Plan Validation and Mixed-Initiative Planning in Space Operations

Richard Howey Derek Long
Maria Fox

richard.howey, derek.long, maria.fox @cis.strath.ac.uk

Department of Computer and Information Systems
University of Strathclyde, Glasgow, UK

Abstract. This paper describes our experiences in using our plan validation tool, VAL, in the development of a mixed-initiative plan construction tool intended to support space operations planning. VAL was initially developed to support the 3rd International Planning Competition (IPC), but has subsequently been extended in order to exploit its capabilities in plan validation and development. In particular, it has been extended to include advanced features of PDDL2.1 that were untested in the 3rd IPC but which have proved important in its application to mixed-initiative planning in the space operations project. The tool has also been extended to keep abreast of developments in PDDL, providing critical support to participants and organisers of the 4th IPC.

1 Introduction

VAL is a plan validation tool for PDDL. It played an important role in the 3rd IPC [9], allowing reliable validation of the several thousand plans produced by the competitors, as well as providing competitors with support for their development and debugging cycles. We have found the capabilities of VAL to be critical in understanding the structures of large plans, providing visualisation and reporting facilities [4]. VAL continued to play an important part in the 4th IPC [3], which saw seen several minor extensions to PDDL and its semantics.

VAL has also been extended to support features that were included in the definition of PDDL2.1, although not used in the competitions, including the expression of continuous change. In this paper we breifly describe the semantic developments of PDDL required to support continuous change and go on to give a motivating example for the role of this extension, based on a project in space operations planning. VAL has proved to be a valuable resource for this project, supporting the development of a domain description, the validation of plans constructed by humans and, using a new extension to the system, providing advice on the correction of flawed plans. In this paper we describe this recent feature and discuss ways in which the tool can be exploited in plan development in a mixed-initiative mode. We also briefly discuss how this approach might be developed to allow VAL to play a key role in fully automated plan construction.

2 PDDL and its Semantics with Continuous Effects

When Drew McDermott and the first planning competition committee proposed PDDL as a community standard [12] planning domain description language, they initiated an important process of development in which the planning research community has seen incremental extensions and modifications as the language has adapted to various goals. In the first instance, the core of PDDL was a STRIPS language, offering an ADL extension. This language has a semantics that is widely accepted, based on a simple state-transition model, with few areas of potential ambiguity. Perhaps the most significant issue for which alternative resolutions exist is concurrency: classical plans are often considered to be *sequences* of steps, representing state transitions, but partial-order planning [11] and Graphplan [2] both offer alternative models in which some form of parallelism is considered.

McDermott developed a simple plan validation tool for PDDL, intended only for sequential plans. However, the question of interpretation for more complex extensions of PDDL is more difficult. There is no prior widely accepted model, so choices must be made that are not necessarily universally accepted. Since the language plays a central role in communication of domains between researchers, it is important that there be a standard by which a common understanding may be developed for the semantics of domains and plans for those domains. A formal semantics is the first component of this. However, a formal semantics is not sufficient by itself, because a formal semantics is notoriously difficult to read. In practice, many formal semantics are read in detail by few and understood in all details by even fewer. To make the semantics accessible, their implementation as a validation tool is an important step. In this form, it is possible to confirm understanding of the semantics by testing various plans and domains with the tool, confirming the behaviour is as expected. VAL supplies a variety of forms of feedback, making it possible to explore quite precisely what might be wrong with a flawed plan and aiding in the interpretation of the more subtle details of the semantics.

Continuous effects are an important extension to PDDL for accurately modelling change in real world situations, their implementation in VAL is crucial in the mixed initiative application for the Beagle 2. Continuous effects can only affect metric quantities: it is not possible to change a propositional fluent continuously. A metric variable that can be changed by a continuous effect is called a *Primitive Numerical Expression (PNE)*. A durative action that has a continuous effect on a PNE changes it so that the values taken are described by a continuous function of time. We have developed formal semantics for the inclusion of continuous effects in PDDL, so called PDDL2.1 level 4. The semantics can be described by continuous activity on a real time line punctuated with discrete activity; for details see [5, 6]. When defining continuous effects in the domain model the rates of change of the PNEs are defined, it is possible for these rates of change to refer to PNEs that are themselves changing continuously. In this way the continuous effects are defined by a system of differential equations. These differential equations must be solved in VAL, however it is infeasible to solve all possible systems of differential equations, so we restrict ourselves to an interesting subset. Namely solutions that are given by polynomials or certain classes of exponential functions (also some numerical solutions). Apart from solving differential equations it is also a requirement to find the roots of real valued functions on given intervals. This is due to invariant conditions of durative actions which must hold over the application of the durative action. An invariant condition may state that a PNE must be below a certain threshold, for example $f < k$ on $(0, T)$, and if f is changing continuously we need to consider the roots of $f - k$ on $(0, T)$. For details on differential equations and rooting finding techniques in the context of plan validation see [5].

3 Space Operations Planning for the Beagle 2 Lander

On December 25th, 2003, a small lander, travelling with the Mars Express orbiter, was expected to land on Mars surface. Unfortunately, as with a high proportion of Mars landers, the Beagle 2 lander was unsuccessful. Various explanations of its failure have been proposed, including the possibility that the density of the Martian atmosphere is not as high as had been thought and, as a result, the parachute-brake failed to slow the lander sufficiently before impact. Despite this major setback, the design of the lander is considered so innovative and efficient (both in terms of cost and in terms of science-to-mass) that a future repeat attempt is being seriously considered. Funded by the European Space Agency, the authors have explored the possible roles of planning technology in space operations [13]. The project includes examining interactions between human operations planners and automated technology, initially in the context of operations plans for Beagle 2. Considerable expertise was built up around the Beagle 2 systems, including a partial domain model for human planning operations, and this has formed the core of a project to exploit planning technology to support mixed-initiative and partially automated planning for the lander operations.

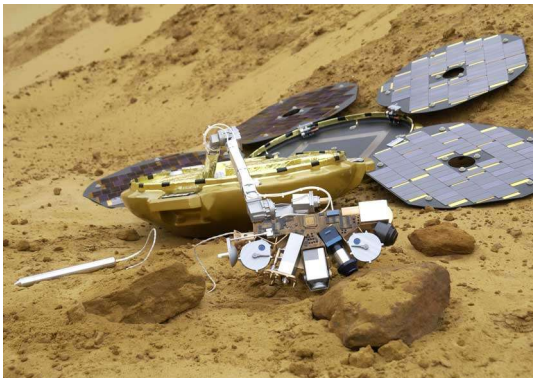

Figure 1: Beagle 2

Beagle 2 is a static lander, equipped with a jointed arm carrying an array of scientific instruments in a "paw" at its end. Included in the paw is a mole capable of drilling into soil around the lander to a distance of more than 2 meters, to retrieve soil samples for gas analysis on board the lander. Beagle 2 is essentially a geological survey system, capable of performing an array of geological and environmental measurements in its immediate surroundings.

All lander operations are constrained by power availability, provided by solar energy with a battery for storage, and by temperatures. In addition, the lander is equipped with storage buffers for instrument data. These are too small to store all the data generated by the succession of experiments performed by the lander. Data must therefore be uplinked from the lander not only to return it to the scientists, but also to free up space for storage of succeeding measurements. Uplink windows are constrained, both in duration and in bandwidth, so that it is not possible to empty all the buffers in a window. Thus, there is a further problem of scheduling the uplinking of data into the communication windows in order to allow buffers to be emptied in time for experiments to be performed. These constraints make the management of data and communication a critical element of the planning problem, in common with other deep space missions.

Therefore, planning lander operations involves managing constraints on continuously changing quantities (the generated power levels and temperatures), scheduling the use of resources, planning the movement sequences of the arm and use of the instruments. Human operations planners were to have carried out the planning for the lander in a complex process involving scientists, providing mission goals and the operations to achieve them, and lander operations personnel, concerned with lander security and, therefore, the power resources and internal lander monitoring systems. When we became involved in the project we discovered that the existing partial domain description was in a form that closely resembled PDDL2.1 durative action descriptions. It was possible to translate the description into PDDL automat-

ically using a simple automatic translator. The domain encoding began with over 50 actions and this has increased to nearly 70 actions following further domain analysis.

Power is the most important continuous factor in the operations of the lander. The lander operates close to margins and the model of the solar generation and the battery charging profiles are vital in determining when operations can be planned. The management of battery and solar power is sufficiently close to the margins of operational envelopes that it plans must interact with the continuous changes involved in the physical system rather than with abstractions into coarse-grained simple step-function changes. Of course, with sufficiently small time-steps a step-function model can approximate the continuous change adequately, but it is infeasible to attempt to model this level of granularity explicitly in the planning domain description. Therefore, this problem demands that the planner has access to a sufficiently detailed model of the continuous changes that affect the power systems.

4 Mixed-Initiative Planning

Mixed-initiative planning is a well-recognised path by which to introduce automatic planning technology into a context in which planning is currently performed manually. The idea is to allow human and automatic planners to cooperate in producing a plan. There are several aspects to this interaction. It simplifies the problems facing the automatic planning, since difficult choices in plan construction can be passed to a human, while the human can benefit from not having to manage the bulk of easier planning decisions and simple book-keeping tasks. However, the requirement for interaction imposes a more stringent demand on the system developers to ensure that feedback from the planner can be provided in a way that makes sense to the human planner, in terms that the human planner can relate to the planning task with which they are familiar. Furthermore, the automatic planning task changes from that of complete plan construction to one of plan repair and iterative plan improvement. We are not concerned with all the details of the demonstrator we have developed in this paper. Instead, we focus on the role that VAL has played in supporting the problems of feedback and plan repair.

4.1 Using VAL *in Mixed-Initiative Planning*

When VAL is used in its simplest form, without any parameters, in the case of plan failure it reports only that the plan has failed. An option is available for verbose output in which the system generates a report explaining which action in a plan has failed. However, this is still of limited use since no indication is given of how an action precondition might have failed to be satisfied. The action precondition might be very complex, but only failed due to one literal with the incorrect truth value. For example, a large factory machine may have an action for starting processing with a complicated precondition, but an instance of the action in a plan might fail simply because the machine is not switched on prior to planned execution of the start action. Feedback from the plan validation reporting that the machine needs to be switched on would be invaluable advice on how to fix the plan. In complex plans identifying even simple failures such as this can be difficult due to the obscuring effects of the actions surrounding the failure.

With the intention of supplying more informed feedback we have developed in VAL a detailed advice sub-system indicating how to satisfy unsatisfied preconditions in an invalid

plan. The advice can be used in a *mixed-initiative planning* cycle in which the human planner firstly produces a plan either by hand or with the aid of software before VAL simulates execution of the plan giving detailed advice on how to repair the plan for each unsatisfied action precondition (or invariant condition or goal). The advice can then be used by the human planner to produce a new plan to correcting the errors, or at least some of them. The new plan can then be executed using VAL which produces new plan repair advice, and so on.

In general, the advice offered by VAL indicates why a given plan failed and what conditions must be achieved in order to repair it. It does not indicate which actions might be applied to achieve those conditions or explore the interactions they might introduce into the plan if they are added to it. Therefore, the advice from VAL must be seen as the first stage in the repair or reconstruction of a flawed plan: other components are necessary to decide how best to act on the advice if this decision is to be made automatically.

4.2 Structure of Plan Repair Advice

The advice given for a failed precondition is derived from a PDDL precondition expression and stored in a structure called an *advice proposition*.

Definition 4.1. Advice Proposition *For a given* PDDL *precondition of an action in a plan the* advice proposition *provides instructions on how the state, S, must be altered at this point in the plan in order to satisfy the precondition. An advice proposition (AP) is one of the following:*

- *Instructions to set A to true, for some literal A.*
- *Instructions to set A to false, for some literal A.*
- *Instructions to satisfy a comparison consisting of numerical expressions where each PNE has its current value reported.*
- *A list of APs where* all *must be followed (conjunction AP).*
- *A list of APs where* at least one *must be followed (disjunction AP).*
- *No advice (the empty advice case).*

VAL produces advice for each unsatisfied precondition by mapping the precondition and state to an advice proposition.

Definition 4.2. *Let ϕ be the mapping from a* PDDL *precondition, P, and a state, S, to an advice proposition defined as follows if P is a literal, comparison or connective respectively.*

$$\begin{aligned}
\phi(P, S) &:= \textit{if } S \models P \textit{ then} \text{ no advice } \textit{else} \text{ set } P \text{ to true} \\
\phi(P, S) &:= \textit{if } P \textit{ is an unsatisfied comparison then} \text{ satisfy } P \\
\phi(\wedge_i X_i, S) &:= \wedge_i \phi(X_i, S), \textit{ for each unsatisfied } X_i \textit{ in } S \\
\phi(\vee_i X_i, S) &:= \vee_i \phi(X_i, S), \textit{ for each unsatisfied } X_i \textit{ in } S \\
\phi(X \rightarrow Y, S) &:= \phi(\neg X \vee Y, S)
\end{aligned}$$

If P is a negation, $P = \neg Q$, then $\phi(P) = \psi(Q)$ where ψ is defined as below if Q is a predicate, comparison or connective respectively.

$$\begin{aligned}
\psi(Q, S) &:= \textit{if } Q \not\models S \textit{ then} \text{ no advice } \textit{else} \text{ set } Q \text{ to false} \\
\psi(Q, S) &:= \textit{if } Q \textit{ is an unsatisfied comparison then} \text{ satisfy } Q \\
\psi(\wedge_i X_i, S) &:= \vee_i \phi(\neg X_i, S), \textit{ for each satisfied } X_i \textit{ in } S \\
\psi(\vee_i X_i, S) &:= \wedge_i \phi(\neg X_i, S), \textit{ for each satisfied } X_i \textit{ in } S \\
\psi(X \rightarrow Y, S) &:= \phi(X \wedge \neg Y, S) \\
\psi(\neg Q', S) &:= \phi(Q', S)
\end{aligned}$$

The map ϕ is well defined since PDDL preconditions and states are finite. Starting from a PDDL precondition that is not satisfied always yields a non-empty advice proposition. The advice takes the form of lists of APs, conjoined or disjoined according to contex. Further advice lists may then be nested. The actual conditions that need to be changed in the state will be the truth value of predicates and the numerical values of PNEs.

4.3 Advice on Invariants depending on Continuous Effects

The introduction of continuous effects into a plan further complicates the validation of an invariant over a given interval. There is a natural extension to the plan repair advice given by ϕ to invariant conditions depending on continuously changing PNEs. An invariant condition must hold for all values on a given interval, this further consideration only changes the advice given by ϕ for comparisons that depend on continuously changing PNEs. Instead of considering just one state the advice for satisfying an invariant must consider: one logical state (for the predicates), and a continuously changing numerical state on the interval in question for comparisons depending on continuous effects. The advice for such a comparison is that it needs to be satisfied on the interval, together with a report of the subset of values of the interval that the comparison is satisfied on.

For a disjunctive advice proposition which states that one of the following must be satisfied the meaning should be interpreted appropriately when referring to invariant conditions. That is, for each time value in the invariant interval one of the advice propositions must be followed. The advice proposition that is followed need not be the same advice proposition for each time value. See [5], section 7.2 for more details on disjunctive invariants.

4.4 Plan Repair Example Using VAL*: Beagle 2 Plan*

In this section we consider an example of the use of VAL in a mixed-initiative setting for the Beagle 2 planning problem. In this simple scenario, starting shortly after dawn, the Beagle begins with its arm stowed and its battery at a low state of charge (due to overnight operations). The operations planner wishes to construct a plan that will allow examination of a rock that has been named "peanuts". The examination procedure consists of taking a close up image with the stereoscopic camera, then a closer image with the microscope, grinding a core sample, transferring it to the ovens and processing it in the gas analysis system (GAP). We will examine only the first few actions. An initial plan is constructed (through a GUI that is not critical to the work described here):

```
1:     (generate-solar-power) [43200] ;Starts continuous power generation
1:     (PAW-move stowed wind_high_modified) [100]
500:   (PAW-move wind_high_modified closeup_peanuts) [800]
1400:  (SEQ-SCS-CLOSEUP closeup_peanuts) [130]
3550:  (SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts) [17300]
```

VAL is then applied to it, reporting failure with advice as shown below.[1]

```
Only one possible achiever for the condition:
      (MIC_FOCUS_RANGED peanuts_sample)
Adding SEQ-MIC-FIND_FOCUS_RANGE to complete before 3550
Added PAW-move action from peanuts_closeup to peanuts starting at 3349
```

[1] For simplicity we show all times in seconds relative to the plan start. In fact, the GUI uses absolute times and reports are given in terms of these.

Examination of the LaTeX report reveals that the reason for this advice is that two invariants for the action `(SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts)` are violated. One is that the microscope must be properly focussed before attempting to use it to capture images and the second is that the microscope must be in position at "peanuts" in order to be able to capture the images. The first condition is achieved by the addition of a new action (there is only one choice) and the second by a proposed addition on an additional PAW movement action. Note, however, that the new action `(SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts)` has a precondition that is also unsatisfied. The validation process is, deliberately, not invoked recursively to attempt to repair flaws in the first phase of repairs. This is to ensure that the human has some opportunity to monitor the process of repair and to interact with it by accepting parts of the repair and rejecting or modifying others.

Where advice is generated for PAW movement, the plan repair machinery finds efficient paths between known arm configurations for the path, possibly generating a sequence of PAW-moves in order to traverse a path. There is no geometric planning involved in this — individual moves between specific pairs of locations are preplanned as command sequences for the individual motors in the arm controller. However, the recognition of the path-planning problem that is involved in identifying the sequence of waypoints for the arm to traverse is an important element of the repair process. This machinery depends on the recognition of the underlying behaviour of PAW-move actions as a *mobile generic type* [8].

If, at this point, the human operator simply chooses to reinvoke the replanning system then the following plan is passed to VAL.

```
1:     (generate-solar-power) [43200] ;Starts continuous power generation
1:     (PAW-move stowed wind_high_modified) [100]
500:   (PAW-move wind_high_modified peanuts_closeup) [800]
1400:  (SEQ-SCS-CLOSEUP peanuts_closeup) [130]
1567:  (SEQ-MIC-FIND_FOCUS_RANGE peanuts) [1982]
3550:  (SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts) [17300]
```

The observant reader will notice that this plan does not contain the additional PAW movement action advised above. This is because the interface between the GUI and VAL relies on a plan representation that is convenient to the human operator, abstracting details that seem obvious. In particular, a PAW-move action only includes its destination argument — it is assumed that the starting point for each move is wherever was the last location of the PAW. In the partially repaired plan proposed after the first cycle the inserted PAW-move comes too late to achieve the invariant condition of the `(SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts)` action, so the system infers that the PAW must actually be moved earlier and that this subsequent move is a redundant attempt to move the PAW from "peanuts" to itself. This is reported as a flawed action and deleted from the plan, which is why it does not appear in the plan listed above. The repair advice is therefore:

```
Error in finding duration for: (PAW-move peanuts peanuts)
Action instance does not exist!
Plan failed to execute
Report in report.ps
Added PAW-move action from peanuts_closeup to peanuts starting at 1366
```

The plan now becomes:

```
1:     (generate-solar-power) [43200] ;Starts continuous power generation
1:     (PAW-move stowed wind_high_modified) [100]
500:   (PAW-move wind_high_modified peanuts_closeup) [800]
1366:  (PAW-move peanuts_closeup peanuts) [200]
1400:  (SEQ-SCS-CLOSEUP peanuts_closeup) [130]
1567:  (SEQ-MIC-FIND_FOCUS_RANGE peanuts) [1982]
3550:  (SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts) [17300]
```

and it still contains an important flaw. The new attempt to position the necessary PAW-move action has created a different conflict, this time with the action `(SEQ-SCS-CLOSEUP peanuts_closeup)`, which requires that the PAW stays immobile during the image capture. Repairing this flaw is a more difficult problem. The plan repair system observes that the profile for the behaviour of the required condition is that it is true in the interval following the second PAW-move up to the third PAW-move. It proposes to exploit this by moving the action `(SEQ-SCS-CLOSEUP peanuts_closeup)` earlier, but there is not sufficient time between the two PAW-moves to allow the stereo-camera image to be captured. Therefore, the third PAW-move is moved later to make space and all of the actions dependent on the third PAW-move are delayed by the same amount of time. This repair is explained in the following output from the system:

```
Propose to delay (PAW-move peanuts_closeup peanuts) by 66 seconds
Propose to delay (SEQ-MIC-FIND_FOCUS_RANGE peanuts) by 66 seconds
Propose to delay (SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts) by 66 seconds
Propose to bring forward (SEQ-SCS-CLOSEUP peanuts_closeup) by 99 seconds
```

The final plan is:

```
1:     (generate-solar-power) [43200] ;Starts continuous power generation
1:     (PAW-move stowed wind_high_modified) [100]
500:   (PAW-move wind_high_modified peanuts_closeup) [800]
1301:  (SEQ-SCS-CLOSEUP peanuts_closeup) [130]
1432:  (PAW-move peanuts_closeup peanuts) [200]
1633:  (SEQ-MIC-FIND_FOCUS_RANGE peanuts) [1982]
3616:  (SEQ-MIC-FULL_SET_COMPRESS_EACH peanuts) [17300]
```

If the final plan is validated one last time, VAL reports success and generates a report complete with graphs illustrating the power levels during execution. If this same plan is executed from a state in which the battery state of charge is close to its critical minimum level then a different outcome can be generated:

```
Problem found without any repairs:
Condition unsatisfied for:
   (generate-solar-power) at time 1
```

This is due to the fact that the initial powerdraw due to the use of the instruments occurs too early in the recharging cycle and the power level dips below the minimum level. This is seen in the report generated by VAL, which explains that the invariant (that the charge remain above the critical level) is unsatisfied over particular intervals and in the graph of charge produced by VAL for the report (figure 2). The curve shows that the charge dips low in the early stages of the plan, when powerdraw exceeds solar-power generation. The shape is complex due to the changing demands of the different activities across the interval. The repair for this is to slide all the activity later in the day when power has been generated by solar generation, but our machinery is not yet capable of generating this repair automatically. (It can handle simpler cases in which single actions must be slipped to account for violation of invariants by continuously changing PNEs).

The process of repair depends on a rich plan representation, capturing the dependency structure between the actions and possible external events. This temporal aspect of the structure is an important additional element, along with the effect of continuous change, that is not considered in the otherwise closely related work of Lemai and Ingrand [7] on plan repair. In their work they consider plans as partial order structures and build repairs using traditional partial-order planning flaw repair strategies. This is a valuable approach to handling plan flaws and can be generalised to handle metric conditions to some extent. However, they do not consider continuous change or its impact on invariants and this is an important aspect of the current work. In the context of the space operations project, management of continuously

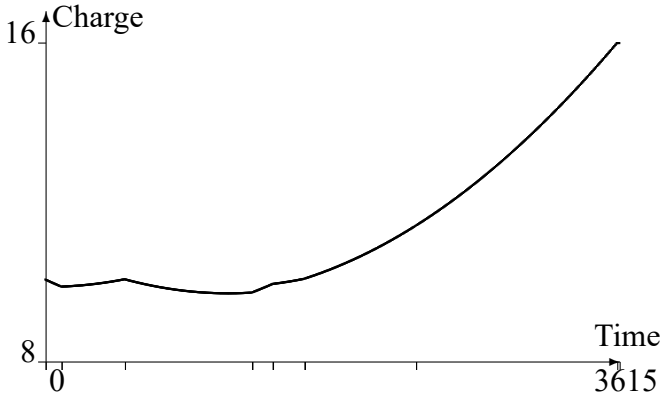

Figure 2: Graph of Charge.

changing power supplies is a vital part of the planning problem, as has also been observed in the development of MAPGEN [1], the NASA tool used for mixed-initiative planning for the Mars Exploratory Rovers mission. MAPGEN does not attempt to model the power management directly, relying instead on a separate purpose-built model. This has the advantage of being highly accurate, but the important disadvantage of being opaque to the planning and validation process. This means that it is difficult to make predictions about the right way to repair plans that violate the power envelope.

5 Further Work and Concluding Remarks

Polynomials were the first continuous effects to be handled by VAL [5]. The approach can be extended to continuous effects described by more complex functions, by using polynomials to approximate the functions. This approach has been implemented for exponential functions in VAL. This extension has been an important step in the exploitation of VAL in the context of the Beagle 2 operations, since the power models are sufficiently complex that they cannot easily be modelled using simple polynomials alone. In fact, to model them it has been necessary to numerically solve certain classes of differential equations: we have used the Runge-Kutta-Fehlberg [10] method with very encouraging results.

We are investigating ways to improve the advice given for repairing a plan, for example which actions should be added, removed or moved within the plan. In particular how a durative action with an invariant condition which depends on continuous activity, may be moved within the plan to satisfy the invariant. We are aiming for the least human input in the mixed-initiative planning process, so that most plan repair activity can be automated by VAL. Ultimately, VAL should support a complete plan repair strategy.

This work is part of a larger project exploring the transfer of planning technology into space operations planning in European space missions. This is in the context of world-wide efforts in space exploration and the NASA mobile robots missions currently being pursued on Mars. These missions also make extensive use of mixed-initiative planning aids, including the MAPGEN[1] tool, used successfully in the Mars Exploratory Rover missions.

Applying planning technology to real problems highlights the need to provide solutions to problems that are often not considered in the pure planning research community. Mixed-initiative planning has long been considered an important bridging technology to help human planners to become familiar with and more trusting of automatic planning technologies, while also solving some of the difficulties faced by planners in complex and realistic domains.

In this paper we have discussed the use of VAL, our plan validation technology, as a tool in mixed-initiative planning for space operations. An important aspect of this tool is that it is directly linked to our development of the semantics of PDDL and therefore provides a basis for formal validation of the domain descriptions we are constructing. The importance of continuous change in this domain is an added complication. We have advocated the role of continuous change in planning domain models for some time and this domain is an illustration that it is of key importance in real domain problems.

In the context of the space operations planning we have integrated VAL into the tool intended for Beagle 2 operations planning. We anticipate that this bridge between theory and practice will continue to inspire and motivate further developments in our planning technology.

References

[1] M. Ai-Change, J. Bresina, L. Charest, J. Hsu, A.K. Jónsson, R. Kanefsy, P. Maldegue, P. Morris, K. Rajan, and J. Yglesias. MAPGEN: Mixed intitive activity planning for the Mars Exploratory Rover mission. In *Proceedings of Demonstration Systems Track, ICAPS'03*, 2003.

[2] A. Blum and M. Furst. Fast Planning through Plan-graph Analysis. In *Proceedings of IJCAI*, 1995.

[3] S. Edelkamp, J. Hoffmann, and committee. The 4th International Planning Competition 2004, `www.informatik.uni-freiburg.de/~hoffmann/ipc-4/`.

[4] R. Howey and D. Long. VAL's progress: The automatic validation tool for PDDL2.1 used in the international planning competition. In *Proc. of ICAPS Workshop on the IPC*, 2003.

[5] R. Howey and D. Long. Validating plans with continuous effects. In *Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, pages 115–124, December 2003.

[6] R. Howey and D. Long. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *Proc. of The 16th IEEE International Conference on Tools with Artificial Intelligence*, Florida, USA, 2004.

[7] S. Lemai and F. Ingrand. Interleaving temporal planning and execution in robotics domains. In *Proceedings of AAAI'04*, 2004.

[8] D. Long and M. Fox. Recognizing and exploiting generic types in planning domains. In *International AI Planning Systems conference, AIPS 2000, Breckenridge, Colorado, USA.*, 2000.

[9] D. Long and M. Fox. The 3rd International Planning Competition: Results and analysis. *Journal of AI Research*, 20, 2003.

[10] J. H. Mathews and K. K. Fink. *Numerical Methods Using Matlab*. Prentice-Hall Inc., New Jersey, USA, 4th edition, 2004.

[11] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 634–639, Anaheim, California, USA, 1991. AAAI Press/MIT Press.

[12] D. McDermott and AIPS'98 IPC Committee. PDDL–the planning domain definition language. Technical report, Available at: `www.cs.yale.edu/homes/dvm`, 1998.

[13] M.J. Woods, R.S. Aylett, D. Long, M. Fox, and R. Ward. Assessing planning and scheduling technologies for deep space exploration. In *Proceedings of 4th British Conference on (Mobile) Robotics: Towards Intelligent Mobile Robots*, 2003.