Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

Translations from Discretised PDDL+ to Numeric Planning

Francesco Percassi,[1] **Enrico Scala,**[2] **Mauro Vallati**[1]
[1] School of Computing and Engineering, University of Huddersfield, UK.
[2] University of Brescia, Italy
f.percassi@hud.ac.uk, enrico.scala@unibs.it, m.vallati@hud.ac.uk

Abstract

Hybrid PDDL+ models are amongst the most advanced models of systems and the resulting problems are notoriously difficult for planning engines to cope with. An additional limiting factor for the exploitation of PDDL+ approaches in real-world applications is the restricted number of domain-independent planning engines that can reason upon those models. With the aim of deepening the understanding of PDDL+ models, in this work we study a novel mapping between a time discretisation of PDDL+ and numeric planning as for PDDL2.1 (level 2). The proposed mapping not only clarifies the relationship between these two formalisms, but also enables the use of a wider pool of engines, thus fostering the use of hybrid planning in real-world applications. Our experimental analysis shows the usefulness of the proposed translation, and demonstrates the potential of the approach for improving the solvability of complex PDDL+ instances.

Introduction

The availability of domain-independent planning engines is fostering the use of planning in a wide range of applications. This is despite the complexity issues inherent in plan generation, which are exacerbated by the separation of planner logic from domain knowledge. A major advantage of the separation of planning logic from domain knowledge lies in the fact that, given a standard language to be used for input and output, the two components can be interchanged in a modular way, without affecting the other component, and with no negative repercussions on the overall application framework where the planning system is usually embedded.

This modular approach promotes the use of reformulation techniques, which can automatically re-formulate or re-represent the domain knowledge and/or the problem instance in order to increase the efficiency of the planning logic module and increase the scope of problems solved. The general idea is to develop reformulation techniques that are agnostic with regards to the domain knowledge and to the planning engine, and use them to form a wrapper around the planning engine, improving its performance for the domain in which it is applied. The transformation is then reversed after a solution has been found, such that the solution is rephrased in the original formulation language. A well-known class of reformulation approaches aims at translating a model from the original input language, to a different one. The idea is usually to remove the use of some poorly supported features of the language (see for instance (Ceriani and Gerevini 2015; Percassi and Gerevini 2019)) or to re-represent the problem in a less expressive language. The latter strategy has the advantage of increasing the number of planning engines that are able to reason upon the planning problem, at the cost of making the model extremely hard to read for human experts. Well-known examples of this approach include the translation of conformant planning problems into classical problems (Taig and Brafman 2013), the re-representation of uncertainty in conformant planning problems (Palacios and Geffner 2009), and the translation of complex temporal aspects in PDDL2.1 (Cooper, Maris, and Rgnier 2010).

In this paper we introduce a reformulation approach for translating discretised PDDL+ problems to PDDL2.1 problems. Hybrid PDDL+ models are amongst the most advanced models of systems and the resulting problems are notoriously difficult to cope with. Further, a limited number of planning engines are able to parse PDDL+ models.

More precisely, we study two translations. The first translation leads to a numeric planning problem which is exponentially larger than the PDDL+ input but preserves the number of discrete transitions. The second one keeps the resulting formulation polynomial but requires more transitions to generate a solution. We start off by revisiting Shin and Davis (2005)'s formalisation, which lets us to formalise the problem without the need of going through Fox and Long (2003)'s hybrid automaton interpretation, and also crisply formalises the connection between the continuous-time representation, and its discretisation. We formally present the two translations, and also show how these can be extended to encode cascade of events emulated by actions. We study both translations theoretically and empirically. In particular, we validate the resulting formulations against a set of challenging benchmark domains, including real-world applications, and well-known planning engines. Our results indicate that the proposed translations can unlock the use of PDDL2.1 planning engines for tackling hybrid PDDL+ problems, with the clear advantage of drastically expanding the number of approaches that can be used to solve a problem.

Problem Formalisation

In this section we formalise the problem of PDDL+ (Fox and Long 2006), and the problem of numeric planning as the one that can be specified in PDDL2.1 (level 2) (Fox and Long 2003), hereinafter simply referred to as PDDL2.1. We first describe the syntax of our problems, then detail the semantics of PDDL+ under both continuous and discrete time, and only sketch the PDDL2.1's one in the interest of space. Our discussion follows Shin and Davis (2005)'s formalization and terminology[1] in a way that is instrumental for our work. We detail our problems using propositional formulas over comparisons and Boolean variables[2]. A comparison is $\xi \bowtie 0$ where ξ is a mathematical expression, and $\bowtie \in \{\leq, <, =, >, \geq\}$.

Definition 1 (PDDL+ problem)**.** *A PDDL+ problem Π is the tuple $\langle F, X, I, G, A, E, P\rangle$ where:*

- *F and X are the sets of Boolean and numeric variables, respectively.*
- *I and G are the description of the initial state, expressed as a full assignment to all variables in X and F, and the description of the goal, expressed as a formula, respectively.*
- *A and E are the sets of actions and events, respectively. Actions and events are pairs $\langle p, e\rangle$ where p is a propositional formula and e is a set of conditional effects of the form $c \triangleright e$ where (i) c is a formula and (ii) e is a set of Boolean ($f = \{\bot, \top\}$) or numeric ($\langle\{ass, inc, dec\}, x, \xi\rangle$ where ξ is an expression over X) assignments.*
- *P is a set of processes. A process is a pair $\langle p, e'\rangle$ where p is a formula and e' is a set of numeric continuous effects expressed as pairs $\langle x, \xi\rangle$ with the meaning that ξ represents the time-derivative of x, with $x \in X$.*

Let $a = \langle p, e\rangle$ be an action or an event or a process, we use *pre*(a) to denote the precondition p of a, and *eff*(a) the effect e of a. In the following we will use a, ρ and ε for denoting an action, process and event, respectively.

A plan for a PDDL+ is an ordered set of timed actions plus a time envelope, organised formally as following.

Definition 2 (PDDL+ plan)**.** *A PDDL+ plan π_t is a pair $\langle \pi, \langle t_s, t_e\rangle\rangle$ where $\pi = \langle\langle a_0, t_0\rangle, \langle a_1, t_1\rangle, ..., \langle a_{n-1}, t_{n-1}\rangle\rangle$, with $t_i \in \mathbb{R}$, is a sequence of time-stamped actions and $\langle t_s, t_e\rangle$, with $t_s, t_e \in \mathbb{R}$, is a real interval, called envelope, within which π is performed.*

Definition 3 (PDDL2.1 problem)**.** *A PDDL2.1 problem Π is the tuple $\langle F, X, I, G, A, c\rangle$ where all elements are as for PDDL+, yet there are neither processes nor events and c associates to each action a rational cost.*

Definition 4 (PDDL2.1 plan)**.** *A PDDL2.1 plan is a sequence of actions $\langle a_0, ..., a_{n-1}\rangle$. The cost of the plan π is the sum of all action costs in π, cost$(\pi) = \sum_{a\in\pi} c(a)$.*

[1] As Shin and Davis (2005), we idealise the execution of actions and events to be truly instantaneous transitions as long as an order is imposed among transitions sharing the same time clock.

[2] With abuse of notations we also use positive and negative literals as shortcut for $f = \{\top\}$.

Intuitively, a PDDL+ problem consists in finding a number of actions along a potentially infinite timeline, whilst conforming to a number of processes and events that may change the state of the world in a continuous or an instantaneous manner as time goes by. Actions have to be applicable when they are scheduled, and have to be such that the agent is into a state in which the goal is satisfied when the plan is finished. In technical terms, PDDL+ prescribes events and processes to be interpreted as *must* discrete and continuous transitions along a potentially infinite timeline, while actions are *may* transitions. A PDDL2.1 problem is the variant where there is no time, and we only seek for a sequence of actions that starts from some initial state and yields a state satisfying the goal. In the rest of the section we focus on the semantics of PDDL+, and in particular on its temporal dimension. We assume the reader is familiar with notions of action/event applicability, and simply use $\gamma(s, \cdot)$ for the state resulting by applying either an action/event ($\gamma(s, a)$) or a sequence of action/events ($\gamma(s, \langle a_0, \ldots, a_n\rangle)$) in state s. For details on PDDL2.1, we refer the reader to Fox and Long (2003). Moreover, we exploit the widely shared assumptions of boundness adopted in PDDL+ problems (Fox and Long 2006). Importantly we assume there always is a finite (possibly empty) and unique acyclic sequence of events that can be triggered, and there is a bound on the number of spontaneous changes of processes over a closed interval. We start off by introducing the notion of time points, intervals and histories over intervals. We use these to define the validity of a PDDL+ plan which we interpret both on a continuous and a discrete timeline, both based on the notion of a projection of a plan.

A time point T is a pair $\langle t, n\rangle$ where $t \in \mathbb{R}$ and $n \in \mathbb{N}$. Time points over $\mathbb{R} \times \mathbb{N}$ are ordered lexicographically, i.e., let $\langle t_i, n_i\rangle$ and $\langle t_j, n_j\rangle$ be two time-points, $\langle t_i, n_i\rangle < \langle t_j, n_j\rangle$ iff either $t_i < t_j$ or $t_i = t_j$ and $n_i < n_j$. Let T_i and T_j be two time points, a closed (open) time interval $\mathcal{I} = [T_i, T_j]$ ((T_i, T_j)) is the non empty set $\mathcal{I} = \{T,\ T_i \leq T \leq T_j\}(\{T,\ T_i < T < T_j\})$.

A history $\mathcal{H}$ over $\mathcal{I} = [T_s, T_e]$ maps each time point in $\mathcal{I}$ into a situation. A "situation at time T" is the tuple $\mathcal{H}(T) = \langle \mathcal{H}_A(T), \mathcal{H}_s(T)\rangle$, where $\mathcal{H}_A(T)$ is the set of actions executed at time T and $\mathcal{H}_s(T)$ is a state, i.e., an assignment to all variables in X and F at time T. We denote by $\mathcal{H}_s(T, v)$ and $\mathcal{H}_s(T, \xi)$ the value assumed in the state at time T by $v \in F \cup X$ and by a numeric expression ξ, respectively. $E_{trigg}(T)$ indicates the set of active events in T. T is a significant time point of $\mathcal{H}$ over $[T_s, T_e]$, iff either $\mathcal{H}_A(T) \neq \emptyset$ or a process has started or stopped in T or $E_{trigg}(T) \neq \emptyset$ or $T = T_s$ or $T = T_e$ or there has been a discrete change just before. A history $\mathcal{H}$ is monotonous over $\mathcal{I}_t = (t1, t2)$, where $t1, t2 \in \mathbb{R}$, if there is no $t \in \mathcal{I}_t$ such that $\mathcal{H}(\langle t, n\rangle)$ is a significant point of $\mathcal{H}$. Note that, for each $\langle t, n\rangle$ such that $t \in (t1, t2)$, the set of active processes does not change and the set of triggered events is empty. We denote the set of active processes over $\mathcal{I}_t$ as $\mathcal{C}(\mathcal{I}_t) = \{\rho \mid \rho \in P \text{ and } \mathcal{H}_s(\langle t1, n1\rangle) \models pre(\rho)\}$, where $n1$ is a sufficiently large natural number beyond which the state, for the clock equal to $t1$, is stable.

In the following definitions we assume Π =

$\langle F, X, I, G, A, E, P\rangle$, and $\pi_t = \langle \pi, \langle t_s, t_e\rangle\rangle$ implicit.

Definition 5 (PDDL+ plan projection)**.** *Let $\mathcal{H}^\pi$ be a history, let I be an initial state and let π_t be a* PDDL+ *plan. We say that $\mathcal{H}^\pi$ is a projection of π_t which starts in I iff $\mathcal{H}^\pi$ induces a sequence of significant time points $T_\mathcal{H} = \langle T_0 = \langle t_s, 0\rangle, \cdots, T_m = \langle t_e, n_m\rangle\rangle$ such that $\mathcal{H}^\pi$ is defined over $\mathcal{I} = [T_0, T_m]$ with $\mathcal{H}^\pi_s(T_0) = I$, $\mathcal{H}^\pi_A(T_m) = \emptyset$, $E_{trigg}(T_m) = \emptyset$ and, for all $0 \leq i < m$, the following rules hold:*

- **R1** *$E_{trigg}(T_i) \neq \emptyset$ iff $\mathcal{H}^\pi_s(T_{i+1}) = \gamma(\mathcal{H}^\pi_s(T_i), E_{trigg}(T_i))$, $\mathcal{H}^\pi_A(T_i) = \emptyset$, $t_{i+1} = t_i$ and $n_{i+1} = n_i + 1$;*
- **R2** *$\mathcal{H}^\pi_A(T_i) \neq \emptyset$ iff $\mathcal{H}^\pi_s(T_{i+1}) = \gamma(\mathcal{H}^\pi_s(T_i), \mathcal{H}^\pi_A(T_i))$, $E_{trigg}(T_i) = \emptyset$, $t_{i+1} = t_i$ and $n_{i+1} = n_i + 1$;*
- **R3** *for each $\langle a_i, t_i\rangle, \langle a_j, t_j\rangle \in \pi$, with $i < j$ and $t_i = t_j$ there exists $T_k, T_z \in T_\mathcal{H}$ such that $a_i \in \mathcal{H}^\pi_A(T_k)$ and $a_j \in \mathcal{H}^\pi_A(T_z)$ where $t_k = t_z = t_i$ and $n_k < n_z$;*
- **R4** *$\mathcal{H}^\pi$ is monotonous over $\mathcal{I}_t = (t_i, t_{i+1})$, iff $t_{i+1} > t_i$ and for each $t \in \mathcal{I}_t$ and for each $x \in X$, we have that:*
  - *$\mathcal{H}^\pi_s(\langle t, 0\rangle, x)$ is continuous and differentiable;*
  - *for each $x \in X$ we have that:*

$$\frac{d\mathcal{H}^\pi_s(\langle t, 0\rangle, x)}{dt} = \sum_{\substack{\langle x', \xi\rangle \in \mathit{eff}(\rho),\ x' = x \\ \rho \in \mathcal{C}(\mathcal{I}_t)}} \mathcal{H}^\pi_s(\langle t, 0\rangle, \xi)$$

  - *for each $x \in X$ we have that $\mathcal{H}^\pi_s(\langle t_{i+1}, 0\rangle, x) = \lim_{t \to t^-_{i+1}} \mathcal{H}^\pi_s(\langle t, 0\rangle, x)$, and values of unaffected variables persist up to t_{i+1} (frame-axiom).*

Definition 6 (Valid PDDL+ plan)**.** *π_t is valid plan for Π iff $\mathcal{H}^\pi_s(T_m) \models G$ and, for each $a \in \mathcal{H}^\pi_A(T)$ with $T \in \mathcal{I}$, $\mathcal{H}^\pi_s(T) \models \mathit{pre}(a)$.*

Given a mathematical expression ξ and a rational value $\delta \in \mathbb{Q}$ we denote with $\Delta(\xi, \delta)$ the discretised expression.

Definition 7 (PDDL+ plan discrete projection)**.** *Let $\delta \in \mathbb{Q}$, let $\mathbb{H}^\pi$ be a history, let I be an initial state and let π_t be a* PDDL+ *plan. We say that $\mathbb{H}^\pi$ is a discrete projection of π_t which starts in I iff $\mathbb{H}^\pi$ induces the significant time points $T_\mathbb{H} = \langle T_0 = \langle t_0, n_0\rangle, \cdots, T_m = \langle t_m, n_m\rangle\rangle$ where either $t_{i+1} = t_i + \delta$ or $t_{i+1} = t_i$, and all rules as for Def. 5 apply, except for R4 that becomes:*

- **R4** *for each pair of contiguous significant time points $T_i = \langle t_i, n_i\rangle$, $T_{i+1} = \langle t_{i+1}, 0\rangle$ such that $t_{i+1} = t_i + \delta$, the value of each numeric variable $x \in X$ is updated as:*

$$\mathbb{H}^\pi_s(T_{i+1}, x) = \mathbb{H}^\pi_s(T_i, x) + \sum_{\substack{\langle x', \xi\rangle \in \mathit{eff}(\rho),\ x' = x \\ \rho \in \{\rho \in P,\ \mathbb{H}^\pi_s(T_i) \models \mathit{pre}(\rho)\}}} \mathbb{H}^\pi_s(T_i, \Delta(\xi, \delta))$$

  and values of unaffected variables remain unchanged (frame-axiom).

Note that plans featuring actions at non discrete time points do not admit any projection, and are therefore ill-defined under discrete interpretation.

In the following we provide a more intuitive description of R1–R4 of Defs. 5 and 7. R1 (R2) states that if an action (event) is executed (triggered) in a significant time point $T1 = \langle t, n\rangle$, then there necessary exists a successor of $T1$, i.e., $T2 = \langle t, n+1\rangle$ having the same clock t and the step increased by one unit, i.e., $n+1$; the successor state associated to $T2$ is calculated by simply applying the discrete effects of the action (event). R3 is used to enforce how actions of a PDDL+ plan π are projected over an history, preserving their original ordering in case they share the same time-stamp in π. R4 is used to enforce how a numeric variable changes continuously over time according to the active processes in those "monotonous" temporal intervals in which "nothing happens (there is no action/event executed/triggered and there is no process which starts/ends).

Definition 8 (δ Discrete valid PDDL+ plan)**.** *π_t is a valid plan for Π under δ discretisation iff $\mathbb{H}^\pi_s(T_m) \models G$ and, for each $T \in \mathcal{I}$ such that $\mathbb{H}^\pi_A(T) \neq \emptyset$, then $\mathcal{H}^\pi_s(T) \models \mathit{pre}(a)$.*

From PDDL+ to PDDL2.1

PDDL+ problems differ from PDDL2.1 for the presence of processes and events. In this section we show a translation schema that transforms all these structures into regular actions, and adds additional predicates to their preconditions and to the goal so that every valid plan that is found in the process and event-free translation, retains its validity on the discretised version of the PDDL+ problem it has been generated from. We do so by explicitly formulating a simulation action that lets the planning engine wait and observe the state of the world for a given amount of time. Instead, when the planner picks some action, time does not flow; rather the state is instantaneously modified by the planning engine.

To make this operational, there are a number of challenges to pursue: (i) we need to capture what processes are active in a given state so that the time-discretised continuous update of the state consistently reflects what the dynamical specification of the system prescribes; (ii) we need to take care of the potentially complex cascade of events that may be triggered for each encountered state.

We proceed in a modular fashion. In what follows we firstly present a solution to point (i) with a (straightforward) exponential encoding first, and then with a more sophisticated polynomial translation. Both are guaranteed to work for event-free PDDL+ problems. Then we face point (ii) by showing how also events can be translated into actions with conditional effects with a translation step that is modular to how we tackle point (i).

Exponential Translation

Given an event-free PDDL+ problem $\Pi = \langle F, X, I, G, A, P, \emptyset\rangle$, we define a context $\mathcal{C}$ to be a non-empty subset of processes, and denote with $\mathcal{P}^+(P)$ the set of non-empty subsets of P, that is the set of all possible contexts.

For an event-free PDDL+ problem Π, the exponential translation generates a PDDL2.1 problem $\Pi' = \langle F, X, I, G, A \cup \{\mathit{SIM}\}, c\rangle$, discretised in $t = \delta$. Π' is almost identical to Π but for the absence of processes and the presence of the special action *SIM* playing the role of the simulator, i.e., what changes when time goes forward. *SIM* is defined as follows:

$$pre(SIM) = \top$$
$$eff(SIM) = \bigcup_{\mathcal{C} \in \mathcal{P}^+(P)} \{contpre(\mathcal{C}) \triangleright conteff(\mathcal{C})\}$$

where

$$contpre(\mathcal{C}) = \bigwedge_{\rho \in P \setminus \mathcal{C}} \neg pre(\rho) \wedge \bigwedge_{\rho \in P \cap \mathcal{C}} pre(\rho)$$
$$conteff(\mathcal{C}) = \bigcup_{x \in X} \{\langle inc, x, \sum_{\langle x', \xi \rangle \in \underset{\rho \in \mathcal{C}}{eff(\rho)},\ x'=x} \Delta(\xi, \delta) \rangle\}$$

Intuitively, the action *SIM* organizes all possible contexts within a unique action, delegating to each conditional effect (i) the conditions under which a context is triggered and (ii) the consequences that such a context has on the state after some time δ has passed. Point (i) is formalised conjoining two conjunctions: the first ensures that no other process of some other context has its precondition satisfied ($\bigwedge_{\rho \in P \setminus \mathcal{C}} \neg pre(\rho)$); the second ensures that all the preconditions of a given context are satisfied ($\bigwedge_{\rho \in P \cap \mathcal{C}} pre(\rho)$). Let x be some numeric variable of our problem, point (ii) is obtained by summing the contribution of each process within the context.

Ultimately, we reflect the effect of the time passing action on the overall make-span of the plan directly in the cost function of our problem, differentiating therefore whether the planning engine chooses some action, or lets the time go for some δ time: $c(a) = 0$ if $a \in A$ while $c(a) = \delta$ if $a = SIM$.

Example 1 (EXP translation)**.** *Let $\Pi = \langle F, X, I, G, A, \emptyset, P \rangle$ be a* PDDL+ *problem without events encompassing one Boolean variable, i.e., $F = \{f_1\}$, four numeric variables, i.e., $X = \{x_1, x_2, x_3, x_4\}$ and two processes $P = \{\rho_1, \rho_2\}$ such that:*

$$\rho_1 = \langle x_1 > 0, \{\langle x_2, x_3 \rangle\} \rangle,\ \rho_2 = \langle f_1, \{\langle x_2, x_4 \rangle\} \rangle$$

According to R4 of Defs. 5-7, ρ_1 affects x_2 according to $\frac{dx_2}{dt} = x_3$ when $x_1 > 0$ holds and, similarly, ρ_1 affects x_2 according to $\frac{dx_2}{dt} = x_4$ when f_1 holds. Finally, if $x_1 > 0 \wedge f_1$, then $\frac{dx_2}{dt} = x_3 + x_4$. The PDDL2.1 *problem obtained using* EXP *discretised in $t = \delta$ is $\Pi' = \langle F, X, I, G, A \cup \{SIM\}, c \rangle$. The novel action SIM is always applicable and features a conditional effect for each element in $\mathcal{P}^+(P) = \{\{\rho_1\}, \{\rho_2\}, \{\rho_1, \rho_2\}\}$, i.e., SIM $= \langle \top, \mathcal{W}_{\{\rho_1\}}, \mathcal{W}_{\{\rho_2\}}, \mathcal{W}_{\{\rho_1, \rho_2\}} \rangle$, where*[3]*:*

$$\mathcal{W}_{\{\rho_1\}} = (x_1 > 0) \wedge \neg f_1 \triangleright \{\langle inc, x_2, x_3 \cdot \delta \rangle\}$$
$$\mathcal{W}_{\{\rho_2\}} = \neg(x_1 > 0) \wedge f_1 \triangleright \{\langle inc, x_2, x_4 \cdot \delta \rangle\}$$
$$\mathcal{W}_{\{\rho_1, \rho_2\}} = (x_1 > 0) \wedge f_1 \triangleright \{\langle inc, x_2, (x_3 + x_4) \cdot \delta \rangle\}$$

[3]Observe that zero increase can be easily omitted as they do not cause any numeric transition.

Lemma 1. *Let $\Pi = \langle F, X, I, G, A, \emptyset, P \rangle$ be a* PDDL+ *problem, and let $\Pi' = \langle F, X, I, G, A \cup \{SIM\} \rangle$ be the* PDDL2.1 *problem obtained by using the* EXP *translation discretised in $t = \delta$. Π admits a solution under δ discretisation iff so does Π'.*

Proof Sketch. ($\Rightarrow$) Let $\pi_t = \langle \pi, (0, t_e) \rangle$ be a valid solution for Π (assume w.l.o.g. $t_s = 0$) under δ discretisation, and let π' be a PDDL2.1 plan constructed in such a way that: i) for each $\langle a, t \rangle \in \pi$ then $a' \in \pi'$ (where a' is the compiled version of a); ii) for each $\langle a_i, t_i \rangle, \langle a_j, t_j \rangle$ with $a_i \prec a_j$ in π then $a'_i \prec a'_j$ in π' iii) a sequence, possibly empty, of *SIM* actions has to be placed before each action $a'_i \in \pi'$ and at the end of π' according to the following structure:

$$\pi' = \langle \langle SIM \rangle \times \frac{t_0}{\delta}, a'_0, \langle SIM \rangle \times \frac{t_1 - t_0}{\delta}, ..., a'_{n-1}, \langle SIM \rangle \times \frac{t_e - t_{n-1}}{\delta} \rangle$$

where $\langle SIM \rangle \times k$ indicates k repetitions of *SIM*.

Let τ be the sequence of states associated to each significant point of $\mathbb{H}^\pi$ and let τ' be the sequence of states generated by the execution of π'.

In order to prove that π is a valid solution for Π', it suffices to show that, let $\tau = \langle \mathbb{H}^\pi_s(T_0), ..., \mathbb{H}^\pi_s(T_{m-1}) \rangle$ and $\tau' = \langle s_0, ..., s_{m-1} \rangle$, $\mathbb{H}^\pi_s(T_i)$ and s_i are equivalent (agree on all values for $F \cup X$) at each $0 \leq i < m$. We prove this by induction on τ (τ'). The base case ($i = 0$) trivially proves true as $\mathbb{H}^\pi_s(T_0) = I$ and $s_0 = I$. For the induction step, we assume true the statement for some $i < |\tau|$, and prove this for $i + 1$ by considering the two types of transitions occurring between two contiguous significant points in $\mathbb{H}^\pi_s$.

Instantaneous transition. Let $T_i = \langle t_i, n_i \rangle$ and $T_{i+1} = \langle t_{i+1}, n_{i+1} \rangle$ be two significant time points of $\mathbb{H}^\pi$ such that $t_{i+1} = t_i$ and $n_{i+1} = n_i + 1$. R2 of of Defs. 5-7 implies that $\mathbb{H}^\pi_A = \{a_i\} \neq \emptyset$. Since the compiled action a'_i is equal to the original one a_i, it is easy to see that the outcomes of the transitions $\gamma(\mathbb{H}^\pi_s(T_i, a_i))$ and $\gamma(s, a'_i)$ are equivalent, and we know that $s_i \models pre(a'_i)$ as $\mathbb{H}^\pi_s(T_i) \models pre(a_i)$ and s_i and $\mathbb{H}^\pi_s(T_i)$ are equivalent by inductive hypothesis.

Temporal transition. Let i be an index such that $T_i = \langle t_i, n_i \rangle$ and $T_{i+1} = \langle t_i + \delta, 0 \rangle$ are two significant time points of $\mathbb{H}^\pi$. According to R4 of Def. 7 each numeric variable $x \in X$ changes over δ by using the active processes in $\{\rho \in P, \mathbb{H}^\pi_s(T_i) \models pre(\rho)\}$. It is easy to see that the *SIM* operator activates exactly the effects entailed by the conditional effects related to the active processes in $\mathbb{H}^\pi_s(T_i)$. Indeed, $\mathbb{H}^\pi_s(T_i)$ is equivalent to s_i by inductive hypothesis. Thus, $\mathbb{H}^\pi_s(T_{i+1})$ and $s_{i+1} = \gamma(s_i, SIM)$ are equivalent.

($\Leftarrow$) Starting from π' we can build a valid PDDL+ plan $\pi_t = \langle \pi, \langle 0, t_e \rangle \rangle$ as follows: i) for each action $a'_i \in \pi'$ such that $a'_i \neq SIM$ then $\langle a_i, t_i \rangle \in \pi$, where t_i is equal to δ multiplied for the occurrences of *SIM* in π' before a'_i; ii) for each a'_i, a'_j such that $a'_i \prec a'_j$ in π' then $\langle a_i, t_i \rangle \prec \langle a_j, t_j \rangle$ in π and iii) t_e is equal to δ multiplied by the number of *SIM* in π'.

In order to show the validity of π_t, we reason on the discrete projection $\mathbb{H}^\pi$ of π_t that is determined as follows. Every action $\langle a_i, t_i \rangle$ in π is associated with a significant time point T_i such that $\mathbb{H}^\pi_A(T_i) = \{a_i\}$. This implies, by using R1 of Defs. 5-7, the existence of a significant point $T_{i+1} = \langle t_i, n_i + 1 \rangle$ such that $\mathbb{H}^\pi_s(T_{i+1}) = \gamma(\mathbb{H}^\pi_s(T_{i+1}), a_i)$. Any

pair of contiguous actions $\langle a_i, t_i\rangle$ and $\langle a_{i+1}, t_{i+1}\rangle$ induces $k = \frac{t_{i+1}-t_i}{\delta}$ significant time points $T_{i+j} = \langle t_i + j \cdot \delta, 0\rangle$ of $\mathbb{H}^\pi$, with $1 \leq j \leq k$. The number of significant time points equates the number of *SIM* actions between a'_i and a'_{i+1}. Once we have built $\mathbb{H}^\pi$ we can generate τ and, knowing that π' is valid for Π', proceed by induction over τ and τ' in a similar way to the opposite direction.

□

Polynomial Translation

The translation that we present in this section relies on the idea to keep the factored processes-based representation and to reflect this into a number of actions (in the next organized in set A_P), each devoted to model whether one particular process is active, and what its effects on the state of the world are. Instead of evaluating which context holds into an exponentially large set of possible alternatives, we let the planning engine develop the applications of actions from A_P in depth, i.e., by sequencing their execution one after the other. This sequencing gets activated when the planning engine switches into simulation modality, activated by so called action *start*, and ends once all actions from A_P are executed.

Since, as their homologous, these actions can change the values of the variables throughout the execution, *start* also makes a full copy of the numeric state values before simulation. It does so by assigning the current value of all relevant variables into a new set of variables, X^{cp}, before any process starts operating. This lets the planning engine safely evaluate all variables each process effect depends on. In order to achieve this last point, each numeric effect in some process is rewritten before being transformed into an action. The rewriting manipulates each formula in a way to substitute every occurrence of a variable from X with its doppelganger in X^{cp} [4]. Let be ψ a formula, we denote with $\sigma(\psi, X^{cp})$ the result of such a rewriting.

This translation trades the exponential blow-up caused by the context switching operation of EXP, with an increase in the length of the plan. The resulting formulation is sound, complete and, more interestingly, is polynomial on the size of the input. For this reason, we call this translation POLY and detail in what follows its precise definition and functioning.

Let $\Pi = \langle F, X, I, G, A, P, \emptyset\rangle$ be an event-free PDDL+ problem, and a discretisation parameter $t = \delta$, POLY generates a new PDDL2.1 problem $\Pi' = \langle F \cup D \cup \{\mathit{pause}\}, X \cup X^{cp}, I, G \wedge \neg \mathit{pause}, A_c \cup A_P \cup \{\mathit{start}, \mathit{end}\}, c\rangle$ such that:

$$X^{cp} = \{x^{copy} \mid x \in X\}$$

$$D = \bigcup_{\substack{ne \in \mathit{eff}(\rho)\\ \rho \in P}} \{\mathit{done}_{ne}\}$$

$$A_c = \{\langle \mathit{pre}(a) \wedge \neg \mathit{pause}, \mathit{eff}(a)\rangle \mid a \in A\}$$

$$\mathit{start} = \langle \neg \mathit{pause}, \{\mathit{pause}\} \cup \bigcup_{x \in X} \{\langle ass, x^{copy}, x\rangle\}\rangle$$

$$\mathit{end} = \langle \bigwedge_{\mathit{done} \in D} \mathit{done} \wedge \mathit{pause}, \{\neg \mathit{pause}\} \cup \bigcup_{\mathit{done} \in D} \{\neg \mathit{done}\}\rangle$$

$$A_P = \bigcup_{\substack{ne:\langle x, \xi\rangle \in \mathit{eff}(\rho)\\ \rho \in P}} \{\langle \mathit{pause}, \{\sigma(\mathit{pre}(\rho), X^{cp}) \triangleright \langle inc, x, \Delta(\delta, \sigma(\xi, X^{cp}))\rangle\} \cup \{\mathit{done}_{ne}\}\rangle\}$$

As it is possible to observe, at any discrete time step, the planning engine can decide to let time pass by an amount of $t = \delta$, and does so by deciding to execute action *start*– note the empty preconditions. From that moment onward, no action from A_c can be executed, and only when all conditional process effects are applied, the planning engine can come back into actual planning mode ($\neg$*pause*). A_P encompasses all such processes effects, and delegates to a conditional effect the check and the consequent update of the variables according to their past value ($\sigma(\mathit{pre}(\rho), X^{cp})$ for the precondition of the process, and $\sigma(\xi, X^{cp})$ for the right-hand side of the numeric effect) under the proper Δ discretisation. Note that, action *start* also ensures that all the variables are copied through an assignment operation, which is responsible for iterating over all numeric variables of the problem and updating their value for the next round of simulation (the snippet $\bigcup_{x \in X}\{\langle ass, x^{copy}, x\rangle\}$).

Similarly to the exponential translation, also in this case we make the planning engine aware of the passage of time through the cost function. As however we do not have only one action that reflects such a passage of time, we attribute a non-zero cost only to action *start*. That is: $c(a) = \delta$ if $a =$ *start*, 0 otherwise.

Example 2 (POLY translation - Continuing on Ex. 1)**.** *Let $\mathit{ne}_1 = \langle x_2, x_3\rangle$ and $\mathit{ne}_2 = \langle x_2, x_4\rangle$ be the numeric continuous effects of ρ_1 and ρ_2, respectively. The* PDDL2.1 *problem obtained using* POLY *discretised in $t = \delta$ is $\Pi' = \langle F \cup \{\mathit{ne}_1, \mathit{ne}_2\} \cup \{\mathit{pause}\}, X \cup \{x_1^{copy}, x_2^{copy}, x_3^{copy}, x_4^{copy}\}, I, G \wedge \neg \mathit{pause}, A_c \cup \{\mathit{SIM\text{-}ne}_1, \mathit{SIM\text{-}ne}_2\} \cup \{\mathit{start}, \mathit{end}\}, c\rangle$ such that:*

$$\begin{aligned}
\mathit{start} =& \langle \neg \mathit{pause}, \{\langle ass, x_1^{copy}, x_1\rangle, \langle ass, x_2^{copy}, x_2\rangle,\\
& \langle ass, x_3^{copy}, x_3\rangle, \langle ass, x_4^{copy}, x_4\rangle, \mathit{pause}\}\rangle\\
\mathit{SIM\text{-}ne}_1 =& \langle \mathit{pause}, \{(x_1^{copy} > 0) \triangleright \{\langle inc, x_2, x_3^{copy} \cdot \delta\rangle\}, \mathit{done}_{ne_1}\}\rangle\\
\mathit{SIM\text{-}ne}_2 =& \langle \mathit{pause}, \{f_1 \triangleright \{\langle inc, x_2, x_4^{copy} \cdot \delta\rangle\}, \mathit{done}_{ne_2}\}\rangle\\
\mathit{end} =& \langle \mathit{pause} \wedge \mathit{done}_{ne_1} \wedge \mathit{done}_{ne_2}, \{\neg \mathit{pause}, \neg \mathit{done}_{ne_1},\\
& \neg \mathit{done}_{ne_2}\}\rangle
\end{aligned}$$

The novel actions are generated in a way that "start" needs to be followed by any permutation of SIM-ne$_1$ and SIM-ne$_2$, plus action "end". "start" and "end" are used to open and close the simulation sequence, while SIM-ne$_1$ and SIM-ne$_2$ are used to simulate the continuous numeric effects of processes ρ_1 and ρ_2, respectively.

Lemma 2. *Let $\Pi = \langle F, X, I, G, A, \emptyset, P\rangle$ be a* PDDL+ *problem, and let $\Pi' = \langle F', X', I, G \wedge \neg \mathit{pause}, A'\rangle$ be the* PDDL2.1 *problem obtained by using the* POLY *translation discretised in $t = \delta$. Π admits a solution under δ discretisation iff so does Π'.*

[4] Recall that every numeric expression is a formula.

Proof sketch. ($\Rightarrow$) The construction of the plan π' for problem Π' follows the procedure explained in Lemma 1, but for rule iii), where, a sequence, possibly empty, of sequences having the form $wait = \langle start, seq(A_P), end \rangle$ (where $seq(A_P)$ is any sequencing of all A_P operators) has to be placed before each action $a'_i \in \pi'$ and at the end of π' according to the following structure:

$$\pi' = \langle \langle wait \rangle \times \frac{t_0}{\delta}, a'_0, \langle wait \rangle \times \frac{t_1 - t_0}{\delta}, ..., a'_{n-1}, \langle wait \rangle \times \frac{t_e - t_{n-1}}{\delta} \rangle$$

Much as we do for Lemma 1, we proceed by induction over the states τ and τ', noticing that τ' can be constructed using *wait* as if it was a single transition, therefore leading us to have $|\tau| = |\tau'|$. In the following we consider equivalent two states if they share the same values over $X \cup F$.

The aspect that we need to prove is the inductive case when there is a temporal transition (the case base and the instantaneous transitions trivially follow from Lemma 1). In particular, starting from two equivalent states in τ and τ', i.e., $\mathbb{H}^{\pi}_{s}(T_i)$ and s_i, we need to prove that $\mathbb{H}^{\pi}_{s}(T_{i+1})$ is equivalent to $s_{i+1} = \gamma(s_i, wait)$. Note that the action *start* in *wait* has its precondition empty so it is executable. This action copies each $x \in X$ into X^{cp}. All A_P operators become thus executable blocking the execution of any other operator until action *end*. The conditional effects of the A_P actions uniquely depends on X^{cp}; hence, as no action in A_P modifies a variable in X^{cp}, effects in $seq(A_P)$ are commutative. By inductive hypothesis, $\mathbb{H}^{\pi}_{s}(T_i)$ and s_i are equivalent and so is the set of active processes in $\mathbb{H}^{\pi}_{s}(T_i)$ and throughout the execution of A_P. From this point onward, for each $x \in X$, the terms summed by the execution of any possible $seq(A_P)$ are the same as those appearing in the summation provided in R4 of Def. 7. Thus, $\mathbb{H}^{\pi}_{s}(T_{i+1})$ and $s_{i+1} = \gamma(a_i, wait)$ are equivalent. Observe, moreover, that each action a', compiled version of a, requires *pause* to be false. This guarantees that actions are only executable before and after a *wait* block.

($\Leftarrow$) The mapping from π' to $\pi_t = \langle \pi, \langle 0, t_e \rangle \rangle$ is similar to what was done for Lemma 1. All the occurrences of sequence *wait* in π' have to be ignored in building π. The timestamp t_i of each action $\langle a_i, t_i \rangle$ mapped in π is equal to δ multiplied by the number of *start* before a'_i in π'. t_e is equal to δ multiplied by the number of *start* in π'. Then, we proceed by induction on the length of the induced trajectories, as done for Lemma 1. □

Handling Events

An event in PDDL+ models can be triggered at any time during the execution of a plan, and it is necessary to track whether and how such an event changes the state. More importantly, the semantics of PDDL+ prescribes that a cascade of events may also occur.

In order to handle this behaviour, we devise a new action, namely *SIMEV*, that is responsible for keeping track of the arisen events, and their impacts on the state. This action does so by encoding in one single unit the potentially repeated check and execution of several events via a sophisticated usage of conditional effects that are evaluated in rounds. *SIMEV* makes use of 4 sets of conditional effects:

1. $\mathcal{W}_{trig}$ that is responsible for actually updating the state with all events having their precondition satisfied;
2. $\mathcal{W}_{fired}$ that is responsible for keeping track of whether there is some event triggered;
3. $\mathcal{W}_{satu}$ that is responsible for capturing whether no event is triggered in the previous round;
4. $\mathcal{W}_{\perp}$ that captures the situation where there is some inconsistency caused by either a set of mutex events active at the same time or a cyclic sequence of events being triggered.

The formalization of such conditional effects in PDDL2.1 makes use of a number of additional fresh predicates that are accumulated in set F_E. We have a fact *sim-ev* that signals the beginning of the event simulation; then we have a fact $fired_{\varepsilon}$ for each event in $\varepsilon \in E$.

The *SIMEV* action is formalised in such a way that:

$$pre(SIMEV) = sim\text{-}ev$$
$$eff(SIMEV) = \mathcal{W}_{trig} \cup \mathcal{W}_{fired} \cup \mathcal{W}_{satu} \cup \mathcal{W}_{\perp}$$

where:

$$\mathcal{W}_{trig} = \bigcup_{\substack{c \triangleright e \in eff(\varepsilon) \\ \varepsilon \in E}} \{pre(\varepsilon) \wedge c \triangleright e\}$$

$$\mathcal{W}_{fired} = \bigcup_{\varepsilon \in E} \{pre(\varepsilon) \triangleright fired_{\varepsilon}\}$$

$$\mathcal{W}_{satu} = \Big\{ \bigwedge_{\varepsilon \in E} (\neg pre(\varepsilon) \vee fired_{\varepsilon}) \triangleright \{\neg sim\text{-}ev\} \cup \bigcup_{\varepsilon \in E} \{\neg fired_{\varepsilon}\} \Big\}$$

$$\mathcal{W}_{\perp} = \Big\{ \Big(\bigvee_{\substack{\varepsilon, \varepsilon' \in E: \\ \varepsilon \neq \varepsilon' \wedge \\ mutex(\varepsilon, \varepsilon')}} pre(\varepsilon) \wedge pre(\varepsilon') \Big) \vee \vee \bigvee_{\varepsilon \in E} pre(\varepsilon) \wedge fired_{\varepsilon} \triangleright \{\perp\} \Big\}$$

As it is possible to observe from the snippet above, *SIMEV* captures which events have been triggered and, on the one hand, applies their effects, and on the other hand, memorizes whether at least one event has been executed. If that is the case, the action needs to re-evaluate the conditional effects; indeed, events can be triggered in cascade. It is easy to see that the triggering of events is blocked whenever the action detects a cycle, i.e., an event that is deemed to be executed more than once. For this reason, *SIMEV* can be performed up to $|E|$ times, that is, after the termination condition induced by $\mathcal{W}_{satu}$ is reached. Observe that this effect ($\{\neg sim\text{-}ev\} \cup \bigcup_{\varepsilon \in E} \{\neg fired_{\varepsilon}\}$) not only interrupts the execution of the simulation of the events, but also resets all *fired* facts; this way, we keep the memory ready for the next round of simulation of events.

The very last set of conditional effects ensures that the reached state does not contain cycles or mutexes events that can be executed at the same time. If either of these two situations arise, *SIMEV* generates an inconsistent state, thereby denoted by the special $\perp$ symbol. This check guarantees to prune states where interfering events leading to nondeterministic outcomes or infinite cascade of events arise.

This complies with the semantics restrictions imposed by Shin and Davis (2005); Fox and Long (2006); Fox, Howey, and Long (2005).

The presented exponential and polynomial translations can be extended to address PDDL+ planning problems with events by forcing action *SIMEV* to be applied in the initial state, and switching back to event simulation (*sim-ev*) modality after any occurrence of an instantaneous action (in both translations), after action *SIM* in EXP and after action *end* in POLY. Let a be an action, we denote with *ev-check*(a) the operation that generates a new action a' such that: *pre*$(a') =$ *pre*$(a) \wedge \neg$*sim-ev* and *eff*$(a') =$ *eff*$(a) \cup \{$*sim-ev*$\}$.

We are now ready to summarize the further translation that is needed in order to make the resulting PDDL2.1 formulation aware of the presence of events, for both cases.

From EXP. Let $\Pi = \langle F, X, I, G, A, P, E\rangle$ be a PDDL+ problem, and let $\Pi_{\text{EXP}} = \langle F, X, I, G, A \cup \{SIM\}, c\rangle$ the PDDL2.1 problem obtained using EXP ignoring set E, respectively. The handling of events E can be achieved by a further translation into $\Pi^{events}_{\text{EXP}} = \langle F \cup F_E, X, I \cup \{$*sim-ev*$\}, G \wedge \neg$*sim-ev*$, A' \cup \{$*ev-check*$(SIM)\} \cup \{SIMEV\}, c\rangle$ with $A' = \bigcup_{a\in A}$ *ev-check*(a).

From POLY. Let $\Pi = \langle F, X, I, G, A, P, E\rangle$ be a PDDL+ problem, and let $\Pi_{\text{POLY}} = \langle F \cup D \cup \{$*pause*$\}, X \cup X^{cp}, I, G \wedge \neg$*pause*$, A_c \cup A_P \cup \{$*start*, *end*$\}, c\rangle$ the PDDL2.1 problem obtained using POLY ignoring set E, respectively. The handling of events E can be achieved by a further translation into $\Pi^{events}_{\text{POLY}} = \langle F \cup D \cup \{$*pause*$\} \cup F_E, X \cup X^{cp}, I \cup \{$*sim-ev*$\}, G \wedge \neg$*pause* $\wedge \neg$*sim-ev*$, A'_c \cup A_P \cup \{$*start*, *ev-check*$($*end*$)\} \cup \{SIMEV\}, c\rangle$ with $A'_c = \bigcup_{a\in A_c}$ *ev-check*(a).

Properties

Theorem 1. *Let $\Pi = \langle F, X, I, G, A, E, P\rangle$ be a* PDDL+ *problem, and let $\Pi^{events}_{\text{EXP}}$ ($\Pi^{events}_{\text{POLY}}$) be the* PDDL2.1 *problem obtained by using the* EXP (POLY) *translation. Π admits a solution under δ discretisation iff so does $\Pi^{events}_{\text{EXP}}$ ($\Pi^{events}_{\text{POLY}}$).*

Proof sketch. We focus on $\Pi^{events}_{\text{EXP}}$ (the proof for $\Pi^{events}_{\text{POLY}}$ is similar) and sketch a proof for the two directions.

($\Rightarrow$) Let $ES(T_i) = \langle E_{trigg}(T_i), ..., E_{trigg}(T_{i+k-1})\rangle$ be the unique and finite[5] cascade of events from a significant time point T_i of the discrete projection $\mathbb{H}^\pi$ of π_t solving Π.

We construct a plan π' from π_t such that π' is valid for Π_{EXP}. Differently from an event-free planning task where we can define a mapping from π_t to π', in this case we have to resort to $\mathbb{H}^\pi$ to build a valid π'. Let $T_{\mathbb{H}} = \langle T_0, ..., T_m\rangle$ be the $m+1$ significant time points of $\mathbb{H}^\pi$, we define π' in two steps. As a first step we define the sequence $\pi'' = \langle a'_0, \ldots, a'_{m-1}\rangle$ such that for all $i \in \{0, \ldots, m-1\}$

$$a'_i = \begin{cases} a_i & \text{if } \mathbb{H}^\pi_A(T_i) = \{a_i\} \neq \emptyset \\ SIMEV & \text{if } E_{trigg}(T_i) \neq \emptyset \\ SIM & \text{otherwise} \end{cases}$$

Then we obtain π' from π'' by inserting a *SIMEV* action just before any action $a \in \pi''$ such that $a \neq$ *SIMEV*, and just before the end of the plan. Let $\tau' = \langle s_0, \ldots, s_m\rangle$ be the sequence of states obtained by applying iteratively actions from π' and filtering out those states produced by any last *SIMEV* of a series, the difficult bit is to show that τ' is equivalent to $\tau = \langle \mathbb{H}^\pi_s(T_0), \ldots, \mathbb{H}^\pi_s(T_m)\rangle$ under variables $F \cup X$. The first observation is that $|\tau| = |\tau'|$; and this follows directly from the fact that the number of states we are filtering out is exactly the number of *SIMEV* that we have added to the plan. Then, in order to prove that $\mathbb{H}^\pi_s(T_i)$ and s_i are equivalent for all $0 \leq i \leq m$, we can use the same arguments of Lemma 1 extended to account for the case where the transition is due to a cascade of events. More precisely, let $T_i = \langle t_i, n_i\rangle$ be a significant time point in which a cascade of events $ES(T_i)$ with $|ES(T_i)| = k$ is triggered. By using R1 of Defs. 5-7 we know that, for each $i \leq j < i+k$, $\mathbb{H}^\pi_s(T_j) \models \bigwedge_{\varepsilon\in E_{trigg}(T_j)}$ *pre*(ε), $\mathbb{H}^\pi_s(T_{j+1}) = \gamma(\mathbb{H}^\pi_s(T_j), E_{trigg}(T_j))$ and $T_{j+1} = \langle t_i, n_j + 1\rangle$.

In order to prove $\mathbb{H}^\pi_s(T_{i+k}) = s_{i+k}$ with $s_{i+k} = \gamma(s_i,$ *SIMEV* $\times\, k)$ we need to show that, for every j such that $i \leq j \leq i+k$, $\mathbb{H}^\pi_s(T_j) = s_j$; we do so, again, by induction. The base case ($j = i$) is trivially proved (inherited by Lemma 2). For the inductive step it suffices to observe that *SIMEV* exhibits a behavior that is equivalent to $\langle \top, \bigcup_{\substack{c\triangleright e\in \textit{eff}(\varepsilon) \\ \varepsilon\in E_{trigg}(T_j)}} c \triangleright e\rangle$ and then it follows that $\mathbb{H}^\pi_s(T_{j+1})$ and $s_{j+1} = \gamma(s_j,$ *SIMEV*$)$ are equivalent. The only thing that is missing is to show that when each action is applied in π the variable *sim-ev* is false. But this directly follows from the fact that the last *SIMEV* is applied when all events have been triggered. Indeed, we have that $E_{trigg}(T_{i+k}) = \emptyset$, and $\mathbb{H}^\pi_s(T_{i+k}) = s_{i+k}$. So, *SIMEV* will make *sim-ev* false and get ready for the next round of execution by resetting all the monitoring variables *fired* to false.

($\Leftarrow$) The mapping from π' to π_t is identical to what was done for Lemma 1, but for the fact that all occurrences of *SIMEV* are ignored. Then, plan π_t can be proved valid against the discretised PDDL+ model by observing that, for each series of *SIMEV* of length k the projection $\mathbb{H}^\pi$ encompasses $k-1$ significant points, one for each *SIMEV* that triggers a change on variables in $F \cup X$. Each significant point generates a set of events whose effects are those that arise from the active conditional effects of the associated *SIMEV*. This is due to the fact that the *SIMEV* conditional effect's condition subsumes the precondition of each event is associated with.

□

Theorem 2. *Let $\Pi = \langle F, X, I, G, A, E, P\rangle$ be a* PDDL+ *problem, and let $\Pi^{events}_{\text{POLY}}$ and $\Pi^{events}_{\text{EXP}}$ be the* PDDL2.1 *problems obtained by using the* POLY *and* EXP *translations, respectively. Translations* POLY *and* EXP *preserve plan size polynomially in the sense of (Nebel 2000).*

Proof Sketch. Let $\pi_t = \langle \pi, \langle t_s, t_e\rangle\rangle$ be a solution for Π under δ discretisation and let π_{POLY} and π_{EXP} be the corresponding plan for $\Pi^{events}_{\text{POLY}}$ and $\Pi^{events}_{\text{EXP}}$, respectively.

Using the rules outlined in Lemma 2 and Theorem 1 to map π_t into π_{POLY} we can prove the upper-bound on $|\pi_{\text{POLY}}|$ as follows: i) each action of π' has to be followed by at least

[5]Note that, under the restriction imposed over PDDL+, for each $E, E' \in ES$, $E \cap E' = \emptyset$ and, for each $\varepsilon, \varepsilon' \in E$ with $E \in ES$, ε and ε' are not mutex, as long as $\varepsilon \neq \varepsilon'$.

one *SIMEV* action up to a maximum of $|E|+1$ and such sequence has to be executed also at the beginning of π' since $I' \models$ *sim-ev*; ii) possible event triggers must be checked after the block of actions that simulates the passage of time, i.e., the *wait* sequence; therefore $|E|+1$ has to be multiplied by the number of time steps occurred within the envelope $\langle t_s, t_e \rangle$, i.e., $\frac{t_e - t_s}{\delta}$; iii) each *wait* in π' consists of two delimiting actions, i.e., *start* and *end*, plus an action for each numeric effect of each process; in the worst case, each process has a numeric effect for each variable of the problem. By adding these contributions we obtain:

$$|\pi_{\text{POLY}}| \leq (|\pi|+1) \cdot \overbrace{(|E|+1)}^{i)} + \frac{t_e - t_s}{\delta} \cdot (\overbrace{(|E|+1)}^{ii)} + \overbrace{(|P| \cdot |X|+2)}^{iii)})$$

The only difference for π_{EXP} is that instead of the *wait* sequence we apply a single action *SIM*. □

Experimental Analysis

Our experimental analysis aims at assessing the extent to which the introduced translations allow to reformulate PDDL+ instances into instances amenable for PDDL2.1 planning engines (the translator and the benchmark suite can be found at https://bit.ly/30gMyNW). In this evaluation we consider three engines at the state of the art for PDDL+ planning: ENHSP version 20 (Scala et al. 2020) with the AIBR heuristic (Scala et al. 2016), SMTPLAN (Cashmore, Magazzeni, and Zehtabi 2020), and DINO (Piotrowski et al. 2016). As a PDDL2.1 planning engine we use the well-known METRIC-FF (Hoffmann 2003). We could not consider other numeric planning systems such as LPG (Gerevini, Saetti, and Serina 2008), OPTIC (Benton, Coles, and Coles 2012) or the same ENHSP because none of them provide an effective support for conditional effects and negated preconditions. All the planning engines have been run using default parameters.

Our experiments were run on an Intel Xeon Gold 6140M CPUs with 2.30 GHz. For each instance we set a cutoff time of 900 seconds, and memory was limited to 8 GB.

For our experimental evaluation, we consider six benchmark domains. Three of them, in particular Linear-Car (Lin-Car), Linear-Generator (Lin-Gen), and Solar-Rover (Rover), are well-known PDDL+ benchmarks. Overtaking-Car (OT-Car) is a version of Linear-Car that extends the original domain by considering multiple lanes, and the need for the car to move between lanes in order to avoid obstacles. Baxter (Bertolucci et al. 2019) and Urban Traffic Control (UTC) (Vallati et al. 2016) are taken from real-world applications. The Baxter domain exploits planning for supporting robots in dealing with articulated objects manipulation tasks. The UTC domain models the use of planning for generating traffic light signal plans, in order to de-congest an area of an urban region.

Table 1 gives an intuition of the size increase due to the use of the translation. For each domain, we reported the average number of ground processes and events, i.e., $\mu(|P|)$ and $\mu(|E|)$ respectively, and a measure of the size increase introduced by the used translation w.r.t. the original grounded problem. Let $\Pi = \langle F, X, I, G, A, P, E \rangle$ be a PDDL+ problem and let $\Pi' = \langle F', X', I', G', A', c \rangle$ the corresponding PDDL2.1 problem obtained by using $\mathcal{T} \in \{\text{POLY}, \text{EXP}\}$, we define the size increase ratio, denoted with r, introduced by $\mathcal{T}$ as $r(\mathcal{T}) = \frac{|A|+|P|+|E|}{|A'|+|\mathcal{W}|}$, where $\mathcal{W}$ denotes a subset of conditional effects of Π' such that $\mathcal{W} = \bigcup_{a \in A'} \{c \triangleright e | c \triangleright e \in \textit{eff}(a),\ c \neq \top\}$.

Domain	$\mu(\|P\|)$	$\mu(\|E\|)$	$r(\text{POLY})$	$r(\text{EXP})$
Rover (20)	4.0	5.0	1.43	1.36
Lin-Car (10)	2.0	0.0	1.67	1.33
Lin-Gen (10)	6.1	8.3	2.04	16.03
UTC (10)	34.1	15.8	2.47	3834.02
Baxter (20)	56.0	22.0	1.52	—
OT-Car (20)	4.1	5.4	1.31	2.1

Table 1: For each domain, $\mu(|P|)$ and $\mu(|E|)$ denote the average number of grounded processes and events, respectively, while $r(\text{POLY})$ and $r(\text{EXP})$ denote the average size increase ratio of the instances. "—" indicates a translation failure due to the size. Between brackets, the number of problem instances considered for each domain.

Domain	METRIC-FF		DINO	ENHSP	SMTPLAN
	POLY	EXP			
Rover (20)	**20**	**20**	**20** *	5	19
Lin-Car (10)	**10**	**10**	**10** *	**10**	**10** *
Lin-Gen (10)	**10**	3	**10** *	**10**	**10** *
UTC (10)	**7**	0	0	**7**	0
Baxter (20)	**19**	0	7	17	8
OT-Car (20)	18	**19**	0	**19**	0
TOTAL	**84**	52	47	68	47

Table 2: Number of problems solved by the considered planning approaches. Between brackets, the number of problem instances considered for each domain. POLY and EXP are used to indicate that, respectively, the polynomial or the exponential translation has been used. "*" denotes that the reported result refers to a variant of the domain model we considered, modified to allow the specific engine to reason upon it. Bold indicates best results.

Table 2 shows the achieved results, in terms of number of solved problems, by the considered planning approaches on the benchmark domains. POLY and EXP are used to indicate that, respectively, the polynomial or the exponential translation has been used to allow METRIC-FF to deal with the considered problem instances. It is also worth remarking that some of the PDDL+ planning engines required the models to be modified in order to generate a solution: this has been indicated in the table using an "*". The presented results highlight that the proposed translation is effective in supporting the use of PDDL2.1 planning engines for solving complex hybrid planning problems. The polynomial translation seems to be more indicated in domains where there is a large number of ground events and processes, such as UTC and Baxter. In the other domains, it seems that the exponential translation can instead better support the planning process of PDDL2.1 engines: a notable example is the OT-Car domain.

Figure 1 gives some insights into the CPU-time needed

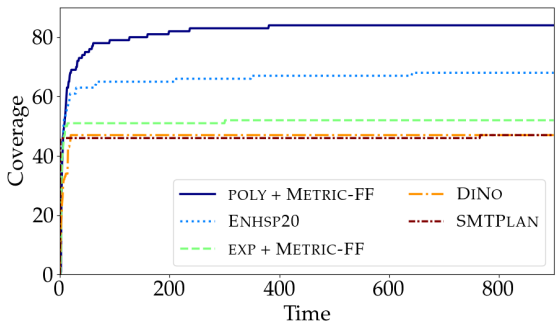

Figure 1: Total number of instances solved by each of the considered planning approaches, over time.

by the considered systems to solve the benchmark problems. All the approaches are able to quickly solve a large number of considered instances. When using the polynomial translation, METRIC-FF is able to solve approximately 80 instances in less than 100 CPU-time seconds.

With regards to the quality of the generated plans, measured as makespan, we did not observe any significant overall difference between plans generated using the PDDL+ or the PDDL2.1 models. Quality seems to be more affected by the planning approach exploited by the engine, rather than by the use of a specific formulation.

Summarising, the results of this experimental analysis indicate that the translations can effectively support the use of PDDL2.1 planning engines for solving PDDL+ instances.

Related Work

A range of techniques have been introduced to support the reasoning on PDDL+ instances by means of translation. Balduccini et al. (2017) proposed an approach for translating a PDDL+ instance in a Constraint ASP instance (Baselice, Bonatti, and Gelfond 2005), but the process has to be done manually by an expert of the field. A number of approaches have been introduced to translate PDDL+ instances into satisfiability modulo theories (SMT) (Barrett and Tinelli 2018) problems (e.g., (Bryce et al. 2015; Cashmore, Magazzeni, and Zehtabi 2020)) or a mix of linear programming and SAT instances (Shin and Davis 2005). These translations differ among each other in the way the compilation is carried on; some of them makes use of an intermediate translation step into hybrid automata (Bryce et al. 2015; Heinz et al. 2019; Cashmore, Magazzeni, and Zehtabi 2020), other performs a more direct translation (Shin and Davis 2005). We follow on these lines, yet, as we map the problem into another planning problem, we are not required to provide an upper limit on the plan length as these approaches do. Those translations indeed all require the ability to anticipate a maximum number of time points, sharing the same issues of incompleteness of SAT-based planning (Kautz and Selman 1992).

Coles and Coles (2014) introduced a translation between a non-discretised PDDL+ problem instance and a PDDL2.1 temporal continuous instance, but showed that such way does not lead to models that are suitable for PDDL2.1 planning engines. Our approach targets the level 2 of the PDDL2.1 language. The language that we are targeting does not have a notion of time, with the result that constructing a planning engine supporting it is much easier than one that needs to natively support temporal reasoning.

A different line of work in automated planning focuses on reformulating models without involving a translation to a different language. With regards to PDDL+ models, Franco et al. (2019) introduced a technique for minimising the ground size of PDDL+ problems by reducing the arity of sparse predicates, i.e., predicates with a very large number of possible groundings, out of which very few are actually exploited in the planning problems.

There is an interesting parallel with compilations devised for classical planning models (Nebel 2000; Gazen and Knoblock 1997). In particular, our exponential translation anticipates the possible contexts a system is in much as the exponential encoding by Gazen and Knoblock (1997) compile away conditional effects, while our polynomial translation captures the semantics of processes unrolling them into a number of actions, much as Nebel (2000) proposes to simulate the execution of conditional effects. As these two approaches have contributed the discovery of a number of techniques and heuristics for classical planning (e.g., (Haslum 2013; Röger, Pommerening, and Helmert 2014)), we believe that our schemata can do the same for the much more involved case of PDDL+.

The idea of tackling problems involving continuous change of variables through simpler forms of discrete planning has been investigated by other works, too (e.g., Löhr et al. (2012) and (Say and Sanner 2019)). Yet, to the best of our knowledge, no previous work has done so using a translation-based approach that starts from a declarative representation of the problem, in our case PDDL+. Understanding synergies among these lines of research is indeed a very important avenue for future work.

Summary and Future Work

To deepen the understanding of PDDL+, and to support the solvability of PDDL+ instances, we introduced two translations from time-discretised PDDL+ to PDDL2.1 (level 2). The exponential translation leads to a numeric planning problem which is exponentially larger than the initial PDDL+, but preserves the number of discrete transitions. The polynomial translation instead leads to a smaller formulation but requires more transitions to generate a solution. Our experimental analysis demonstrated the usefulness of the introduced translations in unlocking the exploitation of PDDL2.1 planning engines to solve challenging PDDL+ instances.

Future work will focus on exploring incomplete translations, where a trade-off can potentially be found between completeness and the size of the resulting PDDL2.1 instances.

Acknowledgements

Mauro Vallati was supported by a UKRI Future Leaders Fellowship [grant number MR/T041196/1].

References

Balduccini, M.; Magazzeni, D.; Maratea, M.; and Leblanc, E. C. 2017. CASP solutions for planning in hybrid domains. *Theory and Practice of Logic Programming* 17: 591633.

Barrett, C.; and Tinelli, C. 2018. Satisfiability modulo theories. In *Handbook of Model Checking*, 305–343.

Baselice, S.; Bonatti, P. A.; and Gelfond, M. 2005. Towards an integration of answer set and constraint solving. In *Proceedings of ICLP*, 52–66.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of ICAPS*. AAAI.

Bertolucci, R.; Capitanelli, A.; Maratea, M.; Mastrogiovanni, F.; and Vallati, M. 2019. Automated Planning Encodings for the Manipulation of Articulated Objects in 3D with Gravity. In *Proceedings of AI*IA*, 135–150.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P. 2015. SMT-Based Nonlinear PDDL+ Planning. In *Proceedings of AAAI*, 3247–3253.

Cashmore, M.; Magazzeni, D.; and Zehtabi, P. 2020. Planning for Hybrid Systems via Satisfiability Modulo Theories. *J. Artif. Intell. Res.* 67: 235–283.

Ceriani, L.; and Gerevini, A. E. 2015. Planning with always preferences by compilation into strips with action costs. In *Proceedings of SoCS*, 161–165.

Coles, A. J.; and Coles, A. I. 2014. PDDL+ Planning with Events and Linear Processes. In *Proceedings of ICAPS*, 74–82.

Cooper, M. C.; Maris, F.; and Rgnier, P. 2010. Compilation of a High-level Temporal Planning Language into PDDL 2.1. In *Proceedings of ICTAI*, 181–188.

Fox, M.; Howey, R.; and Long, D. 2005. Validating Plans in the Context of Processes and Exogenous Events. In *Proceedings of AAAI*, 1151–1156.

Fox, M.; and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.* 20: 61–124.

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

Franco, S.; Vallati, M.; Lindsay, A.; and McCluskey, T. L. 2019. Improving Planning Performance in PDDL+ Domains via Automated Predicate Reformulation. In *Proceedings of ICCS*, 491–498.

Gazen, B. C.; and Knoblock, C. A. 1997. Combining the Expressivity of UCPOP with the Efficiency of Graphplan. In *Proceedings of ECP*, 221–233.

Gerevini, A.; Saetti, A.; and Serina, I. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artif. Intell.* 172(8-9): 899–944.

Haslum, P. 2013. Optimal Delete-Relaxed (and Semi-Relaxed) Planning with Conditional Effects. In *Proceedings of IJCAI*, 2291–2297.

Heinz, A.; Wehrle, M.; Bogomolov, S.; Magazzeni, D.; Greitschus, M.; and Podelski, A. 2019. Temporal Planning as Refinement-Based Model Checking. In *Proceedings of ICAPS*, 195–199.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Artif. Intell. Res.* 20: 291–341.

Kautz, H. A.; and Selman, B. 1992. Planning as Satisfiability. In *Proceedings of ECAI*, volume 92, 359–363.

Löhr, J.; Eyerich, P.; Keller, T.; and Nebel, B. 2012. A Planning Based Framework for Controlling Hybrid Systems. In *Proceedings of ICAPS*, 164–171.

Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *J. Artif. Intell. Res.* 12: 271–315.

Palacios, H.; and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *J. Artif. Intell. Res.* 35: 623–675.

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

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic Planning for Hybrid Systems. In *Proceedings of AAAI*, 4254–4255.

Röger, G.; Pommerening, F.; and Helmert, M. 2014. Optimal Planning in the Presence of Conditional Effects: Extending LM-Cut with Context Splitting. In *Proceedings of ECAI*, 765–770.

Say, B.; and Sanner, S. 2019. Metric Hybrid Factored Planning in Nonlinear Domains with Constraint Generation. In *Proceedings of CPAIOR*, 502–518.

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

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

Shin, J.; and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence* 166: 194–253.

Taig, R.; and Brafman, R. I. 2013. Compiling conformant probabilistic planning problems into classical planning. In *Proceedings of ICAPS*, 197–205.

Vallati, M.; Magazzeni, D.; Schutter, B. D.; Chrpa, L.; and McCluskey, T. L. 2016. Efficient Macroscopic Urban Traffic Models for Reducing Congestion: A PDDL+ Planning Approach. In *Proceedings of AAAI*, 3188–3194.