world; within the planning to learn framework a state encodes a complete set of memory contents. Goals characterize a set of states which a plan is intended to achieve; in our system this is done by specifying a particular feature which a goal state should have. The representation of operators requires information about when they are applicable and what their effects are. Finally the plans themselves need to encode the operations performed, the objects (i.e. memory items) which the operations affect, and the ordering constraints over these operations take place.

3.1 States and Goals

The states which our planning to learn system plans over are actually entire knowledge bases. The initial state of a plan corresponds to all of the knowledge available to the system at the start of the plan and the intermediate and final states correspond to the intermediate and final results of executing the

plan. In our system, we have implemented this knowledge base as a semantic network, illustrated in figure 1. In this figure nodes represent concepts and links represent relationships; for example, baseball is a node which is part of a (positional baseball weak) relationship. Goals in our system are specified as conjunctive sets of variablized relationships. For example, the goal to determine the positionality of anaconda is described as {(positional anaconda ?x)}.

3.2 Operators

Learning actions are represented as operators over knowledge bases, i.e. knowledge transmutations. Four examples of transmutations are:

Generalize: Given that r holds for x and x is a y, assert that r holds for y.
Specialize: Given that r holds for x and y is an x, assert that r holds for y.
Weak Analogize: Given that r holds for x and y and that t holds for x, assert that t holds for y.
Strong Analogize: Given that r and s both hold for both x and y and that t holds for x, assert that t holds for y.

These four transmutations can be completely represented by preconditions and postconditions which are simply variablized relationships. As an example, the specialize transmutation is presented in table 1. The transmutation can apply when knowledge items exist which can bind to the variables (symbols starting with ?) in the precondition. It has the effect of asserting the relationship in the postcondition under those variable bindings. For example, to specialize the fact that draw is strongly positional to 7-draw, we use the preconditions to bind ?fr-2 to 7-draw, ?fr-1 to draw, ?rel-1 to positional, and ?val-1 to strong. The postcondition would then immediately require that (positional 7-draw strong) be asserted. A more complex transmutation, strong-analogize, is described in table 2.

Table 1: A simple example of a transmutation

transmutation specialize	
precondition	(isa ?fr-2 ?fr-1)
	(?rel-1 ?fr-1 ?val-1)
postcondition	(?rel-1 ?fr-2 ?val-1)

Table 2: A relatively complex transmutation

transmutation strong-analogize	
precondition	(?rel-1 ?fr-1 ?val-1)
	(?rel-2 ?fr-1 ?val-2)
	(?rel-3 ?fr-1 ?val-3)
	(?rel-1 ?fr-2 ?val-1)
	(?rel-2 ?fr-2 ?val-2)
postcondition	(?rel-3 ?fr-2 ?val-3)

Note that in general, the transmutations presented above are powerful enough to prove any possible statement. Since we do not have a theory here about which

transmutations should be used under which circumstances, we simply cannot reason from first principles; case-based planning provides a partial solution to this dilemma. We may be able to adapt past cases to develop plausible learning plans.

3.3 Plans

A learning plan specifies a set of operations, a goal, and a set of bindings for the variables specified in the goal. A plan for the (positional anaconda ?x) goal would specify the final binding of ?x to weak as well as a series of steps in the form of transmutations and bindings for the variables in the transmutations. For example, the third and fourth steps in the initial plan described in section 2 are represented in tables 3 and 4; the variables are the same variables that appear in the description of the transmutations in tables 1 and 2.

The third and fourth steps in the positionality of texas-holdem plan

Table 3: The third step

step plan1-3	
of	specialize
has-bindings	(?fr-1 draw)
	(?fr-2 7-draw)
	(?rel-1 positional)
	(?val-1 strong)

Table 4: The fourth step

step plan1-4	
of	strong-analogize
has-bindings	(?fr-1 7-draw)
	(?fr-2 texas-holdem)
	(?rel-1 shown)
	(?val-1 0)
	(?rel-2 total)
	(?val-2 7)
	(?rel-3 positional)
	(?val-3 strong)

4 Algorithm

Our algorithm for case-based planning in the knowledge domain has four basic components: retrieval, adaptation, verification, and storage. Goals and plans correspond to the problems and cases in case-based reasoning respectively; the system begins with a goal, retrieves a plan from memory, adapts it to suit the new goal, stores the modified plan in memory, and returns the new learning plan as output.

4.1 Retrieval

The system we have developed does retrieval by conducting a linear search through a list of plans. The search returns only the first plan which accomplishes

a goal similar the goal of the current problem. Our similarity metric requires that each postcondition for the goal of the plan in memory match the corresponding postcondition for the new goal being examined, with a match between postconditions requiring that they involve the same relation and have at least one of the arguments to the relation be equivalent (i.e. either they are the same node or they are both variables); this is similar to the similarity metric in [2, 16]. Thus, for example, (positional anaconda ?x) would match (positional texas-holdem ?y) but not (shown anaconda ?x) or (positional ?x strong). This algorithm is essentially based on the observation that such matches involve essentially demonstrating the same fact about a different data item; in the examples we have considered, it is far more plausible for such matches to use the same learning plan than it is for other sorts of matches (such as proving something different about the same data item).

4.2 Adaptation

The primary problem that we face in designing adaptation strategies for case-based planning to learn is one that runs throughout the field of case-based reasoning: the trade-offs between simple and complex strategies. Simple adaptation strategies are very limited in the breadth of modifications that they make and thus only work when a case in memory is extremely close to providing a solution to the new problem. Complex adaptation strategies require a great deal of complex reasoning. Consequently, they partially surrender both of the primary advantages of case-based reasoning that we outlined in section 1; they are less efficient than simpler adaptation mechanisms and they often provide weaker evidence for correctness based on similarity to an existing case (since the results are less similar to the existing cases). There is a spectrum of more and less powerful adaptation strategies and an ideal system would be able to draw from strategies all over the spectrum depending on the nature of the problem being addressed.

The most powerful and general adaptation mechanism which we have developed is to substitute the desired values for the nodes in the postconditions of the final step in the plan, then work backwards to construct preconditions which are consistent with those postconditions (using the relationships in the existing plan), and finally construct postconditions for the previous step that are consistent with those preconditions and so on.

Consider the problem described in section 2 of adapting the plan for learning that texas-holdem is strongly positional to the problem of learning that anaconda is weakly positional. We can describe this plan as pursuing the goal (positionality anaconda ?x). The last step of this plan is described in table 4. Our system begins by substituting the new problem's values for the variables in the postcondition of the strong-analogize transmutation; the postcondition of (?rel-3 ?fr-2 ?val-3) which was bound to (positional texas-holdem strong) requires that the value ?fr-2 now be bound to anaconda. Because this adaptation strategy involves holding the relations constant, the bindings for ?rel-1, ?rel-2, and ?rel-3 would be left unchanged. This would leave us with a partially adapted version of this step as shown in table 5.

The different stages which the adaptation of the fourth plan step goes through. Changes are from the previous stage are marked in **bold**.

Table 5: An early version of the last step of the plan; several values have not yet been filled in.

step plan1-4-modified-1	
of	strong-analogize
has-bindings	(?fr-1 -)
	(?fr-2 **anaconda**)
	(?rel-1 shown)
	(?val-1 -)
	(?rel-2 total)
	(?val-2 -)
	(?rel-3 positional)
	(?val-3 -)

Table 6: A prospective version of the last step of the adapted plan; only one value is still missing.

step plan1-4-modified-2	
of	strong-analogize
has-bindings	(?fr-1 **5-stud**)
	(?fr-2 anaconda)
	(?rel-1 shown)
	(?val-1 **4**)
	(?rel-2 total)
	(?val-2 **5**)
	(?rel-3 positional)
	(?val-3 -)

Table 7: The final version of the last step of the adapted plan; all values have been filled in.

step plan1-4-modified-3	
of	strong-analogize
has-bindings	(?fr-1 5-stud)
	(?fr-2 anaconda)
	(?rel-1 shown)
	(?val-1 4)
	(?rel-2 total)
	(?val-2 5)
	(?rel-3 positional)
	(?val-3 **weak**)

The strong-analogize transmutation has a precondition (?rel-1 ?fr-2 ?val-1); since ?rel-1 is bound to shown and ?fr-2 is bound to anaconda, we can directly retrieve from the knowledge base the value of 4 for ?val-1. Similarly, we can directly obtain the value of 5 for ?val-2. This leaves us with with ?fr-1 and ?val-3 still unaccounted for. Since we can't directly infer values for these variables, we can look in the knowledge base for nodes to fill these variables which are consistent with these values; since there are preconditions (?rel-1 ?fr-1 ?val-1) and (?rel-2 ?fr-2 ?val-2) we can deduce that ?fr-1 must be a node which has a value of 4 for shown and a value of 5 for total. The only values for which this is true are anaconda and 5-stud. We can create two different potential bindings to investigate and continue processing each of them. The system uses a heuristic which causes us to favor the new value 5-stud (since doing a strong-analogize from anaconda to itself is clearly not going to get us anywhere). Thus we first consider the possibility that ?fr-1 is bound to 5-stud. This presents us with a nearly complete adaptation of this plan step portrayed in table 6, leaving us with the goal of determining the value of ?val-3. We use this information to form a subgoal and apply this same mechanism to the previous step (table 3). This process continues backwards through the plan until we adapt the first step of the plan, and find that all of the bindings are filled (since we already have a value, weak, for the positionality of baseball). With all the values filled, we can then propagate this value forward fill the missing value slot in each of the other steps. Thus we get the last step of the final, adapted plan as shown in table 7.

4.3 Verification

One of the advantages of using case-based methods is that they may be able to provide plausible solutions in situations in which there is simply not enough

information to provably identify a correct solution. For example, given the information in the poker example in figure 1 it is simply not possible to accurately verify whether the result of a plan execution is correct. In general, there may be a wide variety of characteristics which determine or correlate with correctness of a plan. We have worked primarily in domains in which potential verification is very limited because demonstrably correct solutions are not available; the general problem of verification is an important topic for future research.

4.4 Storage

The storage function for our system simply as appends the new case on to the front of the list of cases. However, it might be possible to make the retrieval function more efficient if a more complex store function involving hash tables or trees is used. Ultimately the resolution of this issue depends directly on the kind of retrieval operation being used; since our research has used very simple retrieval, our storage mechanism is also very simple.

5 Conclusion

We have developed an account of how case-based planning can be extended into the mental domain of learning. One major issue in this research is the question of what domains and problems are particularly amenable to case-based planning to learn. The domain of poker variants provides a concrete example of some features of domains in which case-based planning to learn is particularly applicable:

- Poker variants can often be classified into a relatively broad and deep "isa" hierarchy. This is essential to the usefulness of the particular transmutations (e.g. generalize) which we have implemented.
- Poker variants can also be completely specified as a (generally long) list of completely operationalized features as they are essentially mathematical constructs. This is an important feature of any domain for which we intend to apply concrete, parsimonious inferences such as those found in our knowledge transmutations.
- Most importantly, poker variants also have a great many interesting features (e.g. strategic properties) which are not directly observable and may often be unknown to a reasoner and impossible or intractable to derive from first principles. Often times these sorts of characteristics really are derived from analysis of the relationships of the variants to other variants in which the reasoner has access to a greater understanding. This feature is essential for there to be complex plans for learning. Such complex plans are essential for case-based planning to learn; in a domain in which all plans were extremely short and simple there would be no meaningful way to distinguish between plans in memory to adapt, no computational benefits to reusing plans, and no significant evidence that the new plans are particularly plausible.

While there are many useful approaches to planning which are based on a search through the state space, when there is a library of past planning cases available the technique of case-based planning may provide a variety of advantages over these techniques. Most notably, case-based planning may be considerably faster because it does not require a search through the entire space and may provide plausible solutions to problems where there is insufficient information to construct any solution from first principles. Our work has demonstrated the application of case-based planning to the planning-to-learn framework.

References

1. M. Cox, Introspective Multistrategy Learning: Constructing a Learning Strategy Under Reasoning Failure, Ph.D. Thesis, Technical Report GIT-CC-96/06, College of Computing, Georgia Institute of Technology, 1996.
2. K. Hammond, *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, 1989.
3. L. Hunter, Planning to learn. *Proc. Twelfth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1990.
4. D. Leake, Combining Rules and Cases to Learn Case Adaptation. *Proc. of the Seventeenth Annual Conference of the Cognitive Science Society*, 1995.
5. R. Michalski, Inferential Theory of Learning as a Conceptual Basis for Multistrategy Learning. *Machine Learning*, 11, 1993.
6. R. Michalski and A. Ram, Learning as goal-driven inference. In *Goal-Driven Learning*, A. Ram and D. Leake (eds.), MIT Press / Bradford Books, 1995.
7. T. Mitchell, R. Keller, S. Kedar-Cabelli, Explanation-Based Generalization: A Unifying View. *Machine Learning*, 1, 1986.
8. R. Oehlmann, Metacognitive adaptation: Regulating the plan transformation process. In *Proceedings of the AAAI-95 Fall Symposium on Adapation of Knowledge for Reuse*, D. Aha and A. Ram (eds.), pp. 73 - 79, San Mateo, CA: AAAI-Press.
9. R. Oehlmann, D. Sleeman, and P. Edwards, Learning plan transformations from self-questions: A memory-based approach. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pp. 520-525, Cambridge, MA: AAAI-Press, 1993.
10. A. Quilici, Toward automatic acquisition of an advisory system's knowledge base. Applied Intelligence, In Press.
11. J. Quinlan, Induction of Decision Trees. *Machine Learning*, 1, 1986.
12. A. Ram and L. Hunter, The Use of Explicit Goals for Knowledge to Guide Inference and Learning. *Applied Intelligence*, 2(1), 1992.
13. A. Ram and D. Leake, Learning, Goals, and Learning Goals. In *Goal-Driven Learning*, A. Ram and D. Leake (eds.), MIT Press / Bradford Books, 1995.
14. Learning by Observing and Understanding Expert Problem Solving, Ph.D. Thesis, Technical Report GIT-CC-92/43, College of Computing, Georgia Institute of Technology, 1992.
15. J. Scarne, *Scarne's Guide to Modern Poker*, Simon & Schuster, 1984.
16. Case-Based Reasoning in PRODIGY. In *Machine Learning: A Multistrategy Approach Volume IV*, R. Michalski and G. Tecuci (eds.), Morgan Kaufmann Publishers, Inc., 1994.

This article was processed using the LaTeX macro package with LLNCS style