Reuse of Plans as a Tool for Development of Remote Sensing Expert Systems

D. Charlebois[§], D. Goodenough[+], S. Matwin[§], M. Robson[*†], K. Fung[*]

[§]Department of Computer Science, University of Ottawa, Ottawa, Canada

[+]Pacific Forestry Centre, Forestry Canada, Victoria, Canada

[*]Canada Centre for Remote Sensing, Ottawa, Canada

[†]Intera-Kenting, Ottawa, Canada

ABSTRACT

Systems involving remote sensing analysis for satellite data in combination with geographic information systems are large and complex. The Canada Centre for Remote Sensing (CCRS) has created an expert system shell and several expert systems in order to provide image analysis programs with the necessary knowledge to solve difficult image processing problems, such as updating forest inventory geographic information systems. An Interactive Task Interface (ILTI) provides an expert system with a Prolog module designed to answer queries from an image analysis program by retrieving knowledge from an image analysis knowledge base. Image analysis experts currently create ILTIs. They have found this to be a time consuming task. We have developed a planner that creates plans that emulate an ILTI's behavior by analyzing image processing session dialogs between a human expert and an image analysis program. The planner relies on a knowledge base in order to generalize and modify plans acquired from session dialogs.

INTRODUCTION

Existing remote sensing (RS) software libraries often represent considerable investment on behalf of organizations that have developed them. Moreover, such libraries require considerable computer skills to be applied by end-users. By using the Artificial Intelligence paradigm we aim to bridge the gap between non-programmer end-users and the large repositories of domain knowledge encoded in FORTRAN. The Analyst Advisor Expert System developed at CCRS [2] is an example of this approach. The knowledge encoded in the Analyst Advisor can be conceptually classified into three categories: 1) factual knowledge: what imaging data are required to solve a specific RS problem (e.g.: device on which to calculate image gradients), 2) control or procedural knowledge: what lower-level expertise is required to process the knowledge and in what sequence (e.g.: perform maximum likelihood analysis for supervised classification after having selected training sites), and 3) actual domain knowledge: what are the computational processes required to provide the specific domain expertise (e.g.: the code to compute maximum likelihood probabilities on areas of pixels). The Analyst Advisor uses the RESHELL ES shell to represent the factual and control knowledge, and relies on the FORTRAN code contained in LDIAS (Landsat Digital Image Analysis System [2]) for the domain knowledge. Consequently, there is a need to interface between Prolog (and frame structures that represent factual and control knowledge) with the FORTRAN code that implements the domain experts. In the existing Analyst Advisor system, the interface was implemented by means of ILTIs (Interactive LDIAS Task Interface). An ILTI is a PROLOG program which handles the I/O requirements of both the expert RESHELL modules and the associated LDIAS programs. Each time a new expert which makes use of LDIAS procedures is developed, an ILTI for it has to be programmed from scratch. The task of developing of an ILTI is non-trivial and requires specialized knowledge of representation mechanism of RESHELL on one hand, and familiarity with PROLOG procedures that handle I/O requests for different types of I/O devices on the other hand.

This paper presents continuation of work described in [1] and introduces the use of planning to create ILTIs from examples of the dialogue between an image analyst and an LDIAS program. The approach allows an expert to train the planner by presenting it with a number of typical execution runs of an LDIAS task. The planner creates a plan which integrates all the steps that have occurred in the example runs [8]. The plan is subject to a knowledge-based generalization, so that the resulting generalized procedure will provide an interface for an entire class of applications of a given task. This class will re-use sections of plans given to the planner during training, and extend those plans by simple analogical reasoning. The system that we have developed is dubbed LEAR (LEarning Advisor Rules).

To summarize, LEAR relies on the following AI techniques to build RESHELL interfaces:

- Exemplary Programming, i.e. the methodology which attempts to build programs from examples of their I/O behavior [8],
- Planning, i.e. the methodology which attempts to build sequences of actions (referred to as *microplans*) that, when executed, guarantee the achievement of prescribed goals [3],
- Machine Learning, the methodology which provides for goal-oriented and knowledge-oriented generalization of experience, acquired from individual cases [4].

The next section describes the current methodology of interface development, and presents the parameters of the problem. We then briefly discuss the planning and the learning component of our system. Subsequent section contains the initial results obtained from experimentation. We conclude by discussing the possible evaluation of the system, and outlining future work.

91-72810/92$03.00 © IEEE 1992

ILTI DESIGN

An ILTI is a knowledgeable interface between a user and an LDIAS program. Its function is to provide answers to the LDIAS program prompts by consulting a knowledge base or, if no answer is found, by questioning a user. Consequently, an ILTI can be viewed as a program that, when executed, realizes a certain plan. Elements of this plan perform I/O actions consulting a knowledge base or, alternatively, accepting input from a user. Continuing with this planning perspective, execution of an ILTI corresponds to the execution of a conditional plan. The actual form of a plan is decided by the user input and the history of the plan so far. It becomes clear form the above model that, given the dependence of the next plan execution step on the input and on the history of the plan development so far, a finite state machine is the appropriate framework in which to represent plans for our purposes.

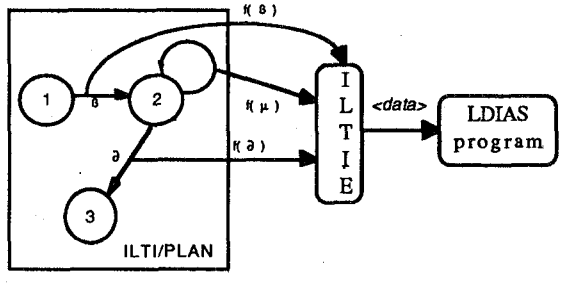

Figure 1: A plan

Figure 1 illustrates how a plan functions. The expert system that activates the plan's execution is not shown. A plan would normally start in an initial state which should agree to the first prompt from the corresponding LDIAS program. For each state, there exists a micro-plan (similar to operators in traditional planning terminology) that computes the input that fires a state transition (e.g.: ß). Once the state transition has been fired, the output from the state transition (e.g.: f(ß)) is sent to the waiting LDIAS program through the ILTIE (ILTI-Engine: a PROLOG module that implements communications between RESHELL and an LDIAS program) as the answer to the prompt. In short, for each LDIAS program there exists a plan that can satisfy different goals given input to state transition. Each state is associated to a specific prompt. Input to states fire transitions that produce output to answer prompts and lead to new states.

Three major design considerations must be addressed before any details of the general design can be laid out. First, contrary to most planning environments, this planner does not have prior knowledge of possible states for an LDIAS program, nor does it know all the legal transitions from a given state. In this sense, the planner must be adaptive and let the user guide it through a new sequence of state transitions or path (an important assumption here is that a new path provided by the user and the LDIAS program is always legal).

Second, the planner must work in an incremental fashion since one execution does not provide the planner with all possible paths through the program. In fact, it displays an unambiguous, single path. For most LDIAS software, several paths can exist and the planner must be able to add new paths to the existing one(s) incrementally. Given an existing ILTI, the planner should supervise a user dialog with the LDIAS program and react when the answer to a prompt does not match those that have previously been recorded for the ILTI. The result of this action would be to add, incrementally, the appropriate predicates that would handle the new path.

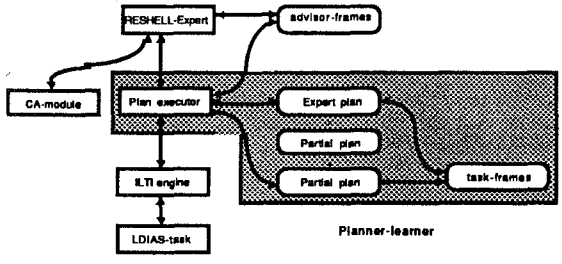

Figure 2: General LEAR design

Figure 2 introduces a new design where the components were defined to facilitate the use of a planner. The RESHELL-expert's task is to start an LDIAS program as well as to initialize the knowledge base to provide the program with the appropriate data to satisfy a given goal. The CA-module (Condition/Action module) supplies the expert system with predicates that perform tasks which are difficult for expert system rules. The ILTI-engine provides the same functionality as described previously.

The Plan-executor supervises plan execution to ensure proper behavior. A plan is a sequence of actions that supply the appropriate information to an LDIAS program given a particular request. Partial-plans are parts (subsequences) of plans destined for reuse by the same plan as well as other plans.

THE PLANNER

The planner has three essential components (figure 3):

- the planner module,
- the planner knowledge base:
  - plans,
  - partial plans,
  - micro-plans,
- the RESHELL knowledge base:
  - advisor frames,
  - planner taxonomy.

The planner module is an adaptive/incremental planner that reacts to new interactions between a domain expert (e.g.: an image analysis expert/researcher) and an LDIAS program. A new interaction is a dialog between the domain expert

and an LDIAS program that the planner has not seen previously and, thus has not provided for in a plan. In these circumstances, the knowledge contained in the plan developed so far by the planner must be extended (new alternatives must be added to the plan).

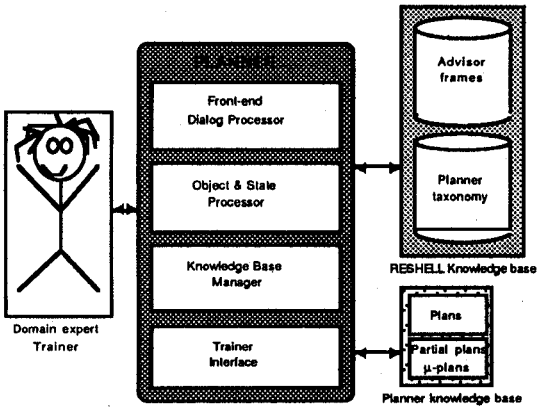

Figure 3: Planner Design

The LDIAS program execution through the use of a plan can best be expressed by:

set up LDIAS program parameters
start LDIAS program
repeat
    get prompt from LDIAS program
    process prompt and return answer
until normal disconnect or program abortion

Each iteration through this loop should place the plan in a new state. The state transitions are fired when an answer is provided by the frames knowledge base or by the domain expert.

Given this algorithm, two situations involve unrecognizable prompts:

- when working from scratch (i.e.: a plan did not exist for the LDIAS program at hand),
- when no answer can be found in the frames knowledge base or when the domain expert replies with an unknown answer (one for which a state transition does not exist).

The planner must recognize the occurrences of these events and handle them in the appropriate manner.

Detecting when the system is working from scratch is trivial and can be done in different ways, the simplest being to check for the presence of a plan dedicated to the LDIAS program at hand.

However, detecting if the domain expert replies with an unknown answer will require more extensive processing. The planner must:

- process the prompt and identify the object(s) the LDIAS program needs,
- process the objects, with help from the domain expert, to determine if the knowledge base contains knowledge of the objects at this stage of processing, or if they should be added to it,
- create a new state identifier and state transition predicate.

The task of processing the prompts and identifying the objects being manipulated is assigned to the Front-end Dialog Processor (FDP) (figure 3). Using simple pattern matching, the FDP can analyze each prompt and identify the actions and the objects used by the LDIAS program and place them in a symbol table. This will allow the planner to recognize the different contexts where objects are used.

Once the objects have been identified, the Object & State Processor (OSP) has essential tasks to accomplish. First the OSP must process the objects. If the objects in the LDIAS program prompt have not been recorded in the knowledge base or the planner taxonomy, the planner must determine:

- if an object is created and used only during the execution of the current LDIAS program,
- if an object should normally be output from another LDIAS program and used by the current program,
- if an object is created by the current LDIAS program and if it should be output and made available to other LDIAS programs.

The second task the OSP must perform is to trace the state transitions the plan executor fires. As known answers are given to prompts, the ILTI changes states. When an answer for which no legal transition exists, the OSP must create a new legal transition from the current state given the new input. The OSP must also determine which microplan is to be executed when the new transition is fired. It can be either an existing microplan, or one suggested by the planner, or one supplied by the human expert (trainer/user). This will trigger a dialog between the trainer and the planner. The planner will have to determine if the new transition leads to an existing state or if it leads to a new state.

If a new transition leads to an existing state, the planner simply has to add a new legal transition to the FSM since the initial state and final state for the new transition are known. Also, if the initial and final states of different transitions are the same, Machine Learning (ML) processing may be necessary to generalize and possibly merge the transitions.

If a new transition leads to a new state, the state transition must be added to the FSM and the process repeated.

Whenever the planner cannot determine the origin of an object, its use by other LDIAS tasks or its place in the planner taxonomy or the knowledge base, the trainer interface must present the problem to the domain expert in order to classify the object. The dialog between the domain expert and the planner will depend on the amount of knowledge the planner has gathered about the object.

Partial-plans and micro-plans are the main building blocks used by the planner. Partial-plans are sequences of state transitions that are designed to answer the needs of frequent occurrences of similar behavior. More specifically, LDIAS programs usually process objects that have the same general structure. Since these objects have the same attributes, gathering their particular values can be accomplished using the same subsequences of plans leading to the reuse of partial-plans. The most

prominent example is that most, if not all, LDIAS programs require image files as input or output. In these cases, a partial-plan that can build a file specification can be reused.

Results

The experimental goal was to determine the rate at which new micro-plans have to be defined as the number of image analysis programs on which the system is trained grows. For that purpose, we have used the planner on a sequence of four LDIAS programs: DCT, TRUGEN, GSTAT, and SEPAR. The order in which these programs were run is significant in that this sequence is necessary to produce and verify an image classification. Since this sequence of programs performs a specific task, it is to be expected that objects handled by the first programs will also be used by the others. In figure 4, on the y-axis, we have plotted the percentage of micro-plans that were reused by the planner while developing plans for a sequence of programs.

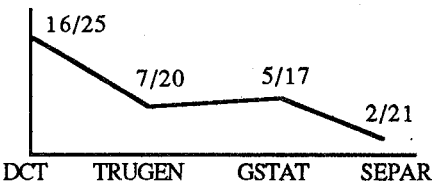

Figure 4: The percentage of new micro-plans

The numbers next to the data points on the graph represent the number of new micro-plans over the total number of micro-plans required to successfully execute the image analysis program. The graph shows that the ratio of new micro-plans that have to be created decreases as training of the system continues. This allows us to conclude that the basic objective of the planner development has been achieved: the more the system is trained, the less the system has to resort to the expert. In other words, the planner exhibits and enhances reuse of interface code, thereby minimizing the programming effort necessary for their development.

Conclusion

The paper shows a design of a case-based planner that applies machine learning techniques to the problem of automated re-use of specialized interfaces between the factual knowledge and the domain knowledge in a RS expert system. The prototype planner has been developed at CCRS. Introductory experimentation, reported in the previous section, indicates that there is significant potential for micro-plan reuse when a training session involves running a sequence of LDIAS programs that will accomplish a specific image analysis task. Furthermore, image analysts are relieved of having to master PROLOG programming since the LEAR system uses a graphics user interface that is well removed from PROLOG. Early experience indicates that even after a successful implementation, several important research issues remain to be addressed.

One such question is how best to train the planner. This is the problem of selection of training runs that will be processed by the proposed system. Should they all be similar?

Another question is what other learning techniques would produce interesting results in connection with the approach presented here. The area of Case-based Reasoning (CBR) [6] is potentially promising, and will be further investigated.

Yet another challenge for the learning component of LEAR is how, given more extensive knowledge of image analysis goals and the different objects that must be handled by LDIAS programs, the planner could implement (with some assistance) the procedures that would initialize the knowledge base and process the results.

Acknowledgments

The work discussed here was supported by the Natural Sciences and Engineering Research Council of Canada, as well as by the Canada Centre for Remote Sensing. The third author kindly acknowledges useful comments on this project from the staff of NASA Ames Research Center's AI Branch.

References

[1] D. Charlebois, J.-C. Deguise, D. Goodenough, S. Matwin, M. Robson A Case-based Planner to Automate Development of ES Software for Analysis of Remote Sensing Data, Procs. IGARSS-91, pp.

[2] Goodenough, D.G., Goldberg, M., Plunkett, G., Zelek, J., 1987, An Expert System for Remote Sensing, IEEE Transactions on Geoscience and Remote Sensing, Vol. GE-25, no. 3, pp 349-359.

[3] Hendler, J., Tate, A., Drummond, M., 1990, AI planning: systems and techniques, AI magazine, vol. 11 no. 2, summer 1990, p. 82.

[4] Michalski, R.S., 1983, Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann Publishers, pp82-129.

[5] Yoo, J. Fisher, D., 1991, Concept Formation over Explanations and Problem-solving Experience, Procs. of IJCAI-91, pp. 630-637, Sydney, Australia.

[6] Riesbeck, C.K., Schank, R.C., 1989, Inside Case-based Reasoning, Lawrence Erlbaum Associates, Publishers.

[7] Robson, M., Goodenough, D.G., Deguise, J-C., 1990, Automated Program Execution in a Hierarchical Expert System: RESHELL, Proceedings of the 23rd DECUS Canada Symposium.

[8] Waterman, D., Faught, W, Klahr, P., Rosenschein, S., Wesson, R., 1986, Exemplary programming: applications and design considerations, In Expert systems: techniques, tools and applications, edited by Klahr, P. and Waterman D., Addison-Wesley, pp 273-309.