

# THE IMPORTANCE OF IMPLICIT AND EXPLICIT KNOWLEDGE IN A PILOT'S ASSOCIATE SYSTEM<sup>1</sup>

David L. Perschbacher  
Keith R. Levi  
Honeywell Systems and Research Center  
3660 Technology Drive MN65-2500  
Minneapolis, MN 55418  
*davep@src.honeywell.com*

Mark Hoffman  
ISX Corporation  
1850 Park Pl. Suite 420  
Marietta, GA 30067

## ABSTRACT

Fielding an operational Pilot's Associate (PA) will require both implicit and explicit representations of knowledge. Speed and memory performance requirements for PA will be aided by the use of implicit representations of knowledge. Acquiring and maintaining the large knowledge bases for PA will, by contrast, be aided by having explicit knowledge representations. We have been investigating such explicit representations in a 10 person-year research project sponsored by the Wright Research and Development Center. A critical contribution of this research has been to develop concepts that make machine learning applicable to real-time control and execution systems such as Pilot's Associate.

## 1. Introduction

The Pilot's Associate (PA) planning systems must operate in an extremely dynamic environment in which many events cannot be accurately predicted or modeled. Traditional artificial intelligence (AI) planners (4,9) have always assumed that events and time scales were such that complete plans could be generated for all contingencies. This is not feasible in the highly dynamic PA domain. Such an inability to fully model the world has led to the recent development in AI of *reactive* planners (1,5,11).

However, systems of these sorts are not appropriate for PA because they typically make it difficult to integrate accepted expert knowledge and they also exhibit poor planning performance. The Lockheed PA team has developed a unique combination of traditional AI planning and reactive planning (3) that captures the positive aspects of both these approaches.

The PA planning systems have until now been mainly concerned with implicit representations because they have been focused on real-time performance in a limited set of scenarios. Explicit representations of information will become critical requirements as they scale the system up to include more scenarios. In this paper we describe how machine learning techniques (2,8) can automatically transform such explicit representations into the implicit representations required by PA.

## 2. Expert Performance Requires Implicit Knowledge Representation

The US Air Force's PA program is developing artificial intelligence software for an electronic copilot. Future aircraft will have tremendous sensor, control, and information resources. These resources will be potentially of great assistance, but they will also be capable of easily

<sup>1</sup>This work was supported in part by the Learning Systems Pilot Aiding contract from the Wright Research and Development Center (Contract Number F33615-88-C-1739). We are pleased to acknowledge the support of our contract's technical monitor, Gurdial Saini. We are also indebted to the following members of Lockheed's Pilot's Associate team: Gary Edwards of ISX, David Smith of Lockheed, Norm Geddes of Applied Systems Intelligence, and Belinda Hoshtrasser of Search Technology. Jerry DeJong of the University of Illinois has also been an important contributor to this effort.

overloading the pilot if they are not made available to him in an intelligent way. The goal of PA is to monitor the status of the pilot's mission and understand the situation and the pilot's goals and intentions. This understanding will be required to properly assist the pilot.

## 2.1 Pilot's Associate Requirements

Monitoring a mission and understanding a pilot's intentions in real-time is an extremely challenging task. It requires *several kinds of expertise*. In particular, the PA system includes five cooperating expert systems. The System Status module monitors the internal state of the aircraft. The Situation Assessment module monitors the state of the world external to the aircraft. The Mission Planner module performs route planning for the entire mission. The Tactics Planner module creates tactical plans for short term actions and maneuvers. The Pilot Vehicle Interface module interprets pilot actions and information requests and transmits information from the other modules to the pilot.

Each of these modules require *large knowledge bases* to attain expert performance in their task. Much of this knowledge is shared between different modules, but it may be represented differently and reasoned about differently in each module. Finally, all of the required interpretation, message passing, and decision making must take place in *real-time* and with a high degree of reliability and accuracy.

## 2.2 Need for Implicit Knowledge Representations

The principal reason that PA must have implicit representations of knowledge is the real-time requirement. One of the key ways for PA to achieve fast performance is to compile out intermediate reasoning steps and retain only the final decisions or actions associated with a given initial state. For example, having a SAM radar tracking you is generally a bad situation. One strategy to defeat the radar is to become invisible by reducing the aircraft's signal-to-noise ratio. One way to do this if the SAM is using doppler is to deny doppler range data. This can be done by flying a constant distance from the SAM site. One way to do this is to fly a circular path around the SAM. This requires that the pilot turns to a certain heading.

PA does not need to retain all of this background knowledge about defeating a SAM site. It simply needs to know what actions should be taken in the situation that a SAM site is tracking it. Thus, PA's Tactics Planner might have a plan that ascertains the conditions under which it was being tracked and then simply recommends that the pilot turn to a certain heading. All of the intermediate reasoning about why this action is appropriate can be omitted.

In addition to omitting intermediate reasoning steps, there is a second important way of implicitly representing knowledge that contributes to efficient performance in PA. This is to have specialized control structures for the different PA modules. For example, most tactical maneuvers have explicit timing constraints that must be satisfied if they are to be successful. The Tactics Planner, however, does not explicitly represent many of these temporal constraints. This is because they are implicitly represented in the Tactics Planner's control structure which continually monitors a situation over time until a particular condition is observed. This condition's occurrence will correspond to the temporal constraint being realized.

For example, there might be a constraint that the pilot maintains a certain heading *until* the SAM site switches from track to search mode. The Tactics planner does not have to explicitly represent this constraint. Rather, its control structure is such that it will repeatedly sample the mode of the SAM site's radar and will report success of the maneuver only when the radar is observed in search mode. The *until* relation is nowhere explicitly represented in the Tactics Planner. It is implicit in the control structure.

To summarize, implicit representations of knowledge play an important role in achieving real-time performance for PA. Two principal sources of implicit knowledge in PA are compiling out intermediate reasoning steps and having special purpose control structures.

## 3. Knowledge Acquisition is Facilitated by Explicit Knowledge Representations

Figure 1 contains two rules that are part of the Tactics Planner's specialized representation for a simple doppler notch plan. One rule, in effect, states that this plan should be selected if there is a SAM-site in track mode. The other says that if the plane is not flying close to a perpendicular

heading from the SAM then suggest a new heading to the pilot that will satisfy the perpendicular condition. We conjecture that it is much more difficult to produce the tactical plan in this complex representation than it is to acquire general rules in a simple representation such as the rule in Figure 2. The Tactics Planner representation is more difficult to acquire because it requires much more information to be built into it. The Tactics Planner could never execute in real-time if it represented and reasoned about rules such as the one in Figure 2. Thus, even though it may be easier to obtain simple rules, they are not in themselves sufficient for the Tactics Planner.

```
(DEFRULE doppler-notch-selection
  TEMPS sam-site radar noa
  IF
    (setq sam-site
      (cadaar (assertions '(degrade +)
                          :who parents)))
    (setq radar (prop 'radar-mode sam-site))
    (eq 'track radar)
  THEN
    (assert '(invocation-request ,(class) ,@plant)
      :who parents)
    (assert '(sam-site-data ,sam-site))
  GOAL ((select-doppler-notch-maneuver)))

(DEFRULE doppler-notch-update-heading
  TEMPS noa heading
  LOCAL sam-site
    (cadaar (assertions '(sam-site-data +)))
  IF
    (setq noa
      (abs (nose-off-angle *lead-plane* sam-site)))
    (or (> noa (+ 90 *perp-heading-deviation*))
        (< noa (- 90 *perp-heading-deviation*)))
  THEN
    (remove-assert '(parameter +))
    (setq heading
      (perpendicular-heading *lead-plane* sam-site))
    (assert '(parameter heading ,heading))
    (suggest)
  GOAL ((update-doppler-notch-maneuver)))
```

Figure 1. Tactics Planner Rules

We have been using Explanation Based Learning (2,8) to (semi-) automatically transform the simple rules into the specialized representation required for PA (3,6,7,10,12). The major steps in this translation process are depicted in Figure 3. We use EBL to learn a tactical plan by observing a single example of a tactic flown on a flight simulator. EBL accomplished this by using an explicit domain theory to explain how the example achieves a stated goal. For example, the goal might be to have a SAM site switch from track mode to search mode. The domain theory contains general knowledge about the world. We have used a set of rules such as the one shown in

Figure 2 to learn a doppler notch plan for the PA Tactics Planner by observing a single example of a pilot flying such a maneuver on a flight simulator. The result of the EBL process is the macro shown in Figure 4.

```
(c-rule
  :name 'maintain-constant-distance
  :type :non-max
  :pattern
  '(<-
    (maintain-constant-distance
      (agent1 ownship)
      (agent2 ?sam-name)
      (interval ?int1))

    (and
      (relative-track-position
        (track-id ?sam-name)
        (azimuth ?track-obj-az)
        (interval ?int1))

      #f(abs-value ?track-obj-az ?abs-az)
      #p(abs-diff-lt ?abs-az 90 7.5) )))
```

Figure 2. Simple Explicit Representation.

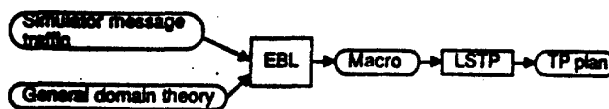


Figure 3. The Machine Learning Process.

```
If (track-class (track-id ?id) (object-type 1))
  (track-status (track-id ?id) (radar-mode track))
  (relative-track-position (track-id ?id) (azimuth ?noa))
  (fctn (abs-value ?noa ?abs-noa))
  (pred (abs-difference-less-than ?abs-noa 90 7.5))

Then (safe-from-sam (target ownship) (sam-site ?id))
```

\* temporal conditions are not displayed

Figure 4. Doppler Notch Macro.

The macro shown in Figure 4 is essentially the explanation of how the pilot achieved the goal of defeating the SAM with the intermediate reasoning steps of the explanation removed. All that remains are the initial conditions and any actions that were performed and the conclusion. This macro rule, however, is still far from the representation required by the Tactics Planner. Most of the required information is present, but it

is far from the correct format. For example, the Tactics Planner uses different types of information in different ways. For example, Figure 1 contains a *selection rule* and an *execution rule*. Selection rules check for conditions that must be true in order for a plan to be viable. That is, only select a plan if these conditions are satisfied. Execution rules monitor conditions that can change during the life of the plan and recommend that the pilot take actions to achieve a desired state of the condition, if necessary.

We have defined two ways of categorizing the clauses in the EBL macro that allow us to properly partition the clauses as required by the Tactics Planner. First, we categorize a clause according to whether it represents something that a pilot can easily control. For example, turning a sensor on or off is easily controllable, but getting a sensor locked-on to a track object or making the sensor operational if it is malfunctioning are not necessarily easily controlled by the pilot. Second, we categorize clauses according to whether their arguments are constants or variables. If the arguments are variables we further distinguish whether they are bound in the goal of the plan.

Using these categorizations we can automatically sort the macro's clauses into Tactics Planner rule types. For example, execution rules come from clauses that are easily-pilot-controllable and have variables that are not bound in the post-conditions of the macro. Since they are not bound in the post-condition, they are updateable during the execution of the plan. Selection rules come from clauses that are not easily controllable and have constants as arguments. The constant arguments act like test conditions. Since they are not easily controllable the plan should only be selected if these clauses are satisfied.

After the macro clauses are partitioned according to different types of rules there is still a further translation step required to put them in the specialized syntax of the Tactics Planner's rules. We are presently working on this step, and it appears that most of this process can also be automated.

In summary, our research has shown that most of the specialized knowledge required for the PA Tactics Planner can be automatically acquired using explanation based learning and an explicit and relatively simple domain theory.

#### 4. System Maintenance is Facilitated by Explicit Knowledge Representations

An important issue for our approach to acquiring knowledge for PA is the size of the required explicit domain theory. If the required domain theory for EBL is sufficiently large then it might be more work to build this domain theory than it would be to simply build all of the tactical plans directly, even if the explicit domain theory rules are each individually easier to build. We believe, however, that this will not be the case. Our argument to support this belief is that the EBL domain theory is essentially a model of the primitive functionality of the aircraft and its environment. Tactical plans, in contrast, model all possible behaviors arising from the functionality of the aircraft and its environment. This is analogous to a set of axioms and the set of all possible theorems that can be derived from the axioms. The former is typically a finite set and the latter is typically infinite.

Thus, this argument implies that it will not only be easier to create an initial PA system using our approach, but once the system has been developed it will be much easier to modify and adapt the system. This is because the same underlying explicit domain theory should be able to be re-compiled in different ways to create the new plans.

At least as important for system maintenance is the fact that (semi-) automatically generated plans should have several advantages over hand-generated plans. Automatically generated plans should be more *consistent*. For example, it is well accepted among TP knowledge engineers that there is significant variability in how different knowledge engineers will develop the same plan, and even how the same knowledge engineer will develop the same plan if he/she were to do it on different days. We expect that automatically generated plans will be more *complete*. For example, our automatically generated doppler notch plan included the TRACK-ID as a parameter to send to the PVI. This parameter had been inadvertently omitted by a Tactics Planner knowledge engineer who had previously developed this plan by hand. This omission was the cause of a pernicious and long-undetected bug in our simulator system. In a similar sense we expect the automatically generated plans should be more *accurate* and also better *justified* than hand-generated plans.

## 5. Summary

Both implicit and explicit representations of knowledge will be required to field an operational Pilot's Associate. The PA program has been focusing mainly on performance aspects of the system to this point. This has directed their attention to implicit representations of knowledge. As PA attempts to scale-up its knowledge bases and begins to worry more about maintaining and adapting the system it will have to worry more about having explicit representations of its knowledge.

## 6. Conclusion

We are quite optimistic that over the next year we will successfully demonstrate that our EBL approach can be successfully employed to learn simple plans for the PA Tactical Planner. There remains, however, the question about how successfully this approach will scale up. At present we only have conjectures that the combinatorics of developing the domain theory needed for EBL will be more favorable than the combinatorics of directly developing tactical plans. In addition to the scaling-up question there are several other important research areas that require attention with respect to our EBL approach, e.g., temporal reasoning, geometric reasoning, reasoning with uncertainty, reasoning about psychological models of wingmen and enemies, and reasoning with imperfect domain theories.

Although this list may appear daunting, we are encouraged by our progress to date. One reason for optimism is that we have made significant progress on at least one of these issues even though we had not originally planned to address it for this effort. This was the problem of temporal reasoning. We had originally hoped that we could choose scenarios that would not require explicit temporal reasoning, but we found that this was not feasible. This was a significant technical challenge for us, but we were able to develop a limited solution to the problem that appears to work well for this domain (10). Finally, we expect that our approach will be useful to PA even before all of these issues are resolved. For example, one benefit of our work to date is that some of our results are already being incorporated by the PA team into their plans for the next phase of PA, e.g., the principles we presented above for automatically identifying types of Tactics Planner

rules. We expect this trend to grow stronger as our work progresses.

## REFERENCES

1. Agre, P., and Chapman, D. Pengi: An Implementation of a Theory of Activity. Proceedings of the National Conference on Artificial Intelligence, Seattle, WA, July 1987, pp. 268-272.
2. DeJong, G., and Mooney, R. Explanation-based learning: An alternative view. *Machine Learning*, 1, 1986, pp. 145-176.
3. Edwards, G., and Hoffman, M. The Kadet Planning Framework. DARPA Knowledge-Based Planning Workshop, December 1987, pp. 16-1.
4. Fikes, R. E., and Nilsson, N. J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2, 3/4 (1971), pp. 189-208.
5. Firby, R. J. An Investigation into Reactive Planning in Complex Domains. Proceedings of the National Conference on Artificial Intelligence, Seattle, WA, July 1987, pp. 202-206.
6. Levi, K. R., Shalin, F. L., Wisniewski, E. D., and Scott, P. An analysis of machine learning applications for pilot-aiding expert systems. Final report for contract no. F33615-86-C-1125. AFWAL Technical Report TR-87-1147.
7. Levi, K. R., Shalin, V. L., and Perschbacher, D. L. Automating acquisition of plans for an intelligent assistant by observing user behavior. *International Journal of Man-Machine Studies* (in press).
8. Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based generalization: A unifying view. *Machine Learning*, 1, 1986, pp. 47-80.
9. Pednault, E. Extending Conventional Planning Techniques to Handle Actions with Context Dependent Effects. Proceedings of the Seventh National Conference on Artificial Intelligence, St. Paul, MN, August 1988.

10. Perschbacher, D. L., Levi, K., Shalin, V. L., and DeJong, G. Learning plans from observations that extend over time. Unpublished manuscript, 1990.
11. Schoppers, M. J. Universal Plans for Reactive Robots in Unpredictable Environments. Proceedings of the Tenth International Joint Conference on Artificial Intelligence. Milan, Italy, August 1987, pp. 1039-1046.
12. Shalin, V. L., Wisniewski, E. D., Levi, K. R., and Scott, P. D. A Formal Analysis of Machine Learning for Knowledge Acquisition. International Journal of Man-Machine Studies, 29, 1988, pp. 429-446.