Planning Applications in Image Analysis

Mark Boddy Jim White Robert Goldman
{boddy | jwhite | goldman}@htc.honeywell.com
Honeywell Technology Center, MN65-2100
3660 Technology Drive
Minneapolis, MN 55418

Nick Short, Jr.
short@dunloggin.gsfc.nasa.gov
ISTO, Code 930.4
NASA Goddard Space Flight Ctr.
Greenbelt, MD 20771

Abstract

We describe two interim results from an ongoing effort to automate the acquisition, analysis, archiving, and distribution of satellite earth science data. Both results are applications of Artificial Intelligence planning research to the automatic generation of processing steps for image analysis tasks. First, we have constructed a linear conditional planner (CPed), used to generate conditional processing plans. Second, we have extended an existing hierarchical planning system to make use of durations, resources, and deadlines, thus supporting the automatic generation of processing steps in time and resource-constrained environments.

1 Introduction

The collection, analysis and distribution of data resulting from NASA science missions is an increasingly daunting task. The National Space Science Data Center (NSSDC) responds to more than 2500 data requests from remote users in a single year [8]. As of 1990, NSSDC's archives included more than 6000 Gigabytes of digital data and 91 million feet of film. By 1995, the NSSDC is expected to contain 40,000 Gigabytes of digital data. Shortly thereafter, the satellites of the Earth Observing System (EOS) will come online, eventually adding new data at a rate of nearly 2000 Gigabytes per day, over an expected mission duration of 15 years [12, 6].

The EOS Data Information System (EOSDIS) is being designed and built to support the storage, analysis, and retrieval of data from this immense archive. Dozier [6] offers the following characterization:

> EOSDIS must allow scientists to easily and quickly acquire usable, understandable, timely data. "Timely" means "a reasonable period following the measurements" – one to two days after the observations, or up to a week for higher-level products. "Quickly" means minutes, not hours. "Easily" means that the user should not have to jump through many hoops to request the data.

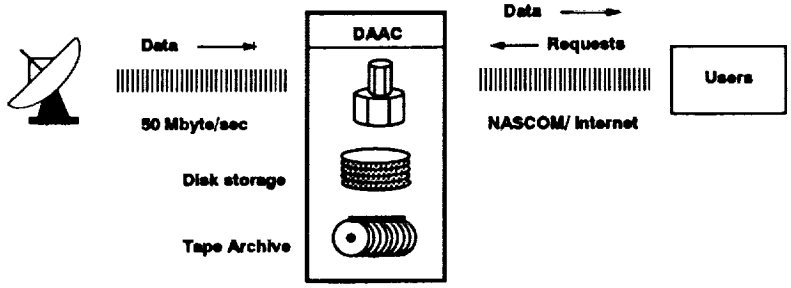

Figure 1: The EOSDIS domain

EOS data will be supplied by several different types of sensors and used by scientists in a variety of disciplines, most with no special knowledge of how EOS data is obtained or organized. The type of sensor from which the data was gathered will affect the processing necessary to render the data useful. The use to which the data is put will determine both the images retrieved (a geologist and an oceanographer will be interested in very different sets of data) and the analysis to which that data is subjected (e.g., topography vs. phytoplankton levels).

Raw and analyzed data will be stored in a distributed network of database sites, known as Distributed Active Archive Centers (DAACs). Also connected to the network will be a variety of special-purpose hardware that can be used for further analysis of either new or retrieved data. These analyses may consist of several steps (e.g., scan line removal, georegistration, or normalization for the incident angle of the sun), and will be run on a distributed network of heterogeneous machine types. Given the enormous amount of data involved, most of it will of necessity be stored off line. We anticipate hierarchical caches for data storage [2] with high-speed disks at the top of the hierarchy and tape archives at the bottom. Data will move up and down in this hierarchy for further analysis.

A high level concept of the resulting system is depicted in Figure 1. Data is received by any of several ground stations from any of a set of satellites, and transmitted to one or more of the archive centers, where it is analyzed as necessary (and as time permits), and then archived. Scientists interested in using the data may make requests that data be retrieved from one or more of the archives and analyzed further.

In both joint and separate work at NASA's Goddard Space Flight Center and the Honeywell Technology Center, we have been working on automating the acquisition, initial processing, indexing, archiving, analysis, and retrieval of satellite earth science data, with particular attention to the processing taking place at the DAACs.

In this paper, we present the results of ongoing work on planning for image process tasks in the EOSDIS *Product Generation System* (PGS). Section 2 presents the problem presented by PGS in additional detail. Section 4 describes the extension of a Nonlin style hierarchical planner to use information about deadlines, durations,

and resources. Section 3 is a dicussion of the use in image processing of *conditional plans:* plans including branching points dependent on the outcome of some earlier action (e.g., an observation of some type).

2 Data Management for Earth Science

NASA's role in the Mission to Planet Earth is the Earth Observing System (EOS) program and several smaller Earth science missions. These missions represent efforts to study the Earth's geosphere, biosphere, atmosphere, and cryosphere, as a system of interrelated processes by modelling surface temperature, ozone depletion and greenhouse effects, land vegetation and ocean productivity, and desert/vegetation patterns to name a few. With participation from the European Space Agency, Japan, Canada, and NASA, several platforms containing a multitude of sensors will be launched in the late 1990's, producing data that will be stored in geographically-oriented data systems such as the EOS Data and Information System (EOSDIS). In general, EOSDIS will manage the mission information, the data acquisition and distribution, the generation of scientific data products, and the interface to external systems.

Ensuring access to this information is a challenging task because of the daunting size of EOSDIS and the potential limitations of current technologies. Over its 15 year life, the Data Archival and Distribution System (DADS), a component of the EOSDIS, will eventually maintain around 11 petabytes [12].[1] While mass storage technology will solve some archiving problems [2], finding data will require new and innovative methods for users to effectively search the archives. The archives will include a variety data types including raster satellite images, ancillary vector/raster maps, derived spatial products from model simulations (e.g., output from global temperature models), and associated engineering and management textual data, suggesting that the archive and meta database will be both diverse and complex.

In current NASA scientific data systems, data are found by users who already know information related to the context of the satellite processing environment, such as the time of the satellite's observation, the satellite and sensor type, and location. This context-based metadata search forces the user to translate scientific needs into project specifications that often contain esoteric NASA nomenclature. A better solution, often called content-based metadata search, is to allow scientists to find data based upon their scientific interests within the imagery. Providing features based upon scientific interests for searching through a database assumes that a system can be created to interpret imagery with the skill of a scientist, yet with the speed of the computer. This automation has been the goal of many researchers in remote sensing, image processing, and computer vision for years; there is no known general solution to the problem.

[1] One petabyte is 10^15 bytes.

2.1 Opportunities for Automation

In this section, we describe the necessary functions for an automated planning system for image classification and indexing according to *browse products*. The entire range of functions described here are actively under development or investigation at this time. In the rest of the paper, we restrict ourselves to a discussion of the generation of plans for image analysis.

Despite the the lack of a general theory, computer-based photo interpretation operations for satellite/aerial imagery can be partially defined as file manipulation, calibration, reduction of the number of channels, image enhancement and correction, segmentation, and pattern classification (see figure 3). These operations often require an expert to "mix and match" the steps depending on the quality of the sensor, the format of the data, the properties of the sensing environment (e.g., atmospheric conditions, direction of sun illumination, etc.), availability of ancillary data such as topographic maps and ground truth observations, and the set of possible features within an image. Typically, the end result of this process is a map labelling pixels to classification categories from proven recognizable schemes for which the sensors were designed. Example schemes include: land use/land cover cloud cover type, vegetation cover, and soil type. While these examples refer to physical objects, properties such as temperature and aerosol content also constitute legitimate labels, only each label represents a range of continuous values. In EOS, much of the work of the PGS will be to recognize these features for processing at level 2 and above.

In the realm of automatic feature recognition, the planner is the component that optimizes accuracy as a function of the resource constraints. If there is a lot of available processing time due to a low incoming data rate, then the planner chooses the image processing sequence with the highest expected accuracy. If the data rate is high, then the planner constructs a sequence that either substitutes computationally cheaper, yet less accurate image processing steps for expensive operations, eliminates steps that can be deleted without a major loss, or uses a fixed-time default plan that implies an upper bound on the highest allowable data rate (e.g., ingest only file header information that comes with the raw data).

The planner must make choices regarding preprocessing steps and image classifiers as a function of the input image's header information, called ephemeris data. For example, suppose that an image from the Moderate-Resolution Imaging Spectrometer-Nadir (MODIS-N) sensor of EOS arrives with its areal extent over Washington D.C. Further suppose that after launch, MODIS-N produced scan lines such as LANDSAT MSS's "sixth-line striping," evidenced by horizontal banding within the images . Modis-N was designed to characterize surface temperature at 1-km resolution, ocean color, vegetation/land surface cover (e.g., leaf area index and land cover type, vegetation indices), cloud cover and properties, aerosol properties, and fire occurrence. Based upon this information, the planner constructs the sequence of steps by first stripping off the header file from the raw data, which indicates the time of observation, the sensor, sun angle and azimuth, location, and

file format. The planner then queries an online database to insert the new header information annotated with a unique image id and waits for known information to be returned related to the header, for a set of neural network weight files that have been created by training over similar conditions, and for any ancillary data files such as digital elevation data, ground truth, and hydrology maps. In this case, because the location of the image is over land, the set of recognizable features will include vegetation, cloud, and temperature classes, while it will exclude ocean related classes. Once the planner has the combined dynamic information of the header with the static knowledge about the sensor, it begins constructing the sequence or image processing plan.

Once the pixel labelling is completed, the planner must choose the form of browse product as a function of the amount of storage available and the importance of the image, as defined by priority. Ranging from low to high available storage, the browse product can be an image classification vector ICV, a "postage stamp" rendition of the classification map, a low resolution version of the classification map, or a classification map that is the size of the original image. Finally, the planner must ingest the browse product into the appropriate database with the associated header information and the sequence of processing steps used. If it is found later that a particular processing step was inadequate, then the meta database can be searched for all browse products containing that step in order to initiate reprocessing. Likewise, if a scientist, through his own analysis, determines that the classification accuracy was incorrect, then he can submit his changes, as well as methods, to the meta database administrators for update.

3 Conditional Analysis Plans

The automatic generation of plans for image analysis is a challenging problem. Preliminary processing (e.g., removal of sensor artifacts) and analysis (e.g., feature detection) involve a complex set of alternative strategies, depending in some cases on the results of previous processing. For example, detailed location of roads and rivers is only worth doing if there is evidence that those features are present in the image. Plans for image processing need to be *conditional*, in the sense that the course of action to be followed is dependent on the outcome of previous actions.

We have developed a conditional planner that advances the state of the art in several respects, including the use of regression in the generation of conditional plans and a careful treatment of the modelling of observations by permitting the specification of a proposition as true, false, or unknown. We have successfully applied our planner to the generation of conditional plans for image analysis in "EOS world" (named by analogy to the "blocks world"), a planning domain based on data analysis problems related to the Earth Observing System's Data and Information System (EOSDIS).

3.1 Motivation and Background

Classical planning has been criticized for its reliance on a complete model of actions [4]. Constructing an elaborate plan to achieve some set of goals makes little sense if the environment is sufficiently unpredictable that the plan is likely to fail at an early stage. There are several approaches to the problem of generating plans for use in a changing and uncertain world. These fall generally into three classes: making plans more robust in the face of changes in the environment [7], modifying plans as new information becomes available, [2] and conditional planning (more precisely, planning with conditional actions): planning which takes into account the uncertain outcomes of actions.

Conditional action planning is suitable for domains in which there is limited uncertainty and in which plans are constructed at a fairly high level of granularity. Preliminary indications are that planning for image analysis is eminently suitable. Robot planning is probably *not* such an application, unless it can be carried out at a level of abstraction sufficiently high that much of the uncertainty can be ignored.

Peot and Smith [11] have developed a non-linear planner for conditional planning. In conventional, "classical" planning applications, non-linear planning is usually an improvement over linear planning because fewer commitments yields a smaller search space, at a relatively minimal added cost to explore each element of that search space [10]. However, it is not clear that this tradeoff operates in the same way for conditional planners. Furthermore, the operation which is needed to properly construct branching plans — resolving clobberers through conditioning apart — is a very difficult operation to direct. Accordingly, a *linear* conditional planner may be a reasonable alternative.

We have developed a linear conditional planner, based on McDermott's regression planner PEDESTAL [9]. This planner has been implemented in Quintus Prolog, running on Sun SPARCstations. It has been tested on Peot and Smith's "Ski World" sample problem and on the simplified model of the EOSDIS image processing domain described above.

3.2 Action representation

Following McDermott, we represent actions in the plan library in terms of three predicates: `precondition`s, `add` lists and `delete` lists.[3] A precondition entry in the database looks as follows:

`precond(`*action*`,`*preconditions*`)`

This database entry specifies the facts which must hold in order that action be performable. These preconditions are necessary, but may *not* be sufficient for the action to achieve the ends we desire.

[2] "Reactive systems" [3, 1] are yet another approach to this issue, in which it is argued that we are better off not planning at all.

[3] In practice, we are free to use a more convenient notation in composing the plan library than the one the planner will use.

Entries describing the effects of actions look like this:

add(*formula*, *action*, *effect-preconditions*)
or
delete(*formula*, *action*, *effect-preconditions*)

These entries specify that if *action* is performed in a world in which *both effect-preconditions* **and** the preconditions for *action* hold, then *formula* will hold (not hold) at the end of *action*.

Here is a simple action from Peot and Smith's ski world example:

```
precond(go(?x, ?y), [at(?x), clear(?x, ?y)])
add(at(?z), go(?_,?z), [])
delete(at(?z), go(?z,?_), [])
```

We have used the underline as in Prolog, as an "anonymous" or don't-care variable.

We expand this representation to allow for conditional actions, like those of Peot and Smith [11]. Such conditional actions may have several different, mutually exclusive, sets of outcomes. We capture this by associating with every such outcome an integer. Integers can be added to the effect-preconditions of a postcondition entry to specify that one particular outcome must happen in order for the postcondition to hold. For example, in the Ski World, Peot and Smith have an operator for observing road conditions between two points. There are two possible outcomes to this operator: either the road will be found to be clear, or it will be seen to be closed. Here is how we represent this operator:

```
precond(observe(road(?x,?y)),[unknown(clear(?x,?y)),at(?x)])
add(clear(?x,?y),observe(road(?x,?y)),[bead(?act,1)])
postcond(not(clear(?x,?y)),observe(road(?x,?y)),[bead(?act,2)])
```

The variable `?act` is a special one, which will be bound to the name of the step — the actual instance of the operator — so that we may have more than one conditional action of the same type in our plan.

3.3 Pedestal

McDermott's PEDESTAL planner is a regression planner which represents its plan as a dense line segment, beginning at the initial conditions and ending at the goal. Steps are incrementally added to the plan by associating them with points on the line segment. In order to control this process, the planner will always have a set of active (not yet solved) goals and a set of protections which must be respected. PEDESTAL's goals are pairs $\langle g, v \rangle$: the first component, g being a proposition to be established, and the second being a step for whose benefit the proposition is to be established. The top-level goals are goals of the form $\langle g, \texttt{finish} \rangle$ for the distinguished final step.

At each point in the planning process, PEDESTAL will pick a goal out of its active set, and resolve it. PEDESTAL resolves its goals $\langle g, v \rangle$ in one of three ways, chosen nondeterministically:

1. **g holds in initial conditions:** In this case, the goal may be achieved without performing any action. PEDESTAL adds a protection which guards the goal from the beginning of the plan until step v and continues.

2. **g is established by existing step:** Call this step s. PEDESTAL does the following:

   (a) adds a protection of g from s until v.

   (b) PEDESTAL must ensure that s has the desired effect of establishing g. Let $\Sigma(g, s)$ be the *causation preconditions* for g with respect to s. Post new goal(s)[4] $\langle \Sigma(g, s), s \rangle$.

   (c) PEDESTAL must also ensure that no already-existing step between s and v negates g. This is done by posting additional goals:

   For all steps x such that $s < x < v$, let the *preservation preconditions* of g with respect to x be $\Pi(g, x)$. Post $\langle \Pi(g, x), x \rangle$ as a new goal.

3. **g is established by a new step:** Choose some point in the plan at which to insert a new step, s. Now proceed as per a preexisting step to achieve the goal. In addition, however, PEDESTAL must post as goals the preconditions for act s. Let those preconditions be $\Phi(s)$. Post goal(s) $\langle \Phi(s), s \rangle$.

3.4 Conditional Pedestal

PEDESTAL admits of a fairly straightforward adaptation to conditional planning. Essentially, one adapts the PEDESTAL algorithm by mapping steps onto a chronicle *tree*, instead of a line segment. When one adds conditional actions to the plan, one adds new branches to the tree, running from the conditional action to newly-created goal nodes. One then plans for each new goal node as well as the pre-existing goal node.

At each point in the planning process, pick a goal out of the active set, and resolve it. As before, goals are resolved either by finding that the goal holds in the initial conditions, is established by a pre-existing step, or by inserting a new step.

There is one (substantial) complication: handling the addition of conditional actions to the plan. Recall that conditional actions have multiple outcomes. When we add the conditional action to the plan, we will do so because one of the outcomes will achieve a goal. However, there will be other outcomes which will not, in general, achieve the same goal. One may think of these as "bad outcomes" for the action. For each bad outcome, we introduce a new goal node following the bad outcome. Informally, one might think of this goal node as causing us to plan a recovery from the bad outcome.

Consider a problem from the Ski World. One wants to get to a resort (Snowbird or Park City). One's plan so far might be as shown in Figure 2. One has planned

[4] Because we are assuming ground actions, we can blur the distinction between posting a single goal which is a conjunction and posting a conjunction of goals each of which is a literal.

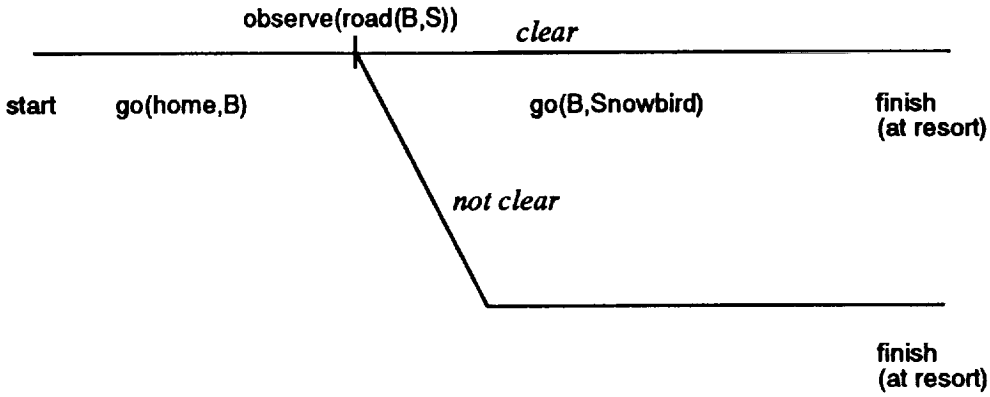

Figure 2: Initial plan to get to the resort.

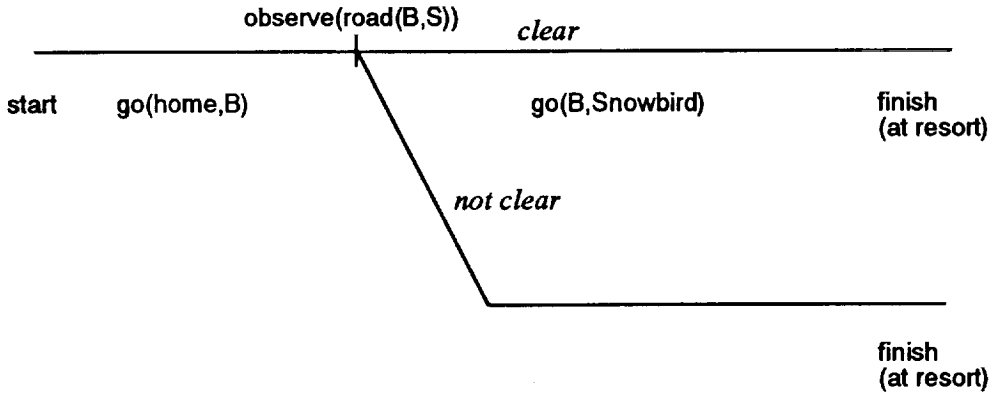

Figure 3: Plan to go to resort after the addition of the conditional action.

to go from home to position B and then from B to Snowbird. However, one has a remaining subgoal, which is to determine that the road from B to Snowbird is clear. Unfortunately, this is not a sure thing. The observation operator has two possible outcomes: either the road will be seen to be clear, or seen to be blocked. In the latter case, one will have to plan a new way to get to the resort. The planner's state after the addition of the observation action is shown in Figure 3. The planner will now have as goals whatever it had before *and* the goal to get to a resort when the road from B to S is not clear, represented by the new goal on the second branch of the plan. Notice that the two plans will share any actions which take place up until the time the status of the road is observed. Notice also that additional actions may be inserted into this shared prefix of the plan: for example, we might as the first step of the plan take some money, if there was a toll on the road from C to Park City. This would only be *necessary* in the event that the road from B to Snowbird is blocked, but would be done before the agent knows whether or not the road is blocked.

3.5 Future work

We are in the process of extending conditional planning to an approach we call *epsilon-safe planning*, in which probabilities are associated with the various outcomes of conditional actions (e.g., with the success or failure of a given classification routine). For any given branch of a conditional plan, we can determine a probability of success. The total probability of success is the sum of the branches which lead to the goal state.

4 Hierarchical Planning with Deadlines

Automated image processing within the *Product Generation System* (PGS) of the Earth Observing System's Data and Information System (EOSDIS) requires the automatic generation of complex analysis plans, detailing the processing steps to be taken to clean up, register, classify, and extract features from a given image. These plans will be executed in a resource-limited environment, competing for such resources as processing time, disk space, and the use of archive servers to retrieve data from long-term mass storage. To complicate matters, it is important that the results of these plans (the completed analysis products) be delivered in a timely fashion to the scientists requesting them.

In joint work at the Honeywell Technology Center (HTC) and NASA's Goddard Space Flight Center, we have developed a planner that generates hierarchical plans for PGS image processing. The schemas used by this planner (based on Nonlin's *Task Formalism* (TF))[13] have been extended to record information about the estimated and worst-case duration of a given task, and about the tasks' resource usage. This information is used during plan construction, for example in the rejection of an otherwise promising expansion for a given sub-task because it requires more time than is available, and in the construction of detailed schedules for image processing tasks.

Accurate estimates of the time required for image processing tasks are hard to come by, particularly for more abstract tasks (e.g., "identify features," rather than a detailed set of file manipulations). We have constructed a routine that traverses the set of tasks defined for the PGS planner, defining worst-case estimates for abstract tasks based on the time required for their subtasks. Use of this routine, coupled with a facility allowing primitive task estimates to be updated either manually or based on statistics gathered as the system runs over time, allows us to continually refine the initially somewhat undependable time estimates, resulting in increasingly effective management of scarce computational resources for the image processing task.

The choice of Nonlin as a starting place was driven by the fact that the TF can be used effectively to describe image processing tasks. Human users tend to break these tasks down into hierarchies of subtasks (e.g., "remove noise" may involve scan-line removal, smoothing, and despeckling, usually in that order) in a way very naturally expressible in TF. Nonlin's main drawbacks included the lack of any facilities for

reasoning about duration, deadlines, and resources. Deviser [14] adds durations, but is not sufficiently flexible and scales poorly. Eventually, the planning function will be integrated with scheduling and dispatch functions that will use the same representations.

Durations and deadlines were added to the TF through the addition of a :duration slot in task schemas. These specifications may be numbers, ranges, or a function of the schema variables evaluated when the schema is instantiated. The underlying representation of time is the TMM [5], in an implementation developed at the Honeywell Technology Center. Calculation of duration bounds during task expansion provides an additional constraint on search: if at any time the time needed for a given task expansion exceeds the time available, the system will backtrack, trying an alternative expansion at the current level or higher planning levels until a time-feasible schedule is found (or the system gives up).

5 Summary and Conclusions

Automating the processing of satellite earth science data is both timely and with a high potential for significant improvement of the current environment. Timely, because the current tools for managing and processing this data are beginning to be overwhelmed. This trend will only worsen as new satellite systems come on line over the next few years, most notably (but not exclusively) EOS. As we have also argued, automation of these tasks shows great potential benefit. Existing research in AI, Operations Research, databases, and distributed systems can be adapted to alleviate the looming data overload, in some cases by freeing humans from the process entirely (e.g., generating browse products on ingest), and in other cases by providing better tools for interactive use (e.g., helping scientists to retrieve and process archived data).

In this paper, we have presented results on the application to image processing of two bodies of work drawn from current research in AI planning: conditional planning and planning with duration and deadlines. These results are promising, but the work is by no means complete. Moving these systems into operational use will require further refinement and development, which we expect to accomplish over the next twelve to eighteen months.

References

[1] Brooks, Rodney A., *A Robust Layered Control System for a Mobile Robot*, A.I. Memo No. 864, MIT Artificial Intelligence Laboratory, 1985.

[2] Campbell, W., Short, N., Jr., Roelofs, L., and Dorfman, E., Using Semantic Data Modeling Techniques to Organize an Object-Oriented Database For Extending the Mass Storage Model, *42nd Congress of the International Astronautical Federation*, 1991.

[3] Chapman, David, *Planning for Conjunctive Goals*, Technical Report AI-TR-802, MIT Artificial Intelligence Laboratory, 1985.

[4] Chapman, David and Agre, Phil, Abstract Reasoning as Emergent from Concrete Activity, Georgeff, Michael P. and Lansky, Amy L., (Eds.), *The 1986 Workshop on Reasoning About Actions and Plans*, Morgan-Kaufman, 1987, 411–424.

[5] Dean, Thomas and McDermott, Drew V., Temporal Data Base Management, *Artificial Intelligence*, **32** (1987) 1–55.

[6] Dozier, J. and Ramapriyan, H.K., Planning for the EOS Data and Information System (EOSDIS), *The Science of Global Environmental Change*, (NATO ASI, 1990).

[7] Firby, R. James, An Investigation in Reactive Planning in Complex Domains, *Proceedings AAAI-87, Seattle, Washington*, AAAI, 1987, 196–201.

[8] Green, J., The New Space and Earth Science Information Systems at NASA's Archive, *Government Information Quarterly*, 7(2) (1990) 141–7.

[9] McDermott, Drew, Regression Planning, *International Journal of Intelligent Systems*, 6(4) (1991) 357–416.

[10] Minton, Steven, Bresina, John, and Drummond, Mark, Commitment Strategies in Planning: A Comparative Analysis, *Proceedings IJCAI 12, Sydney, Australia*, IJCAI, 1991.

[11] Peot, Mark A. and Smith, David E., Conditional Nonlinear Planning, *Artificial Intelligence Planning Systems:Proceedings of the First International Conference*, Los Altos, CA, 1992, 189–197, Morgan Kaufmann Publishers, Inc.

[12] Ramapriyan, H.K., The EOS Data and Information System, *American Institute of Aeronautics and Astronautics*, 1990.

[13] Tate, Austin, Generating Project Networks, *Proceedings IJCAI 5, Cambridge, MA*, IJCAI, 1977.

[14] Vere, Steven, Planning in Time: Windows and Durations for Activities and Goals, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **5** (1983) 246–267.