Intent Inferencer-Planner Interactions

K. Brock Stitts
John M. Hammer

Applied Systems Intelligence, Inc.
11660 Alpharetta Highway
Suite 720
Roswell, GA 30076

bstitts@asinc.com
jhammer@asinc.com

Abstract

Interaction between automated planners and intent inferencers is a difficult problem that is not always well understood. This problem can be framed in terms of a number of key issues: shared ontology, human interaction, and display of proposals.

Shared ontology is a key issue from both practical and theoretical perspectives. Without sharing, intent inference outputs must be interpreted by the planner into its own representations, and, of course, the intent inferencer must do the same with the planner outputs. Furthermore, absent a commitment to a shared ontology, it is unclear practically whether there is any commitment between the design teams to interact in any way whatsoever that would support true collaborative interaction between the human operator and the intelligent system. Committing to a shared ontology is no panacea. Ontologies are hard, and shared ontologies are even harder. Given that knowledge based processing is often embedded within elements of the ontology, the representational needs of planning and intent inference place considerable demands on the shared ontology.

Interaction with a human operator is a difficult problem generally in software and especially so for planners. One of the most difficult problems for the planner is that the human operator is nearly always correct. Consequently, the plans that the planner wants to pursue may not be the plans of the human operator. This has several implications. First, the planner must follow the human's lead, which is, at first glance, a modification to the very essence of planning: selecting courses of action. In other words, the planner will have to activate plans that it has already decided are not the best course of action. This may mean that the planner would be required to override its own knowledge to follow the human's lead. All of the preceding assumes that the planner is truly going to follow this lead. Another possibility is that the human has chosen an erroneous plan. If the planner has any interest in intervening, it would need to differentiate between adequate choices and erroneous choices (a design choice that seems to be infrequently chosen). A third possibility, and not necessarily a bad one, is to follow the operator's lead silently. If the planner were following the lead of another automated planner, a request for rationale could be appropriate. This request might well be inappropriate for a human operator under circumstances in which intelligent systems are currently being deployed. A final difficulty with human operator is individual differences, in which human operators choose make choices for strictly preference reasons. It would be poor for the planner to repeatedly suggest option A when the human always chooses option B.

Display of proposals is another key issue and another opportunity to solve problems that are difficult to address in the preceding issues. The need for and availability of rationale may be as important as the proposed plan. In situations where sufficient time is readily available, the human operator will likely want to examine the rationale that supports the plan. Rationale is clearly related to the plan itself but is something the planner's designers may not have considered to be important. In many operational contexts, the operator has insufficient time for a comprehensive review, and thus the plan itself is displayed. The human operator's visualization of the proposed plan, supported by human pattern recognition, and trust (or lack of trust) in automation will likely determine whether the plan is followed.

Introduction

The basic mechanisms for interaction between an intent interpreter and a dynamic planner have been spelled out in several publications such as Geddes (1989) and Geddes and Hoshstrasser (1989). The purpose here is to extend this work by providing potential solutions to certain problems that arise due to these interactions. These potential solutions will be tested out in work such as the U.S. Army's Vetronics Technology Insertion (VTI) program.

We begin with a description of the knowledge structures and intent interpretation process borrowed from Geddes and Lizza (2001).

Knowledge representations The intent interpreter uses knowledge representations derived from the natural language processing work of Schank, Cullingford and Wilenski (Schank and Abelson 1977; Schank and Riesbeck 1979). Interpreting the intentions of active

entities in a strongly causal environment has many characteristics common to interpreting narrative stories.

Actions. Entities must interact with the environment by performing actions. Actions are chosen to represent the primitive manipulations of the environment that will serve as the directly observable inputs to the intent interpreter.

Scripts. Scripts represent procedural knowledge as weakly ordered sequences of events. Events are composed of actions and constraints. Scripts are segmented to provide serial and parallel execution paths.

Plans. Plans represent abstract methods that might be expected to achieve one or more goals. While a plan may have a script associated with it that defines its nominal execution, this is not required.

Goals. Goals represent desired states of the environment that may be achieved at some time in the future as a result of performance of a plan.

These knowledge entities are combined in a structure called a plan-goal graph (PGG), in which a plan's children are the set of subgoals or actions needed to complete it, while the goal's children are plans that can be performed to satisfy the goal.

Interpretation process

The first step in interpretation of an input primitive action is to determine if the action is a part of an ongoing procedural sequence, represented by an active script. If it is found to be an appropriate part of an active script, the input action is explained. If all active scripts are evaluated and the action is not appropriate for any of the scripts, interpretation using the PGG is attempted. PGG interpretation is an abductive process that traverses the PGG until a successful path is found from the input action to an active instance of a plan or a goal. The simplest case is when the action is directly linked to an active plan instance, and is simply the continued execution of a plan that was already active. If the input action is the start of new intended activity, the search through the PGG will identify hypothetical plans that the action might be a part of and hypothetical goals that might be being pursued. This search may terminate in a path from the action through a series of lower level plans and goals until reaching an active plan or goal instance at a higher level of abstraction. The resultant path is the explanation of the action, and all plan and goal instances on the path are now promoted to active. In addition, truth maintenance processing is performed and any scripts associated with any of the new plan instances are promoted to active. The entities whose behavior is being interpreted are not assumed to be completely consistent or correct.

In addition, the knowledge in the intent interpreter is not assumed to be perfect. As a result, an input primitive action may not be explained. Unexplained inputs provide a mechanism for both causal error analysis and for machine learning.

Decomposition Planner

Dynamic planning is the capability to continuously plan or replan activities in real-time, adapting behavior as the situation unfolds. The decomposition planner, described in Elmore et al. (2000), implements dynamic planning. It does this by working in the opposite direction as the intent interpreter. Beginning with the highest-level plan in the PGG, the planner attempts to decompose the graph until it has reached leaf-level plans. These plans are associated with scripts that generate actions. Because the graph is only decomposed as events dictate, there is no master plan. Planning is "just in time."

The planner and the intent interpreter are part of an associate system, which is a real-time decision aid that can act in concert with users to perform complex tasks. It is described in Geddes (1997).

Shared Ontology

Some shared ontology is needed. Ideally, the ontologies for both the planner and the intent interpreter would be identical. This is typically the goal when building associate systems. But for virtually every problem, there is some degree of mismatch between how a decomposition planner would solve the problem and how a human would. At least in some cases, a decomposition planner will use some numerical recipe that a human would not. Suppose, for example, there is some computation made to determine whether to use Plan A or Plan B. This algorithm may not be one a human would actually use. But as long as there exists a similar decision made by the human, the intent interpreter can cope with differences between the human and the planner. To continue the example, suppose the planner picks Plan A, but the human operator later selects Plan B (by performing an action that the intent interpreter can explain with Plan B, but not with Plan A). Through appropriate design of the PGG, Plan A will be revoked once Plan B is instantiated by the intent interpreter. The associate system still has a general understanding of the situation and can plan and act accordingly.

This indicates at what level knowledge should be represented. A computation could in many cases be represented as a set of PGG nodes. But if such a set does not correspond to an operator's plans and goals, there is

no reason to represent it as a PGG. At the other end of the spectrum, if a planner represents with a single plan a set of the operator's plans and goals, the planner will not be able to react appropriately when the operator's plans and goals differ from the planner's. For example, if the planner were a resource allocator using an operations research approach, there would be a single plan that took in all the relevant data and produced a global schedule for use of the resources. If an operator wants to modify a portion of such a schedule, he or she would have to input new data and rerun the entire planning process again. Localized replanning is not possible.

Human Operator Interactions

One of the principle problems in simultaneous planning and intent interpretation involves overriding the planner's plans with an operator's plans. Once a plan has been overridden, there needs to be some mechanism to guarantee that a planner capable of dynamically replanning will not simply override the operator's override upon receiving a trigger to replan. One way of preventing such an override is to tag the operator's plan as having been "operator initiated." The decision logic for determining whether to replan or not then must include information about this tag. A difficulty occurs here if there are some cases in which it makes sense to override the operator's plans because the context in which the operator made these plans has sufficiently changed. This is dealt with by monitoring for such conditions. When the condition is detected, the planner replans and presents a new proposal.

For example, suppose a driver's associate system has a plan to get to a location by a certain time. Initially, the planner suggested taking a route using interstate highways. The driver, however, overrode the plan and selected a route that used back roads. Suppose further that the driver is halfway there, but has made such slow progress that the current route would no longer permit him to get to the location on time. At this point, the associate system can detect this problem and propose a new plan.

Asymmetries in planning and intent interpretation

As the planner decomposes the graph, it performs computations to determine which branch to take and/or what parameter values to use. Having the intent interpreter perform these computations does not make sense; at the very least, the inverse of the computation (with the inputs and outputs of the computation reversed) should be performed. But since the purpose of these computations is to aid the planner in making a decision, and the intent interpretation process involves trying to explain a decision that has already been made, there is no need for the intent interpreter to perform these computations.

Referring back to the driving example, if the planner has a computation that tells it to use interstate highways, there is no need for the intent interpreter to perform such a computation, as the driver has already decided to use back roads.

What to do when the wrong explanation is picked

Intent interpretation, as an abductive reasoning process, is not logically valid. Unless the knowledge is designed in such a way that only one interpretation is possible at any one time, the intent interpreter will sometimes pick the wrong explanation. For real-world knowledge, it is difficult to avoid such situations. As an example, intent interpretation accurately explained operators' actions 90% of the time (see Geddes and Hoshstrasser (1989)). But what happens in those 10% of cases in which an action is incorrectly explained? Does the planner become hopelessly lost because it does not know the correct situation? If the knowledge is designed well, subsequent intent inferences can correct previous mistakes. If script A was selected instead of script B, but a further operator action is only explained by script B, script A is revoked and the mistake overcome.

In the knowledge design process, situations where incorrect inferences are possible should be identified. The consequences of an incorrect inference need to be evaluated. If these consequences can lead to serious errors, a knowledge design problem exists, and the PGG should be reorganized.

Continuing the driving example, if the driver applies the brakes at a stop sign, then the intent interpreter takes this action to be a step in a script for stopping the car and activates this script (which has as a next step to turn off the engine), disastrous results ensue. This poor design can be corrected by simply removing the next step in the script; in this case it is better to restrict the intent interpreter's capability to understand the situation.

Error Monitoring

Error monitoring involves evaluating the consequences of operator actions to determine if there are adverse consequences. If a driver has accelerated to 70 mph, and the associate knows the speed limit is 40 mph, it can warn the operator of a potentially dangerous (or costly) situation. In general, if the operator's action has dangerous consequences, it is flagged as an error. In contrast, because an action cannot be explained does not imply necessarily that it is an error; the intent interpreter may not have access to sufficient information to explain

the action. For further discussion of error monitoring see Hammer and Geddes (1987).

Following the operator's lead

One way to avoids conflicts between the operator and the planner is to let the operator take the lead at all times. The planner would not generate any actions to be performed by the system. This would still permit the associate system functions of error monitoring and information management (see Howard, Hammer, and Geddes (1988) for a discussion of information management).

Dealing with individual differences between operators

Suppose Joe likes to take back roads on his trips. His driving associate has been built so that it gives him route plans that use these back roads. But one day Bill tries out this driving associate software and it drives him nuts because it never picks the interstate routes. He ends up overriding the associate every time, and so the associate becomes useless. The general problem is this: how do you accommodate differences between operators?

One way of dealing with this is to identify key parameters associated with each planner decision. A set of user preferences is then made available via a user interface. This, as it turns out, is not much different than traditional software applications.

Another method would be for the associate to learn operator preferences. This could use an algorithm similar to the intent interpretation algorithm. Given operator selection of plans, what set of system preferences is compatible with those selections? For example, Bill picks a route from point A to point B that uses interstates maximally. There are alternative routes that use back roads. Therefore the associate infers that Bill 'prefers interstates' and stores that fact in Bill's user preferences.

A more difficult situation occurs for multi-operator, multi-vehicles scenarios. For example, in VTI the battle team is envisioned to have two operators in a single manned vehicle controlling as many as ten unmanned vehicles. The operators can divide up control in any manner they deem appropriate. But what if each operator has different preferences?

If a set of user preferences is defined for each operator, one operator can be "in charge," and so his or her preferences will be used. Clearly this is inadequate for plans and goals that the "subordinate" operator is managing. What is needed is for the associate to know, for a given plan, which operator is responsible for it. If the intent interpreter inferred the plan, the operator that performed the action that lead to activation of the plan can be marked as the responsible operator. If one uses the rule that all children of the plan inherit its responsible operator, then that operator's preferences can be used with respect to the child plans and goals. Another means of assigning a responsible operator is by having the operator who accepts a plan proposal be assigned the responsible operator for it and all its children (unless a child plan is proposed and accepted by a different operator). These two methods provide almost full coverage concerning a responsible operator. Only high-level plans that are automatically accepted are not covered. But by making an initial default designation of the responsible operator, full coverage can be obtained. See Geddes (1986) for additional research concerning individual differences and intent interpretation.

Display of Plan Proposals

There are two sets of additional information that can be provided with a planner's proposal. First, alternative plan proposals can be presented. Second, rationale for picking a particular plan and its associated parameters can be displayed.

Display of alternate proposals can to some degree help with the problem of user preferences. If Jane tends to disagree with the planner's decision concerning the best plan, Jane can simply select her favorite from the list of alternatives. An even more flexible method, however, is to present the plan proposal in a dialog box that lists all the appropriate parameters and allows modification of those parameters. The user could either modify specific parameters or, by selecting an alternative plan, have those parameters adjusted in the dialog box. A more complex situation occurs when part of the plan includes information best represented graphically, such as a route. There needs to be some means of connecting the graphical information with the dialog box information. A plausible option is to highlight or flash the related graphical information when the dialog box pops up and/or have a button on the dialog that does this.

Displaying rationale for the proposal is a more complex problem. In traditional rule-based systems, one might try displaying the chain of logic that lead to the proposal. Likewise, one might show the PGG parent instances that lead to the proposal. But for typical users, who probably don't know what a PGG is, or at least how it works, such information is unlikely to provide value. A simple "plain English" description of a decision algorithm the planner uses would allow users to have some idea of the rationale for the proposal. Since decision algorithms are typically originally described in plain English by a domain expert (e.g., how a driver decides what route to take), this description could be provided to the user upon request

(input parameter values used by the algorithm might also be included).
A further consideration involves when to display rationale and/or alternatives. There are three factors that would aid in determining this: the nature of the particular plan being proposed, the context in which the plan is being proposed, and the time criticality of the situation.

As users become familiar with the associate system, providing rationale for a particular plan may be useless; the user already knows what the planner is doing. For example, if a planner is providing a route that makes the most use of interstates, there is no useful rationale to be given. Likewise, would it be of any benefit to display an alternate route that made slightly less use of interstates?

Time factors can also influence whether or not to display rationale or alternatives. In a highly dynamic environment, there won't be enough time for the operator to ponder the rationale used by the associate.

The associate needs to be able to identify situations when displaying additional information is not beneficial. Further, it should be able to know when not to display the proposal at all. For example, if the anti-lock brakes on a car have engaged and the gas tank is getting low, it is not a good time to propose stopping at the nearest gas station. This implies that knowledge concerning the overall status of operations needs to be maintained and that plans should have a criticality factor assigned to them to prevent situations such as the one described above.

As is the case with issues regarding operator preferences, multi-operator situations require careful knowledge engineering. One way of handling the situation is to display all proposals to each operator. The problem here is that operators will be distracted by proposals for which they are not concerned. Another is to only display proposals to the operator designated as responsible for that plan. There is a problem here, too. Once a plan has been assigned to an operator, and the assignment of responsibility algorithm describe above is used, the other operator will see no proposals for any of the child plans (unless that operator performs an action which leads the intent interpreter to activate one of these plans). It is almost an "all or nothing" situation. A compromise is to use the criticality factors discussed above. Rules such as the following can be used: if the non-responsible operator is not busy, go ahead and display all plan proposals to him or her. Otherwise, show only those proposals whose plans have been assigned to that operator. As above, overall status of the system must be assessed so that it can be determined if the operator is busy.

Summary

In the associate work done by Applied Systems Intelligence, Inc. and others over the past 14 years, many of the problems concerning planner-intent interpreter interaction have been dealt with including: knowledge representation issues, asymmetries in planning and intent interpretation, and error monitoring. At this stage, it is a matter of refining the overall process of building associate systems. What have been outlined here are various methods of doing this: methods for handling operator differences and plan proposals. Future work in associate systems in areas such as unmanned systems and supply chain management will provide a testbed for these new methods as well as an opportunity to refine the old ones.

References

Geddes, N.D. and Lizza, C.S. (2001) Practical Applications of a Real Time, Dynamic Model of Intentions. Intent Inference for Collaborative Tasks. Papers from the 2001 AAAI Fall Symposium. Benjamin Bell and Eugene Santos, Cochairs. Technical Report FS-01-05.

Elmore, W., Dunlap R., Campbell R. and Perkins, D. (2000) Intelligent Control of Automated Vehicles: A decision Aiding Method for Coordination of Multiple Uninhabited Tactical Aircraft. Final Report for DARPA Order No. G178 under contract #MDA972-98-C-0011.

Geddes, N.D. and Lizza, C.S. (1999) Shared plans and situations as a basis for collaborative decision making in air operations. 1999 World Aeronautics Conference, Anaheim, CA SAE Paper 1999-01-5538

Geddes, N. D. (1997) Associate Systems: a framework for human-machine cooperation, *Human Computer Interactio*n, San Francisco September 20-23, 1997.

Geddes, N.D. and Hoshstrasser, B.H. (1989) Operator Intent Inferencing for Intelligent Operator support Systems. *Proceeding of the IJCASI-89 Workshop on Integrated Human-Machine Intelligence in Aerospace Systems.*

Geddes, N.D. (1989) *Understanding Human Operator's Intentions in Complex Systems* (Doctoral Dissertation, Georgia Institute of Technology), Atlanta, GA

Hammer, J.M. and Geddes, N.D. (1989) Design of an Intelligent Monitor for Human Error in a Complex System. *Proceedings of the 1987 American Institute of Aeronautics and Astronautics Computers in Aerospace Conference.* Boston, MA.

Howard, C.W., Hammer, J.M. and Geddes, N.D. (1988). Information Management in a Pilot's Associate. *Proceedings of the 1988 Aerospace Applications of Artificial Intelligence Conference*, 1, 339-349.

Geddes, N.D. (1986). The Use of Individual Differences in Inferring Human Operator Intentions. *Proceedings of the Second Annual Aerospace Applications of Artificial Intelligence Conference*, Dayton, October 1986.

Geddes, N.D. (1985) Intent inferencing using scripts and plans. *Proceedings of the First Annual Aerospace Applications of Artificial Intelligence Conference,* 160-172.

Schank R. S. and Riesbeck, C. (1979) *Inside Computer Understanding*, Lawrence Earlbaum Associates, Hillsdale, N.J. 1979

Schank, R.S. and Abelson, R. (1977) *Scripts Plans Goals and Understanding*. Lawrence Earlbaum Associates, Hillsdale, NJ 1977.