PPS: An Efficient Java-based Simulator for Time-Discrete PDDL+

Enrico Scala[1], **Francesco Percassi**[2], **Mauro Vallati**[2]

[1]Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy
[2]School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom
enrico.scala@unibs.it, f.percassi@hud.ac.uk, m.vallati@hud.ac.uk

Abstract

The expressive power of PDDL+ is crucial in a wide range of real-world applications, where it is necessary to represent hybrid discrete-continuous changes and environmental dynamics. Given the complexity of the dynamics that can be modelled in PDDL+ and the scale of the problems involved, the ability to validate plans and simulate their trajectories is essential for assessing the accuracy of the models.

In this paper, we present PPS (PDDL Plus Simulator), a Java-based tool that enables seamless validation and simulation of PDDL+ plans under time-discrete semantics.

Code — https://github.com/hstairs/pps

Introduction

The ability to represent hybrid discrete-continuous changes is essential for leveraging automated planning to solve challenging problems from real-world applications. This necessity led to the design of the PDDL+ language (Fox and Long 2006), which supports the compact encoding of hybrid models involving mixed discrete/continuous effects, processes, exogenous events, and continuous change. PDDL+ has fostered the successful use of planning in a range of complex application domains, with notable recent examples including traffic signal optimisation (Kouaiti et al. 2024), personalised medicine (Alon et al. 2024), and unmanned aerial vehicle control (Kiam et al. 2020). PDDL+ is well-suited for simulating the evolution of real-world applications and their environments, thanks to its ability to concisely model complex dynamics and maintain a clear logical separation between the planning agent and its surroundings. This is particularly valuable in domains where the underlying laws and logic are well understood and can be represented to capture interactions and dynamics.

Validation and simulation are crucial processes in real-world planning applications, where they support the knowledge engineering task by playing a major role in assessing the quality of knowledge models, supporting the debugging of encodings, and ensuring the correctness of the overall planning system (Percassi, Scala, and Vallati 2022; McCluskey, Vaquero, and Vallati 2017). Informally, we refer to *validation* as the process of assessing the correctness of a plan and its ability to reach a goal state, and to *simulation* as the process of reproducing the evolution of the world according to a given plan, which may not achieve a specified goal or, in the general case, may not even have a defined goal at all. Despite the importance of validation and simulation, there is a limited number of tools and approaches available for PDDL+.

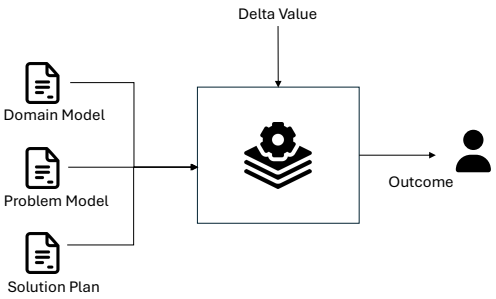

Figure 1: PPS receives in input the domain and problem specifications, the solution plan, and the delta value to be considered for the simulation and validation. The resulting output is then presented to the user in textual form.

To address this need, we present PPS (PDDL Plus Simulator), an off-the-shelf, Java-based tool that enables users to simulate and validate PDDL+ plans under time-discrete semantics (Percassi, Scala, and Vallati 2025). Since discretisation is a well-established approach for reasoning about such models, a tool that can validate and simulate the behaviour of plans under time-discrete semantics, using the same or different discretisation deltas, is particularly valuable for debugging knowledge models and assessing the accuracy of encoded processes and events.

The PPS System

PPS is implemented in Java and builds upon the capabilities of the JPDDLPLUS library[1], which serves as the backend of the ENHSP planning engine (Scala et al. 2016). Leveraging

[1]https://github.com/hstairs/jpddlplus.

the ENHSP backend ensures the use of a well-tested, reliable system widely adopted in PDDL+ planning. The overall framework is illustrated in Figure 1. PPS takes as input from the user the domain model file name, the problem file name, the solution plan, and the discretisation delta. If no delta value is provided, the default is set to 1.0.

The validity of a plan is assessed by simulating its execution at discrete timestamps that are multiples of delta. This simulation accounts for the effects of the actions included in the plan, any triggered events, and the active processes as time progresses. The simulation is represented in PPS as a trace, i.e., a sequence of PDDL states, where every state provides a complete representation of the state of the world. A provided plan is valid with respect to the provided domain and problem models if and only if, for every applied action, the preconditions are satisfied in the corresponding state from the plan projection, and the goal is achieved in the final state. For further details, the reader is referred to the work by Percassi, Scala, and Vallati (2022).

The main tasks performed by PPS can be summarised as follows:

- Parsing of the domain model, problem model, and solution plan files. During this phase, PPS also initialises the required data structures based on the characteristics of the given models.
- Grounding of the PDDL+ domain model in the presence of lifted representation. If the domain model is provided in a lifted form, PPS performs a grounding step using the capabilities of the JPDDLPLUS library. This library includes an efficient mechanism that minimises the grounded model size. Note that grounding may not be necessary for actions, since only those included in the plan are relevant. However, events and processes must be grounded independently, as they are not explicitly specified in the plan. Otherwise, if the model is already grounded, this step ensures the correct generation of data structures.
- Simulation and validation. This is the core activity of PPS. The simulation is done by executing the active processes and the events in stages, each determined by a pair of successive actions belonging to the plan. More precisely, PPS processes all actions in the plan sequentially. Each action is associated with the time it is planned to be executed. The simulation is run up to that time. Once that time is reached, the effects of the action are applied, provided the action is applicable, and the cycle continues. If the action is not applicable, the simulation stops.
- Goal distance estimation. For invalid plans, PPS provides a list of the unsatisfied goals, and estimates the goal distance using the AIBR heuristic (Scala et al. 2016).

Accepted Plan Syntax

Accepted input plans are expressed as time-triggered plans, that is, as a map from time to a list of actions. Each element of the map corresponds to a row in the file, and time is expressed in decimal notation. The action must be specified as a ground action from the domain model. Figure 2 shows an excerpt of a syntactically correct plan. Note also that PPS requires the specification of the end time of the plan, to account for running processes and events. This indicates when the plan is expected to meet the goal and is encoded via a special keyword `@PlanEND`. As it is apparent, anything listed after `@PlanEND` is discarded by PPS and not considered during simulation or validation.

```
0.0: (start_movement_increase L2 L3 ZAXES)
7.0: (stop_movement_increase L2 L3 ZAXES)
7.0: (start_movement_increase L4 L5 xyaxes)
...
21.0: (start_movement_decrease L5 L6 xyaxes)
22.0: @PlanEND
```

Figure 2: Excerpt of the accepted plan syntax. Please note the `@PlanEND` token that indicates when a goal state is reached.

```
...
Plan Invalid.
(T1_trainVoy_IE5_S_V) inapplicable at 290.0
 ==================================

Goal Is Not Reached!
Estimated Goal Distance 23734.0
(trainHasStopped T5) UNSAT
(trainHasEndedVoy T5) UNSAT
(trainHasStoppedAtStop T1 S_V) UNSAT
....
```

Figure 3: Excerpt of output for an invalid plan. As soon as an inapplicable action is identified, the process is stopped and the distance from the goal is heuristically estimated.

Mode of Operations

PPS can be used in two modes: validation only, and step-by-step simulation with trajectory output. In the former case, the output consists of the provided plan, its validity, and a check of whether the goal has been reached. Figure 3 shows an example of the output for an invalid plan.

In the step-by-step simulation with trajectory output, PPS provides the state of the world at each time step. This includes both numeric and boolean predicates and aims to provide a complete overview of the trajectory of the considered plan. This can be very useful for debugging model dynamics. Finally, the trajectory granularity is controlled by the delta value specified as input to the system.

Conclusion

Despite the growing importance of automated planning techniques in real-world applications, support for the validation and simulation of complex hybrid PDDL+ models remains very limited. In this context, PPS is a novel Java-based tool that simulates and validates PDDL+ plans under time-discrete semantics. Future work will focus on developing modules to support the visualisation of validated plans and on suggesting methods to repair invalid ones.

Acknowledgments

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/Z00005X/1].

References

Alon, L.; Weitman, H.; Shleyfman, A.; and Kaminka, G. A. 2024. Planning to be Healthy: Towards Personalized Medication Planning. In *ECAI*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, 4232–4239.

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Artif. Intell. Res.*, 27: 235–297.

Kiam, J. J.; Scala, E.; Jávega, M. R.; and Schulte, A. 2020. An AI-Based Planning Framework for HAPS in a Time-Varying Environment. In *ICAPS*, 412–420.

Kouaiti, A. E.; Percassi, F.; Saetti, A.; McCluskey, T. L.; and Vallati, M. 2024. PDDL+ Models for Deployable yet Effective Traffic Signal Optimisation. In *ICAPS*, 168–177.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering Knowledge for Automated Planning: Towards a Notion of Quality. In *K-CAP*, 14:1–14:8.

Percassi, F.; Scala, E.; and Vallati, M. 2022. The Power of Reformulation: From Validation to Planning in PDDL+. In *ICAPS*, 288–296.

Percassi, F.; Scala, E.; and Vallati, M. 2025. On the Notion of Plan Quality for PDDL+. In *ICAPS*, 102–111.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2016. Interval-Based Relaxation for General Numeric Planning. In *ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 655–663.