Towards an Intelligent Planning Knowledge Base Development Environment

Steve Chien
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 525-3660
Pasadena, CA 91109-8099
chien@aig.jpl.nasa.gov

Abstract
This abstract describes work in developing knowledge base editing and debugging tools for the Multimission VICAR Planner (MVP) system. MVP uses Artificial Intelligence (AI) Planning techniques to automatically construct executable complex image processing procedures (using models of the smaller constituent image processing subprograms) in response to image processing requests made to the JPL Multimission Image Processing Laboratory (MIPL).

1. Introduction

A major factor in determining the feasibility of applying AI planning techniques to a real-world problem is the amount of effort required to construct, debug, and update (maintain) the planning knowledge base. Yet despite the criticality of this task, relatively little effort has been devoted to developing an integrated set of tools to facilitate constructing, debugging, and updating specialized knowledge structures used by planning systems.

This paper describes two types of tools developed to assist in developing planning knowledge bases - static analysis tools and completion analysis tools. Static analysis tools analyze the domain knowledge rules and operators to see if certain goals can or cannot be inferred (clearly such checks must be incomplete for reasons of tractability). Static analysis tools are useful in detecting situations in which a faulty knowledge base causes a top-level goal or operator precondition to be unachievable - frequently due to omission of an operator effect or a typographical error. Completion analysis tools operate at planning time and allow the planner to complete plans which can achieve all but a few focussed subgoals or top-level goals. Completion analysis tools are useful in cases where a faulty knowledge base does not allow a plan to be constructed for a problem that the domain expert believes is solvable. In the case where the completion analysis tool allows a plan to be formed by assuming goals true, the domain expert can then be focussed on these goals as preventing the plan from being generated.

The static analysis and completion analysis tools have been developed in response to our experiences in developing and refining the knowledge base for the Multimission VICAR Planner (MVP) (Chien 1994a, 1994b) system, which automatically generates VICAR image processing scripts from specifications of image processing goals.

Section 2 describes the application area, automated image processing in the VICAR image processing language. Section 3 describes the Multimission VICAR Planner (MVP) system for automating image processing, and describes the types of knowledge representations used by MVP. Section 4 describes two types of tools we have developed to assist in developing and debugging planning knowledge bases developed for MVP: static analysis tools and completion analysis tools.

2. VICAR Image Processing

Currently, a group of human experts, called analysts, receive written requests from scientists for image data processed and formatted in a certain manner. These analysts then determine the relevant data and appropriate image processing steps required to produce the requested data and write an image processing program in a programming language called VICAR (for Video Image Communication and Retrieval[1]) (LaVoie et al 1989).

Unfortunately, this current mode of operations is extremely labor and knowledge intensive. This task is labor intensive in that constructing the image processing procedures is a complex, tedious process which can take up to several months of effort. There are currently tens of analysts at MIPL alone whose primary task is to construct these VICAR programs. Many other users at JPL and other sites also write VICAR scripts, with the total user group numbering in the hundreds.

This paper describes work performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Other past and present members of the MVP team are Helen Mortensen, Christine Ying, Shouyi Hsiao, Alex Gray, Joe Nieten, and Jean Lorre.

[1] This name is somewhat misleading as VICAR is used to process considerable non-video image data such as MAGELLAN synthetic aperture radar data.

The VICAR procedure generation problem is a knowledge intensive task. Constructing VICAR procedures requires diverse types of knowledge such as knowledge of:

1. image processing in general and VICAR image processing programs (as of 1/93 there were approximately 50 frequently used programs, some having as many as 100 options)
2. database organization and database label information to understand the state of relevant data
3. the VICAR programming language to produce and store relevant information.

Because of the significant amount of knowledge required to perform this task, it takes several years for an analyst to become expert in a VICAR image processing area.

3. Automated Image Processing Using MVP

The overall architecture for the MVP system is shown in Figure 1. The user inputs a problem specification consisting of processing goals and certain image information using a menu-based graphical user interface. These goals and problem context are then passed to the decomposition-based planner. The decomposition-based planner uses image processing knowledge to classify the overall problem type which the user has specified using *skeletal planning* techniques (Iwasaki & Friedland 1985). This classification is then used to decompose the problem into smaller subproblems using *hierarchical planning* techniques (Stefik 1981). During this decomposition process, MVP determines which information on the database state is needed by the planner to solve the subproblems. The subproblems produced by the decomposition process are then solved using traditional operator-based planning techniques (Pemberthy & Weld 1992), in which a planner uses a description of possible actions (in this case image processing steps) to determine how to achieve subproblem goals as indicated by the problem decomposition. The resulting plan segments are then assembled using constraints derived in the decomposition process. The resulting plan is then used to generate an actual executable VICAR PDF using conventional code-generation techniques.

MVP uses both decomposition and operator-based planning paradigms for two reasons: search control and user understandability. Plans in the MVP domain can be of considerable length (up to 100 steps) and each step (or VICAR program) can involve reasoning about numerous complex effects (many operators have tens of effects). Due to the large search space caused by this complexity, conventional operator-based planning approaches are not able to tractably construct plans in the VICAR domain without significant control knowledge. By using the decomposition planning paradigm, MVP breaks up the large search space planning problems caused by the complexity of the image processing problems in to multiple smaller problems, thus reducing the search problems encountered during operator-based planning. Indeed, the problem decomposition rules used in MVP can be considered a very important form of search control knowledge essential to MVPs image processing capability.

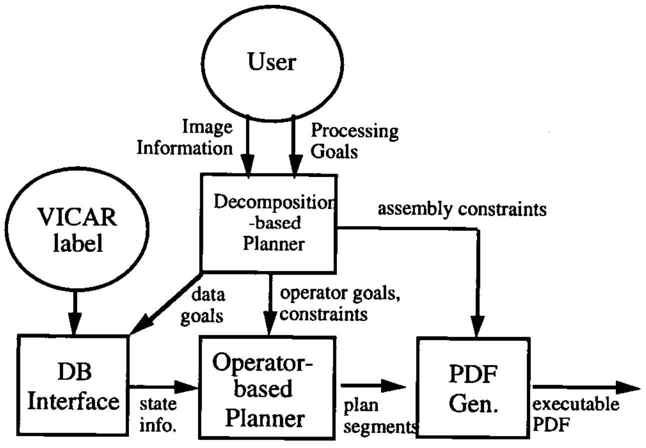

Figure 1: MVP Architecture

MVP also uses decomposition-based planning for reasons of user-understandability. Even if a purely operator-based planning approach were able to generate plans to solve the VICAR problems, these plans would be difficult for MIPL analysts to understand because MIPL analysts do not consider an entire image processing problem all at once. Typically, analysts begin by classifying the general problem being addressed into one of a general class of problems, such as mosaicking, color triple processing, etc. They then use this classification and the problem context to decompose the plan into several abstract steps, such as local correction, navigation, registration, touch-ups, etc. Because MVP uses decomposition-based planning to reduce the original image processing problem, it can easily produce an annotated trace of how the overall problem was classified and decomposed, simplifying analyst understanding of the plan generation process.

MVP uses decomposition-based (Lansky 1993) planning techniques to implement skeletal and hierarchical planning and traditional operator-based (Pemberthy & Weld 1992) planning paradigms to construct VICAR image processing scripts. Consequently, MVP uses two main types of knowledge to construct image processing plans (scripts):

1. decomposition rules - to specify how problems are to be decomposed into lower level subproblems; and
2. operators - these specify how VICAR programs can be used to achieve lower level image processing goals (produced by 1 above). These also specify how VICAR programs interact.

These two types of knowledge structures are described in further detail below.

3.1 Skeletal and Hierarchical Planning in MVP

MVP uses a decomposition-based approach (Lansky 1993) to perform Skeletal (Iwasaki & Friedland 1985) and

Hierarchical planning (Stefik 1981). In a decomposition-based approach, decomposition rules dictate how in plan-space planning, one plan can be legally transformed into another plan. The planner then searches the space plans defined by these decompositions. Decomposition-based approaches are extremely powerful in that many other paradigms (such as modal truth criterion planning can be implemented in a decomposition-based approach (Lansky 1993). Syntactically, a decomposition rule is of the form:

LHS		RHS
G_I = initial goal set/actions		G_R. = reduced goal
set/actions		
C_0 = constraints	==>	C_1 = constraints
C_2 = context		N = notes on decomposition

This rule states that a set of goals or actions G_I can be reduced to a new set of goals or actions G_R. if the set of constraints C_0 is satisfied in the current plan and the context C_2 is satisfied in the current plan provided the additional constraints C_1 are added to the plan.

Skeletal planning in MVP is implemented in by encoding decomposition rules which allow for classification and initial decomposition of a set of goals corresponding to a VICAR problem class. The LHS of a skeletal decomposition rule in MVP corresponds to a set of conditions specifying a problem class, and the RHS specifies an initial problem decomposition for that problem class. For example, the following rule represents a decomposition for the problem class mosaicking with absolute navigation.

LHS	RHS
G_I mosaicking goal present	G_R = 1. local correction,
C_0= null	2. navigation
C_2= an initial classification	3. registration
has not yet been made	4. mosaicking
	5. touch-ups
	C_1 = these subtasks be performed in order 1. 2. 3. 4. 5. protect local correction until mosaicking
	N = the problem class is mosaicking

This simplified decomposition rule states that if mosaicking is a goal of the problem and an initial problem decomposition has not yet been made, then the initial problem decomposition should be into the subproblems local correction, navigation, etc. and that these steps must be performed in a certain order. This decomposition also specifies that the local correction goals must be protected during the navigation and registration processes.

In general, MVP permits goals and abstract steps to be specified in the G_I & G_R fields. The constraints C_0 & C_1 may be ordering and codesignation constraints and the context may specify the presence or absence of attributes over the plan or goals (such as a certain goal not being present, etc.).

MVP also uses decomposition rules to implement hierarchical planning. Hierarchical planning (Stefik 1981) is an approach to planning where abstract goals or procedures are incrementally refined into more and more specific goals or procedures as dictated by goal or procedure decompositions. MVP uses this approach of hierarchical decomposition to refine the initial skeletal plan into a more specific plan specialized based on the specific current goals and situation. This allows the overall problem decomposition to be influenced by factors such as the presence or absence of certain image calibration files or the type of instrument and spacecraft used to record the image. For example, geometric correction uses a model of the target object to correct for variable distance from the instrument to the target. For VOYAGER images, geometric correction is performed as part of the local correction process, as geometric distortion is significant enough to require immediate correction before other image processing steps can be performed. However, for GALILEO images, geometric correction is postponed until the registration step, where it can be performed more efficiently.

This decomposition-based approach to skeletal and hierarchical planning in MVP has several strengths. First, the decomposition rules very naturally represent the manner in which the analysts attack the procedure generation problem. Thus, it was a relatively straightforward process to get the analysts to articulate and accept classification and decomposition rules for the subareas which we have implemented thus far. Second, the notes from the decomposition rules used to decompose the problem can be used to annotate the resulting PDF to make the VICAR programs more understandable to the analysts. Third, relatively few problem decomposition rules are easily able to cover a wide range of problems and decompose them into much smaller subproblems.

3.2 Operator-based Planning in MVP

MVP represents lower level procedural information in terms of classical planning operators. These are typical classical planning operators with preconditions, effects, conditional effects, universal and existential quantification allowed, and with codesignation constraints allowed to appear in operator preconditions and effect conditional preconditions. For reasons of space constraints the operator representations are not described here (for a good description of a classical planning operator representation similar to ours see (Pemberthy & Weld 1992)).

3.3 Current Status of MVP

The current version of MVP is in use in the JPL Multimission Image Processing Laboratory (MIPL) and covers the image processing areas of radiometric correction, mosaicking with absolute navigation, and color triplet reconstruction. An expert analyst estimated that MVP reduces the effort for these tasks for an expert analyst from

1/2 a day to 15 minutes and for a novice analyst from several days to 1 hour.

In the current version, MVP 2.0, there are on the order of 40 decomposition rules (of these approximately 10 perform skeletal planning classification and 30 perform hierarchical decomposition). These decomposition rules cover on the order of hundreds of goal combinations and problem contexts. These decomposition rules are able to break down the script generation problem into several (typically 5) goal sets each of approximately 5 to 10 goals, where each goal set is typically achievable by a subplan of 10 operators or less. This size of subplan is easily handled by the operator-based planner with search of on the order of thousands of plans and can be constructed on the order of 10s of seconds for a Sparcstation 10. The operator-based planner for MVP version 2.0 uses approximately 45 planning operators to reason about approximately 30 VICAR programs, and models approximately 50 file attributes. Many of these operators are quite complex and contain 10-20 effects (many are conditional effects).

4. Knowledge Acquisition and Refinement in MVP

In order for MVP to be able automatically generate VICAR image processing procedures, the knowledge base for MVP must represent large amounts of knowledge in the form of decomposition rules and operators. Unfortunately, eliciting and encoding this knowledge is a tedious, time-consuming task. In order to facilitate this key process of knowledge acquisition and refinement we have been developing a set of knowledge-base editing and analysis tools. These tools can be categorized into two general types: (1) static knowledge base analysis tools; and (2) completion analysis tools. Because MVP uses two types of knowledge: decomposition rules and operator definitions, each of these tools can be used with each of these representations. We describe the capabilities of these tools below.

4.1 Static Analysis Tools for MVP

Static analysis tools are used to perform simple subgoal analyses to detect cases where a user has made an error in defining the preconditions or effect so as to make certain problem goals unachievable. The knowledge engineer in MVP is required to define a problem context, listing which predicates (or negations) may be true in the initial state and which predicates (or negations) may be given as part of the goal specification. By a straightforward analysis of the operators required preconditions and effects, a static analysis can detect simple cases in which goals cannot be achieved - such as when no operator has an effect to achieve a goal, when the only operator to achieve a goal has a precondition which is not listed as an initial state condition of the effect of any operator, and other similar cases. While these cases may seem rather obvious and easy for the user to catch, for domain descriptions of even moderate size, errors such as these can be painful to manually track down. We are currently investigating using more sophisticated static analysis techniques to detect less obvious cases where goals are unachievable [Etzioni to appear, Ryu & Irani, 1992].

For example, a user might define a problem space with predicates G1, G2, and G3 as goals, predicates I1, I2, and I3 as initial state facts, and operator O1 with {effects / preconditions}is {G1 / F1 F2 I1}, operator O2 is {F1 / I1 I2} and operator O3 is { F2 / F3 I3} and no other operators have F2 as an effect. A straightforward static analysis reveals that there is no way of achieving the subgoal F2.

Similar static analyses can be performed on decomposition rule knowledge. In this case the user declares the: high level goals, initial plan state facts, and operational lower-level goals (to which the planner is trying to reduce the plan) that can be given and the system analyzes the decomposition rules to detect cases in which certain goals cannot be reduced into operational goals.

For example, a user might define a problem space with predicates G1, G2, and G3 as input goals, predicates OG1 OG2, OG3, and OG4 as operational goals, and decomposition rules {LHS / RHS}: R1 { G1 G3 / OG1 G4}, R2 { G4 / OG2 OG3}, R3 { G2 G5 / OG4}. In this case G2 cannot be decomposed because rule R3 requires G5 to be applicable. Static analysis of this rule set can detect this type of interaction.

4.2 Completion Analysis Tools in MVP

The second type of knowledge base development tool used in MVP is the completion analysis tool. In many cases, a knowledge engineer will construct a domain specification for a particular VICAR problem, test it out on known files and goal combinations. Two possible outcomes will occur. First, it is possible that the domain specification will produce an invalid solution, in this case it is not too difficult to use the inconsistent part of the solution indicate the flawed portion of the domain theory. The second possibility is that the domain specification fails to allow the desired solution. In this case, detecting the flawed part of the knowledge base is more difficult, because it is difficult to determine which part of the domain specification caused the desired output plan to fail.

Completion analysis tools directly address this problem. The completion analysis tools allow the decomposition or operator-based planner to construct a proof with assumptions that an extremely limited number of goals or subgoals can be presumed achievable (typically only one or two). By seeing which goals if assumable, make the problem solvable, the user can sometimes focus more quickly on the flawed portion of the knowledge base. In the operator-based planner, completion analysis is permitted by adding another goal achievement method which corresponds to assuming that the goal is magically achieved. When the planner exceeds resource bounds after finding a number of solutions, these solutions are then reported back to the user to assist in focussing on possible areas of the domain theory for refinement.

For example, a user might define a problem space with predicates G1, G2, and G3 as goals, predicates I1, I2, and I3 as initial state facts, and operator O1 with {effects /

preconditions}is {G1 / F1 F2 I1}, operator O2 is {F1 ~F2 / I1 I2} and operator O3 is { F2 ~F1 / F3 I3} and no other operators have F1 or F2 as an effect. In this case, operators O2 and O3 are incompatible, and thus cannot be used to achieve the preconditions of O1. O2 undoes the needed effect of O3 (F2) and O3 undoes the needed effect of O2 (F1) (a so-called double-cross). In this case the planner will be unable to complete a normal plan. However, with the completion analysis tool, it is possible to find a plan which presumes achievability of either F1 or F2 (depending on the exact search strategy used one or both will be found). In this case, the user has been provided with the additional information that focussing on the domain theory relevant to the subgoals F1 and F2 will indicate the domain theory flaw. While some of these cases could be caught via a static analysis of the planning operators, our general approach has been to keep the static analysis to a quick and dirty process, that can be performed so quickly at operator definition time so as to go unnoticed by the user (except in cases where it detects inconsistencies). In cases where the user has reason to believe there are flaws in the domain theory which prevent construction of supposedly feasible plans, the completion analysis tools can help to focus user attention.

In the decomposition-based planner, the ordering of the goals is not relevant, so that the termination condition for the decomposition is modified to allow some number of non-operational goals to be present in the solution. For example, with a problem space with predicates G1, G2, and G3 as input goals, predicates OG1 OG2, OG3, and OG4 as operational goals, and decomposition rules {LHS / RHS}: R1 { G1 G3 / OG1 G4}, R2 { G4 / OG2 OG3}, R3 { G2 ~OG3 / OG4}. In this case G2 cannot be decomposed because R3 and R2 are incompatible. Completion analysis tools would allow for completion of the plan decomposition, but would indicate that G4 or G2 was assumed to be operational to produce the plan.

The main drawback of the completion analysis tools is that they dramatically increase the size of the search space. Thus, with the completion analysis tools, the user can specify that only certain types of predicates can be presumed true, or predicates relating to certain operators. This has been fairly effective in focussing the search. Unfortunately, we have as of yet not been able to determine any good heuristics for controlling the use of these tools in a more tractable way. However, in their current form, the completion analysis tools have proved quite useful in debugging the MVP radiometric correction and color triplet reconstruction knowledge base.

4.3 Future Work

One area for future work is development of explanation facilities to allow the user to introspect into the planning process. Such a capability would allow the user to ask such questions as "Why was this operator added to the plan?" and "Why is this operator ordered after this operator?", which can be answered easily from the plan dependency structure. More difficult (but also very useful) questions are of the form "Why wasn't operator O2 used to achieve this goal?" or "Why wasn't this problem classified as problem class P?". We are currently investigating using completion analysis tools to answer this type of question.

5. Discussion

This paper has described two classes of knowledge base development tools being developed for use in constructing and maintaining image processing knowledge bases for the MVP planning system. Static analysis tools allow for efficient detection of certain classes of unachievable goals and can quickly focus user attention on the unachievable goals. Completion analysis tools allow the user to quickly focus on which goals (or subgoals) are preventing the planner from achieving a goal set believed achievable by the knowledge base developer.

References

(Chien, 1994a) S. Chien, "Using AI Planning Techniques to Automatically Generate Image Processing Procedures: A Preliminary Report," Proceedings of the Second International Conference on AI Planning Systems, Chicago, IL, June 1994, pp. 219-224.

(Chien, 1994b) S. Chien, "Automated Synthesis of Complex Image Processing Procedures for a Large-scale Image Database," Proceedings of the First IEEE International Conference on Image Processing, Austin, TX, November 1994.

(Etzioni, to appear) O. Etzioni, "Acquiring Search Control Knowledge via Static Analysis," Artificial Intelligence, to appear.

(Iwasaki and Friedland, 1985) Y. Iwasaki and P. Friedland, "The Concept and Implementation of Skeletal Plans,"Journal of Automated Reasoning 1, 1 (1985), pp. 161-208.

(Lansky, 1993) A. Lansky, "Localized Planning with Diversified Plan Construction Methods," Technical Report FIA-93-17, NASA Ames Research Center, June 1993.

(LaVoie et al. 1989) S. LaVoie, D. Alexander, C. Avis, H. Mortensen, C. Stanley, and L. Wainio, VICAR User's Guide, Version 2, JPL Publication D-4186, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 1989.

(Pemberthy & Weld, 1992) J. S. Pemberthy and D. S. Weld, "UCPOP: A Sound Complete, Partial Order Planner for ADL," Proc. of the Third International Conference on Knowledge Representation and Reasoning, October 1992, pp. 103-114.

(Ryu & Irani, 1992) K. Ryu & K. Irani, "Learning from Goal Interactions in Planning: Goal Stack Analysis and Generalization," Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, CA, 1992, pp. 401-407.

(Stefik, 1981) M. Stefik, "Planning with Constraints (MOLGEN: Part 1)," Artificial Intelligence 16,2(1981), pp. 111-140.