UPMurphi Released: PDDL+ Planning for Hybrid Systems

Giuseppe Della Penna
University of L'Aquila
Italy
giuseppe.dellapenna@univaq.it

Benedetto Intrigila
University of Roma "Tor Vergata"
Italy
intrigil@mat.uniroma2.it

Daniele Magazzeni
King's College London
United Kingdom
daniele.magazzeni@kcl.ac.uk

Fabio Mercorio
University of Milan-Bicocca
Italy
fabio.mercorio@unimib.it

Abstract

In this tool paper, we present the release of UPMurphi, a universal planner for PDDL+ domains. Planning for hybrid domains has found increasing attention in the planning community, motivated by the need to address more realistic scenarios. While a number of techniques for planning with a subset of PDDL+ domains have been proposed, UPMurphi is able to handle the full range of PDDL+ features, including nonlinear continuous processes, exogenous events, Timed Initial Literals and numeric Timed Initial Fluents.

This paper describes the UPMurphi framework and presents its main features, together with a guide for using the tool, and some examples where UPMurphi has been successfully applied.

1 Introduction

A hybrid system is one in which there are both continuous control parameters and discrete logical modes of operation. It represents a powerful model to describe the dynamic behaviour of modern engineering artefacts. Hybrid systems frequently occur in practice, e.g., in robotics or embedded systems. Dealing with hybrid systems is becoming more and more an important challenge, as many real-world scenarios feature a mixture of discrete and continuous behaviours. Some example applications include coordination of activities of a planetary lander, oil refinery management, autonomous vehicles. Such scenarios motivate the need to reason with mixed discrete-continuous domains.

Planning for hybrid domains has found increasing attention in the planning community, motivated by the need to address more realistic scenarios and to interact with robotics and control frameworks. PDDL+ (Fox and Long 2006) is the extension of PDDL which allows the modelling of hybrid domains through the use of discrete actions, processes (that model continuous change over time), and exogenous events (that model changes that are initiated by the environment).

A number of techniques for PDDL+ planning have been proposed (Penberthy and Weld 1994; McDermott 2003; Li and Williams 2008; Coles et al. 2012; Shin and Davis 2005; Coles and Coles 2014; Molineaux, Klenk, and Aha 2010; Bryce and Gao 2015; Bogomolov et al. 2014; 2015). However, despite the recent efforts in proposing new algorithms and approaches for this domain, UPMurphi is currently the only available tool able to handle the full range of PDDL+ features.

The purpose of this paper is to accompany the release of the UPMurphi tool. To this aim, in the rest of the paper we overview the main features of the planner and we then describe the main techniques used for dealing with hybrid domains. In Section 3 we describe the general framework and provide a guide for using the tool. In Section 4 we survey some applications for which UPMurphi has been successfully used. Section 5 concludes the paper.

What UPMurphi Can Do. UPMurphi is a forward-search planner for PDDL+ domains. It can handle the whole PDDL+ language, including nonlinear continuous processes, exogenous events, Timed Initial Literals and numeric Timed Initial Fluents. It can be used either to find a single plan from the initial state to a goal state or to find a universal plan, i.e., a policy for handling the state space generated from the initial state.

2 How UPMurphi Works

UPMurphi is based on the planning-as-model-checking paradigm (Cimatti et al. 1997), and it is built on top of the CMurphi model checker (Cached Murphi Web Page 2006).

Planning in hybrid domains is challenging because in addition to the discrete state explosion problem, the continuous behaviour causes the reachability problem generally even to be undecidable. UPMurphi handles the hybrid dynamics through discretisation of time and continuous variables and by planning within a finite horizon. In this way, the state space is finite.

UPMurphi implements the *Discretise and Validate* approach (Della Penna et al. 2009) which is sketched in Figure 1. Here, the continuous dynamics of the system is relaxed into a discretised model, where discrete time steps and corresponding step functions for continuous values are used in place of the original continuous dynamics. Then, UPMurphi performs a forward reachability analysis in the discretised state space, searching for a path from the initial state to a state satisfying the goal condition. The discrete solution is then validated against the continuous model through the plan validator VAL (Howey, Long, and Fox 2004) to check whether the solution is valid or not. If it is invalid, the discretisation is refined and the process iterates. If UPMurphi fails to find a plan at one discretisation the process can be iterated at a finer grained discretisation. The validation output can guide the user in identifying a suitable finer discretisation.

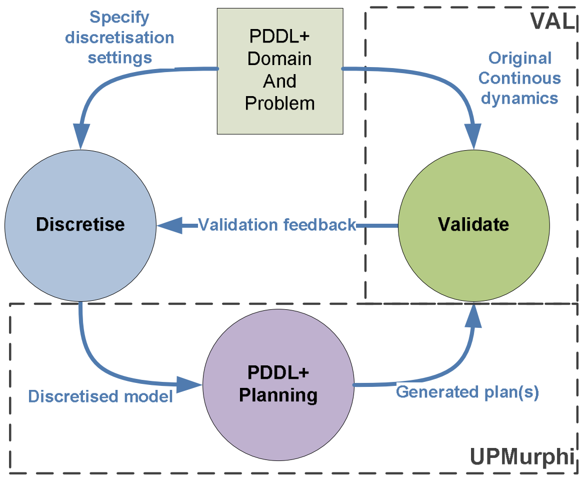

Figure 1: Graphical representation of the D&V approach

3 How to Use UPMurphi

The overall UPMurphi architecture is sketched in Figure 3. UPMurphi can be invoked through the `upmc` command by passing both the PDDL+ domain and problem as arguments. This intitiates the following chain of operations.

PDDL+ translation: the PDDL-to-UPmurphi compiler, built on top of the VAL PDDL+ parser, takes as input a PDDL+ model and outputs a semantically equivalent UPMurphi model (up to the discretisation of time and continuous variables)[1] `<domain_name>.m`. This, in turn, is compiled into an executable `<domain_name>_planner`.

PDDL+ Planning: The executable generated in the previous phase can now be invoked to start the planning/universal planning tasks. Several options can be specified to fine-tune this phase, as we describe below. The UPMurphi engine applies an explicit algorithm for building the system dynamics, and searches it through a forward search.

Plan output: Once the final plan(s) have been generated, they are written by default as PDDL+ plans, but the user can choose among a variety of different output formats (i.e., text, binary, and CSV), even in verbose mode.

Plans VALidation: UPMurphi is designed to automatically interface with the VAL plan validator, so that the generated plans are executed and validated. To enable this functionality, the user must install VAL separately.

This process is completely automatic. When running the planner, the user can specify the discretisation to be used for time and continuous variables, although the default settings can be used.

[1]The translation process is fully detailed in (Della Penna, Magazzeni, and Mercorio 2012).

3.1 UPMurphi Features

In this section we give details of the main UPMurphi features.

Disk-based Search. Starting with an initial discretisation (e.g., the one provided by default), the Discretise and Validate process should be iterated until a valid discretisation is used. However, the finer the discretisation, the larger the resulting state space, and this may lead to the state explosion phenomenon too early. To mitigate this issue, in this release UPMurphi employs the disk during the forward-search for storing both the state space generated so far and the current solution. Specifically, UPMurphi exploits the disk to store the full state description (i.e., the state values) whereas only the state signature is stored in memory using 40-bits for the encoding. This approach is beneficial for continuous domains as they often present a high number of discretised real values that grow the state size. Furthermore, it also allows trying several discretisation settings without affecting the number of states that can be visited during the search. UPMurphi is able to adapt its algorithm to increase or decrease the disk usage with respect to the user specified options and the size of the system under analysis. Furthermore, to avoid an excessive time overhead, the disk structures have been designed and implemented by taking into account their usage patterns, i.e., how (and how frequently) each structure is accessed during each phase of the planning process. This makes it possible to reduce the number of disk seek-and-read operations, which are the bottleneck of any disk algorithm, as seeks suffer from a latency time that is much higher than the actual read/write time. To give an example, UPMurphi privileges sequential read/writes, at the cost of duplicating some information and/or requiring more disk space, which is not a problem as large disks are nowadays very common.

Serialisation. Thanks to the disk-based exploration, UPMurphi can store the system graph, and access to it directly without having to load data into memory. This would allow one to *resume* the analysis by loading a (previously visited) system graph also on another machines.

State Compression. UPMurphi inherits from CMurphi a number of techniques to optimise the state representation (i.e., the *bit compression* and *hash-compaction*), and adapts them working even with the disk-based exploration described above.

Exploration Strategy. UPMurphi distinguishes between two planning modalities, namely (i) *Planning* in which a feasible plan is generated to reach a goal and (ii) *Universal Planning* that could be seen as a *collection* of plans (*aka* a set of policies) able to bring the system to the goal from *any* reachable state for which a plan exists in the given setting. Note that both these modalities support the specification of the optimality requirement for minimising the plan makespan[2].

[2]Here optimality is dependent on the discretisation and the finite horizon used for planning.

(Some) Exploration Settings. UPMurphi provides some other options that can be set for customising the state space exploration. They include the specification of the (highest) amount of memory to be used by the planning process, the enabling of a deadlock check (here intended as a non-goal state without any action applicable), and the specification of either the maximum number of BFS levels to explore or a maximum plan length.

Stepwise Exploration. UPMurphi allows the use of a step-by-step exploration useful for debugging purposes. At each step the user can specify which action has to be applied (among the ones applicable in the current state). The values of each PDDL+ predicate and fluent are shown to the user and the process iterates.

Discretisation settings. By default, UPMurphi discretises the time to 0.1 units while real scale and real fraction digits are set to 8 and 2 respectively. Although we found these values suitable for a number of planning problems, the user is free to specify different values by passing them as arguments while invoking the UPMurphi planner.

Supporting the PDDL+ semantics. UPMurphi has been designed to support the PDDL+ semantics according to the start-process-stop model introduced by (Fox and Long 2006), that works by transforming a durative-action into a chain of PDDL+ elements, namely: (i) a pair start/end actions that apply the discrete effects *at start* and *at end* of the action respectively; (ii) a process that applies the continuous change over the action execution (iii) and an event that checks whether all the *overall* durative-action conditions are satisfied during its execution. This motivated the need to properly model the processes and events interaction within UPMurphi to fully support the whole PDDL+ semantics. Figure 2 shows an example of how UPMurphi represents and reasons with the PDDL+ elements as a whole over the discretised time-line. The time is uniformly discretised in clock ticks (T) and a built-in action time-passing (TP) is responsible for advancing the time accordingly. Then, an action A_1 can activate a process $P1$ that, after three clock ticks, triggers event $E3$, which in turn activates process $P2$. UPMurphi is able to handle process/events interleaving as well as Timed Initial Literals and numeric Timed Initial Fluents. Clearly, a fine enough discretisation must be used in order to capture the happenings of TIFs and TILs. On the other hand, the time granularity of TIFs and TILs can be used as a guidance for choosing the initial time discretisation.

Limitations. The main limitation of UPMurphi is that currently there is no heuristic to guide the search, and a blind BFS is performed. UPMurphi also requires the PDDL+ domain to be typed for being processed. Finally, the only metric actually supported by UPMUrphi is `:minimize total-time`.

4 UPMurphi's Track Record

UPMurphi has been applied to several challenging PDDL+ domains. In the following we present some of them.

The Planetary Lander (Fox and Long 2006). A rover has to perform two observation tasks, that require either to perform the corresponding preparation tasks or to execute a single cumulative preparation task for both observations. The goal is to find a solution minimising the plan makespan. As a challenge, the domain presents a nonlinear system dynamics, deriving from the system equations (e.g., the energy generated by solar panels is influenced by the position of the sun), concurrence between processes (i.e., the rover may generate energy while is charging the battery), and processes/events interactions that may invalidate the plan due to tight resources and time constraints. In Figure 4 we show a log of a UPMurphi execution for the planetary lander. UPMurphi here searches for a feasible plan using up to 1Gb RAM and outputs the resulting plan in PDDL+ format. Finally, the plan is shown and saved into a file as well. More details on the use of UPMurphi in this domain can be found in (Della Penna, Magazzeni, and Mercorio 2012).

The Batch Chemical Plant. A production system is designed to obtain a concentrated saline solution recycling the remaining part for the next cycle. The system has many tightly connected components, regulated by a nonlinear dynamics (due to the equations modelling temperature and concentration variations), unknown action durations and a large set of safety constraints. UPMurphi has been used to synthesise a set of policies for many different initial production configurations. Note that thanks to the efficient use of the disk during the state space exploration, UPMurphi was able to generate up to 7 million plans. More details on the use of UPMurphi for the batch chemical plant can be found in (Della Penna et al. 2010).

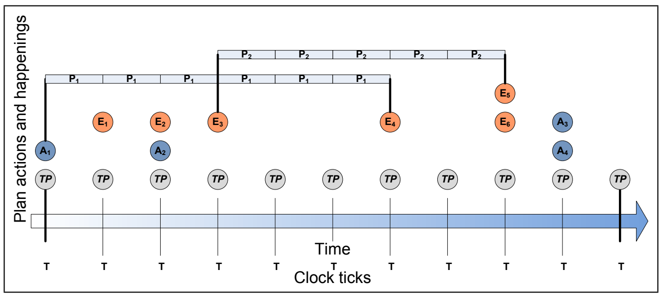

Figure 2: Processes in the discretised plan timeline

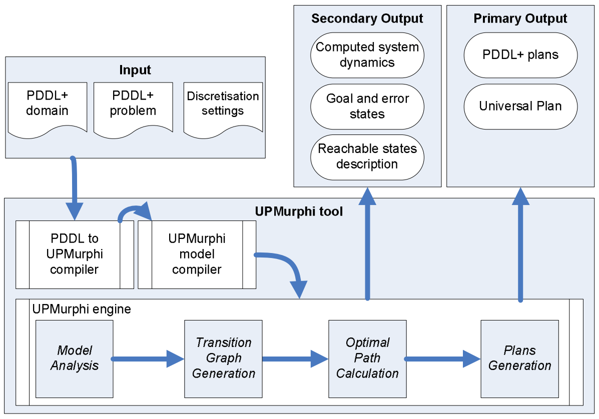

Figure 3: Overview of the UPMurphi architecture

```
cmd: ./planetary_lander_planner -search:f -m1000 -format:pddl

* Source domain: planetary_lander.pddl
* Source problem: planetary_lander_problem.pddl
* Planning Mode: Feasible Plan
* Output format: PDDL+
* Epsilon separation: 0.001
* Output target: "planetary_lander_problem_plan.pddl"

* UPMurphi Model: planetary_lander
* State size 498 bits (rounded to 64 bytes).
* Allocated memory: 1000 Megabytes

**  Time Discretisation = 0.1
**  Digits for representing the integer part of a real =  8
**  Digits for representing the fractional part of a real =  2

=== Analyzing model... ===============================
* Maximum size of the state space: 64726931 states.
  with states hash-compressed to 40 bits.
[0:0:2.87] states explored: 100000, actions fired: 101880
BFS level: 63, states queued: 7368, goals found: 0, errors: 0
max plan length: 6.30
34843.21 states/sec, 35498.26 actions/sec, 0.15\% memory used
....
[0:1:30.32] states explored: 3000000, actions fired: 3194058
BFS level: 170, states queued: 7657, goals found: 0, errors: 0
max plan length: 17.00
33215.23 states/sec, 35363.80 actions/sec, 4.63\% memory used

======================================================
Model exploration complete (in 92.76 seconds).
3282884 actions fired
1 start states
3084414 reachable states
1 goals found

=== Building model dynamics... =======================
Transition Graph mode: Memory Image
Model dynamics rebuilding complete (in 97.71 seconds).
3084414 states
3282884 transitions
out degree: min 0 max 10 avg 1.06

=== Finding paths... =========================
* Search Algorithm: Feasible Plan.

=== Collecting plans... ==============================
Plan(s) generation complete (in 98.41 seconds).
1 plans
plan length (actions): min 186 max 186 avg 186.00
plan duration (time): min 0 max 180 avg 180.00
plan weight: min 0 max 180 avg 180.00

=== Writing final results... =========================
* Output format: PDDL+
* Output target: "planetary_lander_problem_plan.pddl".

; --Plan #00001-------------------------
; -- Discretisation: 0.100----------------
; --------------------------------------
0.000: ( fullprepare dum unit1) [3.000]
3.001: ( observe2 unit1) [8.000]
11.002: ( observe1 unit1) [7.000]
; --------------------------------------
; --Plan duration: 18.002, weight: 0180----
; --------------------------------------
```

Figure 4: Log of a UPMurphi exeuction for the *Planetary Lander* domain

	Planetary Lander	*Chemical Plant*
State Space Size	10^{24}	10^{29}
Reachable States	$31,965,220$	$29,968,861$
Generated Plans	$5,309,514$	$7,154,464$

Table 1: Some statistics for Planetary Lander and Chemical Plant domains

Nonlinear generator. It is the continuous model of the well-known generator domain (Howey and Long 2003). A generator is powered by a fuel tank with a limited capacity of 60 fuel units and consumes one fuel unit per second. During the generator activity (modelled by the consume durative action), two fuel tanks of 25 fuel units each can be used to refuel it (through the refuel durative action). The refuelling process has a variable duration (i.e., its duration must be decided by the planner) and is described by the Torricelli's law, which makes the system dynamics nonlinear. Moreover, the domain also involves concurrency, since the consume and refuel actions take place continuously and concurrently, and are modelled through continuous processes. The goal is to make the generator run for 100 seconds. Table 2 summarises the results of the universal planning process after three Discretise and Validate iterations. We first considered a time discretisation of 5.0 and 2.5, both resulting in invalid solutions. We then refined the discretisation to 1.0 which proved to be fine enough for obtaining valid plans.

Time discretisation (sec)	5.0	2.5	1.0
State space size	10^{15}	10^{16}	10^{18}
Reachable states	$26,276$	$399,189$	$29,119,047$
Generated plans	0	$10,015$	$126,553$
Total synthesis time	3.7	20.71	$1,430.11$
Valid	NO	NO	**YES**

Table 2: Universal Plan statistics for the generator domain with time discretisation from 5.0 down to 1.0 seconds

Other examples were UPMurphi has been applied can be found in (Della Penna, Magazzeni, and Mercorio 2012), while works built on top of UPMurphi are described in (Fox, Long, and Magazzeni 2012; Campion et al. 2013; Boselli et al. 2014) and (Mezzanzanica et al. 2015).

5 Concluding Remarks

In this paper we presented the release of the PDDL+ planner UPMurphi, overviewing its main features that allow it to handle the full range of PDDL+ features, including nonlinear continuous processes, exogenous events, Timed Initial Literals and numeric Timed Initial Fluents. On a practical note, UPMurphi has been designed to work natively on Linux distributions (Ubuntu specifically), but it has been extensively tested on Windows with Cygwin environment, too. Finally, a MacOS compilation is also supported. Please refer to the UPMurphi web page (UPMurphi Web Page 2015) for download, installation instructions, more details, and news about UPMurphi development.

References

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle, M. 2014. Planning as model checking in hybrid domains. In *Proceedings of the Twenty Eighth Conference on Artificial Intelligence (AAAI-14)*. AAAI Press.

Bogomolov, S.; Magazzeni, D.; Minopoli, S.; and Wehrle, M. 2015. PDDL+ planning with hybrid automata: Foundations of translating must behavior. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS-15)*. AAAI Press.

Boselli, R.; Cesarini, M.; Mercorio, F.; and Mezzanzanica, M. 2014. Planning meets data cleansing. In *The 24th International Conference on Automated Planning and Scheduling (ICAPS-14)*, 439–443.

Bryce, D., and Gao, S. 2015. SMT-based nonlinear PDDL+ planning. In *Proceedings of the Twenty Nineth Conference on Artificial Intelligence (AAAI-15)*. AAAI Press.

Cached Murphi Web Page. 2006. `http://www.di.univaq.it/gdellape/murphi/cmurphi.php`.

Campion, J.; Dent, C.; Fox, M.; Long, D.; and Magazzeni, D. 2013. Challenge: Modelling unit commitment as a planning problem. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS-13)*.

Cimatti, A.; Giunchiglia, F.; Giunchiglia, E.; and Traverso, P. 1997. Planning via model checking: A decision procedure for *AR*. In *Recent Advances in AI Planning, 4th European Conference on Planning, (ECP'97)*, 130–142.

Coles, A. J., and Coles, A. I. 2014. PDDL+ planning with events and linear processes. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2012. COLIN: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research (JAIR)* 44:1–96.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: A tool for universal planning on PDDL+ problems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*. AAAI.

Della Penna, G.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2010. A PDDL+ benchmark problem: The batch chemical plant. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS-10)*, 222–225.

Della Penna, G.; Magazzeni, D.; and Mercorio, F. 2012. A universal planning system for hybrid domains. *Applied Intelligence* 36(4):932–959.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research (JAIR)* 27:235–297.

Fox, M.; Long, D.; and Magazzeni, D. 2012. Plan-based policies for efficient multiple battery load management. *J. Artif. Intell. Res. (JAIR)* 44:335–382.

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

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 294–301.

Li, H. X., and Williams, B. C. 2008. Generative planning for hybrid systems based on flow tubes. In *ICAPS*, 206–213.

McDermott, D. V. 2003. Reasoning about autonomous processes in an estimated-regression planner. In *ICAPS*, 143–152.

Mezzanzanica, M.; Boselli, R.; Cesarini, M.; and Mercorio, F. 2015. A model-based evaluation of data quality activities in KDD. *Inf. Process. Manage.* 51(2):144–166.

Molineaux, M.; Klenk, M.; and Aha, D. W. 2010. Planning in dynamic environments: Extending htns with nonlinear continuous effects. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI*.

Penberthy, J. S., and Weld, D. S. 1994. Temporal planning with continuous change. In *AAAI*, 1010–1015.

Shin, J.-A., and Davis, E. 2005. Processes and continuous change in a sat-based planner. *Artif. Intell.* 166(1-2):194–253.

UPMurphi Web Page. 2015. `https://github.com/gdellapenna/UPMurphi`.