On the Reformulation of Discretised PDDL+ to Numeric Planning (Extended Abstract)*

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

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

Abstract

PDDL+ is an expressive planning formalism that enables the modelling of hybrid discrete-continuous domains. The resulting models are notoriously difficult to cope with, and few planning engines are natively supporting PDDL+. To foster the use of PDDL+, this paper revisits a set of recently proposed translations allowing to reformulate a PDDL+ task into a PDDL2.1 one. Such translations permit the use of a wider set of engines to solve complex hybrid problems.

Introduction

Automated planning is a solid branch of artificial intelligence that aims at designing methodologies for the automated synthesis of decisions capable to transform a given state, i.e., the initial state, into the desired state, i.e., the goal state. In real-world scenarios, such a synthesis has to take into account that hybrid systems can be quite complex, as they are characterised by the coexistence of a discrete and continuous dynamic.

Hybrid systems are modelled in the planning community using the PDDL+ formalism (Fox and Long 2006). PDDL+ provides a representation that puts together an agent, via an action-oriented formalisation, with an explicit representation of the environment and its exogenous dynamics. PDDL+ problems consist in finding a number of time-stamped actions along a continuous (or discrete) timeline, whilst conforming to the instantaneous and continuous changes prescribed by events and processes, respectively. Valid plans are those where actions have their preconditions satisfied when they are executed, and the final state satisfies the goal. A major challenge in tackling PDDL+ planning problems concerns the ability of planning whilst tracking numeric variables that can change as an effect of processes and events through time; only a restricted set of planning engines natively support PDDL+ in an effective manner.

To overcome this problem, we show how to leverage the consolidated approach of reformulation by translating problems expressed in a very expressive and rich formalism into ones expressed in less complicated formalism (Percassi and Gerevini 2019; Bonassi et al. 2021; Percassi, Scala, and Vallati 2022). In particular, here we report on a recent line of research that investigate translations from PDDL+ problem into discrete numeric planning (Percassi, Scala, and Vallati 2021a,b).

We give an overview of the theoretical properties of the different translations, and novel experimental results obtained using optimal numeric planners. We do this to provide a crisper and more distilled picture of how different translations affect the search performed by planning techniques.

From Discretised PDDL+ To PDDL2.1

Percassi, Scala, and Vallati (2021b) proposed an exponential (EXP) and a polynomial (POLY) translation to increase the pool of planning engine that can tackle PDDL+ problems. The aim is to transform a time discretised PDDL+ into a simpler problem expressed in PDDL2.1 (Fox and Long 2003).

Both schemata compile processes and events into the agent's actions. EXP does so anticipating the occurrence of all contexts, i.e., a subset of processes that can be active in a state, with a single simulation action having several conditional effects. This has the advantage of keeping the depth of the search tree limited, but the drawback of having exponentially many conditional effects, one per each context.

POLY avoids such an exponential behaviour by unrolling all processes into several consecutive actions. This sequence is structured as $\langle start, seq(A_P), end\rangle$, where *start* initialises the unrolling, and each element in $seq(A_P)$ is an arbitrary total order over actions from A_P; these actions are associated to each numeric continuous effect of the processes. Each such action applies if the corresponding process holds when *start* is applied. Finally, *end* closes the unrolling. POLY prevents the exponential blow-up but makes the search tree much deeper. To produce a more compact encoding, a third translation, namely POLY^-, has been proposed (Percassi, Scala, and Vallati 2021a). Intuitively, POLY^- leverages the advantages of EXP, using a single action for simulating the advance of a discrete quantum of time, and avoids the exponential blow up with a schema that ignores some of the possible transitions. This results in an approach that, differently from both EXP and POLY (which are both sound and complete), is sound but complete only for a syntactic subclass of PDDL+ tasks.

In PDDL+ problems, there can be multiple processes that

*The extended abstract reports on the work previously appeared in the papers (Percassi, Scala, and Vallati 2021b,a).

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

h^{blind}	Coverage			Time			Exp. Nodes ($\times$1000)		
Domain	POLY	POLY$^-$	EXP	POLY	POLY$^-$	EXP	POLY	POLY$^-$	EXP
ROVER	**1**	**1**	**1**	61.2	**20.3**	33.7	12013	**3622**	**3622**
LIN-CAR	**10**	**10**	**10**	3.3	3.0	**2.9**	200	**27**	**27**
LIN-GEN✗	**1**	**1**	**1**	**3.1**	3.2	3.3	84	25	**23**
BAXTER✗	4	**7**	0	51.3	**9.2**	—	1294	**123**	—
OT-CAR	**5**	**5**	**5**	16.8	**3.7**	4.6	2644	**252**	**252**
DESCENT✗	2	0	**3**	19.6	—	**8.5**	444	—	**134**
HVAC	0	0	0	—	—	—	—	—	—
Σ	23	**24**	20						

h^{max}	Coverage			Time			Exp. Nodes ($\times$1000)		
Domain	POLY	POLY$^-$	EXP	POLY	POLY$^-$	EXP	POLY	POLY$^-$	EXP
ROVER	**1**	**1**	**1**	121.8	**28.7**	54.1	11898	**3513**	**3513**
LIN-CAR	**10**	**10**	**10**	3.2	**2.9**	3.3	40	**25**	27
LIN-GEN✗	1	**2**	**2**	3.5	2.7	**2.6**	84	5	**2**
BAXTER✗	4	**7**	0	44.8	**6.5**	—	1041	**68**	—
OT-CAR	**5**	**5**	**5**	16.2	**5.2**	5.5	1271	**278**	**278**
DESCENT✗	2	0	**3**	13.6	—	**8.7**	246	—	**112**
HVAC	0	**16**	**16**	—	**3.5**	**3.5**	—	**7**	**7**
Σ	23	**41**	37						

Table 1: Performance achieved by h^{blind} (*left*) h^{max} (*right*) when run on models generated using the POLY, POLY$^-$, EXP translations with $\delta = 1$. ✗ is used to indicate domain models for which there is no guarantee of optimality for the solution found over the POLY$^-$ models.

	POLY	EXP	POLY$^-$
soundness	✓ Lemma 2 ($\Leftarrow$) in Percassi et al., 2021b	✓ Lemma 1 ($\Leftarrow$) in Percassi et al., 2021b	✓ Prop. 1 in Percassi et al., 2021a
completeness	✓ Lemma 2 ($\Rightarrow$) in Percassi et al., 2021b	✓ Lemma 1 ($\Rightarrow$) in Percassi et al., 2021b	✗ (general case) Prop. 2 in Percassi et al., 2021b
$size_Z$	N_{tot}	$2^{\lvert P \rvert} - 1$	$\lvert P \rvert$

Table 2: Properties of *soundness* and *completeness* and size of the numeric translated tasks obtained through $Z \in \{\text{EXP}, \text{POLY}, \text{POLY}^-\}$ for a PDDL+ task Π.

additively affect the same numeric variable. The generation of the successor state when time flows in POLY$^-$ occurs in parallel, whereas in POLY it occurs sequentially. The sequentialisation allows the encapsulation of all the ways in which the state can evolve over time according to the processes, while parallelisation does not. To be specific, the invalid transitions in POLY$^-$ concern those states in which there are at least two processes affecting the same numeric variable. This limitation causes the incompleteness of POLY$^-$.

All the schemata produce encodings having a linear or a polynomial number of variables and actions in the size of the original PDDL+ task. What controls the size of the resulting encoding is how the contexts are enclosed in the compiled actions, which we denote as $size_Z$. Since EXP explicitly enumerates all the possible contexts except the empty one, the size of the resulting planning task is as follows: $size_{\text{EXP}} = 2^{|P|} - 1$. In POLY, all possible contexts are implicitly encoded in $\langle start, seq(A_P), end\rangle$ within the A_P operators, and then $size_{\text{POLY}} = |A_P| = N_{tot}$ where N_{tot} is a constant denoting all continuous numeric effects of the PDDL+ task. POLY$^-$ explicitly encloses a restricted number of contexts in one action. This restriction allows to get $size_{\text{POLY}^-} = |P|$. Percassi, Scala, and Vallati (2021b; 2021a) discussed the property of soundness and completeness for the proposed translations by studying the relationship between the solution space of the PDDL+ task and the numeric one. Table 2 summarises all the discussed results.

Experimental Results

To empirically compare the discussed translations we use ENHSP20 (Scala et al. 2020), which allows us to tackle numeric planning with non-linear dynamics, and provides the use of customised search strategies. We tested two optimal heuristics, i.e., h^{blind} and h^{max} (Scala, Haslum, and Thiébaux 2016). As benchmarks we considered those used in (Percassi, Scala, and Vallati 2021b), plus two non-linear domains, i.e., the well-known DESCENT and HVAC. All experiments were run on an Intel Xeon Gold 6140M CPU with 2.30 GHz. For each instance, we allotted 180 seconds and limited memory to 8 GB.

Table 1 shows the results. In domains characterised by few processes, POLY$^-$ and EXP are preferable to POLY. Conversely, when the PDDL+ tasks include numerous processes, as in BAXTER, EXP becomes infeasible. The overhead introduced in the search by POLY, due to the lengthening of the plans, prevents finding solutions in HVAC. Finally, the incompleteness of POLY$^-$, although often advantageous in terms of speedup, may lead to cases where all the solutions are pruned from the search space, as in DESCENT.

Discussion

EXP is a didactical baseline and it has the limitation of being infeasible for PDDL+ problems involving many processes. On the contrary, POLY, in which the time flow involves the execution of a polynomial number of actions, leads to a greater search effort. However POLY produces polynomial encoding, making the translation more feasible than EXP.

POLY$^-$ produces numeric tasks having the same structure as those produced by EXP, and then it is advantageous in terms of search, but its incompleteness allows it to be used safely only in restricted cases. Overall EXP and POLY$^-$ obtain an appreciable result in the optimal context but the latter loses the guarantee on the optimality of the solutions found in some domains.

We see several avenues for future work. We are interested in exploring the online selection of the best translation to be used according to the structure of the problem considered. We are also interested in investigating the possibility of automatically combining different translations, and further exploiting the structure of the problem.

Acknowledgments

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/T041196/1].

References

Bonassi, L.; Gerevini, A. E.; Percassi, F.; and Scala, E. 2021. On Planning with Qualitative State-Trajectory Constraints in PDDL3 by Compiling them Away. In *Proc. of ICAPS*, 46–50.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *JAIR*, 20: 61–124.

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

Percassi, F.; and Gerevini, A. E. 2019. On Compiling Away PDDL3 Soft Trajectory Constraints without Using Automata. In *Proc. of ICAPS*, 320–328.

Percassi, F.; Scala, E.; and Vallati, M. 2021a. A Sound (but Incomplete) Polynomial Translation from Discretised PDDL+ to Numeric Planning. In *Proc. of AIxIA*.

Percassi, F.; Scala, E.; and Vallati, M. 2021b. Translations from Discretised PDDL+ to Numeric Planning. In *Proc. of ICAPS*, 252–261.

Percassi, F.; Scala, E.; and Vallati, M. 2022. The Power of Reformulation: PDDL+ Validation Through Planning. In *Proc. of ICAPS*.

Scala, E.; Haslum, P.; and Thiébaux, S. 2016. Heuristics for Numeric Planning via Subgoaling. In *Proc. of IJCAI*, 3228–3234.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *JAIR*, 68: 691–752.