POLITECNICO DI TORINO
Repository ISTITUZIONALE

Initial Condition Retrieving for Hybrid and Numeric Planning Problems

Original
Initial Condition Retrieving for Hybrid and Numeric Planning Problems / Cardellini, M., Percassi, F., Maratea, M., Vallati, M.. - 35:(2025), pp. 21-29. (The 35th International Conference on Automated Planning and Scheduling Melbourne, Victoria (AUS) November 9-14, 2025) [10.1609/icaps.v35i1.36097].

Availability:
This version is available at: 11583/3002390 since: 2025-08-12T09:24:15Z

Publisher:
AAAI

Published
DOI:10.1609/icaps.v35i1.36097

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 November 2026

Initial Condition Retrieving for Hybrid and Numeric Planning Problems

Matteo Cardellini[1], Francesco Percassi[2], Marco Maratea[3], Mauro Vallati[2]

[1]DIBRIS, University of Genova, Genova, Italy
[2]School of Computing and Engineering, University of Huddersfield, Huddersfield, United Kingdom
[3]DeMaCS, University of Calabria, Rende, Italy
matteo.cardellini@edu.unige.it, f.percassi@hud.ac.uk, marco.maratea@unical.it, m.vallati@hud.ac.uk

Abstract

Real-world applications of planning techniques often deal with dynamic and noisy environments, where sensor readings are often inaccurate, and the world's states can evolve in unexpected ways. This is particularly challenging for hybrid discrete-continuous planning approaches, where processes and events can be strongly affected by even slightly different initial conditions of the world, and planning tasks are notoriously difficult to cope with. In this paper, we introduce the Initial Condition Retrieving (ICR) problem to foster hybrid planning in real-world applications. Given a knowledge model of a planning task and a trace, solving the ICR problem allows identifying the space of all the initial conditions from which the provided plan is guaranteed to reach a goal state. We define three tasks: (i) retrieving any valid initial condition, (ii) fixing only some desired initial values and retrieving a complete initial condition that fills in the unassigned values, or (iii) retrieving the closest achievable initial condition to a fully specified one from which the goal cannot be reached. Experiments on well-known hybrid planning domains demonstrate the efficacy of our approach in solving such tasks. Moreover, given that our approach can be applied to numeric planning without any change, we extend our analysis to numeric domains, where we obtain positive results.

Introduction

The nature of real-world applications usually requires the ability to reason in hybrid discrete/continuous changes of numeric variables. In automated planning, this necessity led to the design of hybrid planning, expressible with PDDL+ (Fox and Long 2006), that introduces the notions of processes and events to represent continuous and instantaneous changes on numeric variables, and has already proved very effective in solving complex real-world problems such as Traffic Control (Vallati et al. 2016), Train Dispatching (Cardellini et al. 2021), Unmanned Aerial Vehicle Control (Kiam et al. 2020) and Pharmacokinetic Optimization (Alaboud and Coles 2019). Hybrid planning tasks are notoriously difficult to cope with, due to the intrinsic difficulties of reasoning with numeric variables (Helmert 2002) and time in an intertwined way. The challenges of hybrid planning are also exacerbated when it comes to the validation of generated plans and verification of initial state conditions (Percassi, Scala, and Vallati 2023b; Scala, McCluskey, and Vallati 2022). A well-established approach to reasoning about hybrid planning tasks is through discretisation (Penna, Magazzeni, and Mercorio 2012; Percassi, Scala, and Vallati 2023a; Cardellini et al. 2024), which allows breaking down complexity by assuming the time is discrete, and so are the actual numeric changes. This enables the reuse of well-established general search techniques based on forward state-based exploration to address hybrid planning tasks; it is indeed widely exploited by existing domain-independent planning engines such as DiNo (Piotrowski et al. 2016), UPMURPHI (Penna, Magazzeni, and Mercorio 2012) and ENHSP (Scala et al. 2016).

In real-world applications, we could often find ourselves in the following situation: we observe a current state s_G that satisfies a goal condition G, and we can inspect a trace of logs $\mathcal{T}$, i.e., the complete description of how the agent acted, and the system evolved until s_G. We ask ourselves what initial conditions could have originated $\mathcal{T}$ and from which we could reach s_G or any other goal state. This approach is helpful in a range of domains, for instance in a cybersecurity framework (Hoffmann 2015; Parkinson, Khan, and Chrpa 2020) where we are keen to discover the possible initial states from which an attack $\mathcal{T}$ reaches an undesired set of states G, to patch these initial states and protect the system from the attack. Here, we dub this problem the Initial Condition Retrieving problem. While significant work has been dedicated to plan fixing (Percassi, Scala, and Vallati 2023b), state repairing (Bezrucav et al. 2022), domain fixing (Lin, Grastien, and Bercher 2023), and goal recognition (Pereira, Oren, and Meneguzzi 2020; Chiari et al. 2023), there is a lack of work in addressing the repair of the initial condition (Göbelbecker et al. 2010; Herzig et al. 2014), and only propositional planning has been considered so far. In the context of hybrid automata (Henzinger et al. 1998; Henzinger and Kopke 1999) (from which hybrid planning takes inspiration, see Bogomolov et al. 2015), the concept of *weakest precondition set* (Boreale 2020) aims at finding the whole possible initial space given any hybrid automata (i.e., for any possible hybrid planning task), which is undecidable even in the simplest setting. This differs from our focus, which is on identifying the initial space of a hybrid planning task given a finite and valid trace.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In this paper, we formally define the Initial Condition Retrieving (ICR) problem. To the best of our knowledge, this is the first work on the retrieval and repair of initial conditions in hybrid planning. Solving the ICR problem requires identifying an Initial Condition Space (ICS) from which, by emulating a trace from any initial condition inside it, we are guaranteed to reach a goal state. This paper makes several key contributions. First, we formally define how to construct the ICS, focusing on discretised hybrid planning. Second, we introduce an efficient approach for addressing the ICR problem, translating it into a mathematical programming problem. Third, we empirically demonstrate its usefulness in a range of benchmarks to solve different scenarios like (i) retrieving any initial condition inside the ICS, (ii) fixing only some desired initial values and retrieving the initial condition in the ICS which fills the unassigned values, and (iii) specifying a full initial condition from which we are unable to reach the goal and retrieve its closest "repaired" counterpart in the ICS. Fourth, we show that the proposed approach can be seamlessly applied to numeric planning tasks. In the experimental analysis, we will benchmark these three scenarios on well-known domains in the literature of hybrid planning and on problems of the latest IPC track on numeric planning (Taitler et al. 2024), obtaining positive results on a wide range of planning domains.

Background

Discrete Hybrid Planning. Let $\delta \in \mathbb{Q}_{\geq 0}$ be a *discretisation step*. A *Discrete Hybrid (*DH*) planning task*, expressible with PDDL+, is a tuple $\Pi_\delta = \langle V_\mathbb{B}, V_\mathbb{R}, I, G, A, E, P_\delta \rangle$. $V_\mathbb{B}$ and $V_\mathbb{R}$ are finite sets of propositional and numeric variables with domains in $\mathbb{B} = \{\top, \bot\}$ and $\mathbb{R}$, respectively. A *state* s is a total assignment to the variables in $V_\mathbb{B}$ and $V_\mathbb{R}$ to their respective domains. The *initial condition* I is a state. A *propositional condition* is an expression of the form $v = \top$ or $v = \bot$ with $v \in V_\mathbb{B}$. A *numeric expression* φ is a formula over the variables in $V_\mathbb{R}$ and coefficients in $\mathbb{Q}$, in which variables can appear summed, subtracted, multiplied and divided. A *numeric condition* is an expression of the form $\varphi \trianglerighteq 0$, with φ being a numeric expression and $\trianglerighteq \in \{=, >, \geq\}$. The *goal* G is a set of propositional and numeric conditions. The sets of *actions* A, *events* E and *processes* P_δ are sets of happenings. A *happening* h is a tuple of the form $\langle \mathrm{pre}(h), \mathrm{eff}(h) \rangle$ where the precondition $\mathrm{pre}(h)$ is a set of propositional and numeric conditions and the effect $\mathrm{eff}(h)$ is a set of propositional and numeric assignments. A *propositional assignment* is an expression of either $v := \top$ or $v := \bot$ with $v \in V_\mathbb{B}$. A *numeric assignment* is an expression of the form $x := \varphi$ with $x \in V_\mathbb{R}$ and φ a numeric expression. Even if, syntactically, the actions A, the events E and the processes P_δ are all sets of happenings, semantically they are quite different. Actions prescribe *may transitions*, meaning that, even if they are applicable, the agent could decide to apply them. Events and processes instead prescribe *must transitions*, meaning that, if their preconditions are satisfied, they must immediately affect the state. Events model *one-time* change in the propositional and numeric variables, instead, processes model a flow of change in the numeric variables. For this reason, we will assume in the following that processes have, in their effects, only numeric effects in the form $x := x + \varphi_1 \cdot \delta$, with φ_1 is a numeric expression denoting the discrete change of x.

Let s be a state, v be a variable in $V_\mathbb{B} \cup V_\mathbb{R}$ and φ a numeric expression, we denote with $s(v)$ the value assumed by the variable v in the state s, and with $s(\varphi)$ the value obtained by substituting in φ all the variables $x \in V_\mathbb{R}$ with $s(x)$. A set of propositional or numerical conditions Ψ is satisfied in a state s, written as $s \models \Psi$, if for each $v = \top$ or $w = \bot$ in Ψ, we have $s(v) = \top$ and $s(w) = \bot$ and for each $\varphi \trianglerighteq 0$ in Ψ, we have $s(\varphi) \trianglerighteq 0$. A happening h is applicable in a state s if $s \models \mathrm{pre}(h)$. Applying a happening h to the state s results in the state $s' = \gamma(h, s)$, in which (i) if $v := e \in \mathrm{eff}(h)$, with $e \in V_\mathbb{B}$, $s'(v) = e$; (ii) if $x := \varphi \in \mathrm{eff}(h)$, with φ being a numeric expression, $s'(v) = s(\varphi)$; and (iii) $s'(v) = s(v)$ otherwise. Applying a sequence of happenings $\langle h_1; \ldots; h_k \rangle$ to a state s results in the state $\gamma(\langle h_1; \ldots; h_k \rangle, s) = \gamma(h_k, \gamma(h_{k-1}, \gamma(\ldots, \gamma(h_1, s))))$.

We indicate with $E(s) \subseteq E$ the set of events which are triggered at a state s, i.e., $E(s) = \{e \in E \mid s \models \mathrm{pre}(e)\}$. In this paper, we follow the semantics of events specified in (Fox, Howey, and Long 2005) for guaranteeing the determinism of events. In particular, we impose that (i) each pair of events in $E(s)$ are not in mutual exclusion with each other (i.e., applying them in any order doesn't change the outcome), and (ii) after applying all the events in $E(s)$ it is no longer possible to apply any event in $E(s)$ again, thus avoiding infinite repetition of events. We denote with $\gamma(E(s), s)$ the result of applying all the events in $E(s)$ in an arbitrary sequence. Similarly to $E(s)$, we define $P_\delta(s)$ as the set of processes applicable in the state s. For this reason, given a state s, with applicable processes $P_\delta(s)$, we can compute the state s' resulting from the application of all the applicable processes as $s' = \gamma(P_\delta(s), s)$ such that $s'(v) = s(v)$ for each $v \in V_\mathbb{B}$ (since propositional variables can't continuously change) and

$$s'(x) = s(x) + \sum_{\varphi \in \Phi_\delta(x,s)} s(\varphi) \quad \text{for each } x \in V_\mathbb{R}, \qquad (1)$$

where $\Phi_\delta(x, s) = \{\varphi_1 \cdot \delta \mid x := x + \varphi_1 \cdot \delta \in \mathrm{eff}(p), p \in P_\delta(s)\}$ is the set of discrete changes to x applicable from s.

DH Plan Validity. Let δ be a discretisation step and Π_δ a DH planning task. A DH *plan* π of Π_δ is a sequence of length n of timestamped actions where $\mathrm{action}_\pi(k) \in A$ is the k-th action of the plan which is applied at the timestamp $\mathrm{time}_\pi(k) \in \{i \cdot \delta \mid i \in \mathbb{N}\}$, with $k \in \{0, \ldots, n-1\}$. The plan is coupled with a value $M_\pi \in \mathbb{Q}_{\geq 0}$ which represents the *make span* of the plan, i.e., when the last action or event triggers or process terminates, in general $M_\pi \geq \mathrm{time}_\pi(n-1)$. Since the plan only contains actions, to validate it, we need to emulate the behaviour of events and processes to understand how the state of variables changed over time. For this reason, we project the plan onto a *history*, where time is discretised into intervals of size δ. A history $\mathcal{H} = \langle \mathcal{S}_1; \ldots; \mathcal{S}_m \rangle$ is an ordered list of *situations*. A situation $\mathcal{S}_i$ is a tuple $\langle t_i, c_i, s_i, a_i \rangle$ where $t_i \in \mathbb{Q}_{\geq 0}$ is the time of $\mathcal{S}_i$, $c_i \in \mathbb{N}$ is a counter, keeping track of the number of actions projected until $\mathcal{S}_i$, s_i is the state of the situation $\mathcal{S}_i$, and $a_i \in A \cup \{\epsilon\}$

is the action which is applied at $\mathcal{S}_i$, with ϵ be a no-op action with empty preconditions and effects. The following rules specify how to construct in a forward manner the plan projection for Π_δ:

$A0$ $\mathcal{S}_1 = \langle t_1, c_1, s_1, a_1\rangle$ with $t_1 = 0$, $c_1 = 0$ and $s_1 = I$,

and for each two contiguous situations $\mathcal{S}_i = \langle t_i, c_i, s_i, a_i\rangle$ and $\mathcal{S}_{i+1} = \langle t_{i+1}, c_{i+1}, s_{i+1}, a_{i+1}\rangle$ with $i \in \{1, \ldots, m-1\}$

$A1$ if $E(s_i) \neq \emptyset$ then $a_i = \epsilon$, $s_{i+1} = \gamma(E(s_i), s_i)$, $c_{i+1} = c_i$ and $t_{i+1} = t_i$,

$A2$ if $E(s_i) = \emptyset$, and $\text{time}_\pi(c_i) = t_i$ then $a_i = \text{action}_\pi(c_i)$, $s_{i+1} = \gamma(a_i, s_i)$, $t_{i+1} = t_i$ and $c_{i+1} = c_i + 1$,

$A3$ if $E(s_i) = \emptyset$, $\text{time}_\pi(c_i) \neq t_i$ and $t_i < M_\pi$ then $a_i = \epsilon$, $s_{i+1} = \gamma(P_\delta(s_i), s_i)$, $t_{i+1} = t_i + \delta$ and $c_{i+1} = c_i$.

Rule $A0$ describes the initial situation, in which no action is yet projected, and the state is equal to the initial condition. After the initialisation, one of three things can happen: either events trigger, an action is applied, or processes are applied, bringing the time forward. Rule $A1$ states that if events can be triggered, they *must* be triggered, keeping the time still. If applying the events causes other events to become applicable, $A1$ is applied at all the following situations until the set is empty. Rule $A2$ states that if no event is applicable and the next action in the plan (based on the counter c_i) happens at the current time t_i, we apply it. If, after applying the action, other events are applicable, rule $A1$ is invoked until no event can trigger. Finally, when all events have triggered and all the actions at that time (if any) have been applied, we can apply the processes with rule $A3$, moving the time forward by δ.

Let π be a plan of a DH planning task Π_δ, and let $\mathcal{H}^\pi_\delta = \langle \mathcal{S}_1; \ldots; \mathcal{S}_m\rangle$ be a projection of π with discretisation step δ. We say that π is valid w.r.t. the discretisation step δ iff (i) for each $\mathcal{S}_i = \langle t_i, c_i, s_i, a_i\rangle$ with $i \in \{1, \ldots, m-1\}$ we have $s_i \models \text{pre}(a_i)$, and (ii) in $\mathcal{S}_m = \langle t_m, c_m, s_m, a_m\rangle$ we have $s_m \models G$.

A log of a projection is a pair $\langle t, H\rangle$ where t is the timestamp associated with the happening $H \in A \cup \mathcal{P}(P) \cup \mathcal{P}(E)$, where $\mathcal{P}(\cdot)$ is the power set of its argument, i.e., a single action, a set of processes or a set of events. We now see how a DH projection of a plan induces a trace of logs. Let $\mathcal{H}^\pi_\delta = \langle \mathcal{S}_1; \ldots; \mathcal{S}_m\rangle$ the projection of a plan π for a DH planning task Π_δ. The trace log induced by $\mathcal{H}^\pi_\delta$ is defined as the sequence of logs $\langle \mathcal{L}_1; \ldots; \mathcal{L}_m\rangle$ where, for each $\mathcal{S}_i = \langle t_i, c_i, s_i, a_i\rangle$ we have that $\mathcal{L}_i = \langle t_i, H_i\rangle$, where:

$$H_i = \begin{cases} \{a_i\} & \text{if } a_i \neq \epsilon \\ E(s_i) & \text{if } a_i = \epsilon \text{ and } E(s) \neq \emptyset \\ P_\delta(s_i) & \text{if } a_i = \epsilon \text{ and } E(s) = \emptyset. \end{cases}$$

Numeric Planning. Numeric planning can be viewed as a special case of a hybrid planning task Π_δ, in which time is static ($\delta = 0$) and both events and processes are absent, i.e., $\Pi_0 = \langle V_\mathbb{B}, V_\mathbb{R}, I, G, A, E = \emptyset, P_\delta = \emptyset\rangle$. A numeric plan π of Π_0 is a special case of a hybrid plan in which all the $\text{time}_\pi(k) = 0$, for $k \in \{0, \ldots, n-1\}$. The concept of trace log, which coincides with π, can thus be straightforwardly applied to numeric planning tasks.

Example 1 (Running Example)**.** *A car needs to move in a straight line from its initial position d_{init} to a desired goal distance between d_{min} and d_{max}. Initially, the agent can only execute the action $turnOn$ to start the engine. At any point, it can gas or $brake$, which increases or decreases its acceleration a by one, until reaching the maximum acceleration A or maximum deceleration D. Two processes, $speed$ and $move$, oversee updating the velocity v based on the acceleration a and the distance d based on the velocity v. The car features stop-start technology, which triggers the engine to switch off when the velocity is near 0. Let us assume $d_{init} = 0$, $d_{min} = 2$, $d_{max} = 4$, $A = 1$, and $D = -1$. The* DH *planning task Π^{LC}_δ for the Linear Car domain is composed of:*[1]

$$\begin{aligned} V_\mathbb{B} &= \{on\}, \qquad V_\mathbb{R} = \{d, v, a, A, D\}, \\ I &= \{d := 0, v := 0, a := 0, on := \bot, A := 1, D := -1\}, \\ G &= \{d \geq 2, d \leq 4, on = \bot\},\ A = \{turnOn, gas, brake\}, \\ E &= \{idle\}, \qquad P_\delta = \{move, speed\}, \\ turnOn &= \langle \{on = \bot\}, \{on := \top\}\rangle, \\ gas &= \langle \{on = \top, a < A\}, \{a := a + 1\}\rangle, \\ brake &= \langle \{on = \top, a > D\}, \{a := a - 1\}\rangle, \\ idle &= \langle \{v < 0.1, a < 0.1, on = \top\}, \{on := \bot\}\rangle, \\ speed &= \langle \{on = \top\}, \{v := v + a \cdot \delta\}\rangle, \\ move &= \langle \{on = \top\}, \{d := d + v \cdot \delta\}\rangle. \end{aligned}$$

A valid plan for Π^{LC}_δ with $\delta = 1$ and $M_\pi = 4$ is, among others,

$$\pi = \langle \langle 0, turnOn\rangle; \langle 0, gas\rangle; \langle 1, brake\rangle; \langle 3, brake\rangle\rangle.$$

The trace log $\mathcal{T}^\pi_\delta$ produced by the history $\mathcal{H}^\pi_\delta$ consists of

$$\begin{aligned} \mathcal{T}^\pi_\delta = &\langle \langle 0, \{turnOn\}\rangle; \langle 0, \{gas\}\rangle; \langle 0, \{move, speed\}\rangle; \\ &\langle 1, \{brake\}\rangle; \langle 1, \{move, speed\}\rangle; \langle 2, \{move, speed\}\rangle; \\ &\langle 3, \{brake\}\rangle; \langle 3, \{move, speed\}\rangle; \langle 4, \{idle\}\rangle\rangle \end{aligned} \quad (2)$$

Initial Condition Retrieving Problem

We now examine the scenario where a trace log, generated by a plan, is observed. We aim to deduce the initial state that best explains this trace log. Depending on the applied context, we may lack complete or correct information about the initial state, so we aim to reconstruct the most plausible state based on the observed log. To do this, we define the *Initial Condition Space* (ICS) as the set of candidate initial states capable of generating the observed trace log, from which the most preferred state is selected according to specific criteria.

Let V be a generic set of variables with a specific domain. We denote as $\text{Asgn}(V)$ the set of all possible assignments to the variables of V within their respective domains.

Definition 1 (Initial Condition Space)**.** *Let $\mathcal{T}_\delta$ be a trace log for a* DH *planning task Π_δ, having variables $V_\mathbb{B}$ and $V_\mathbb{R}$. The* Initial Condition Space (ICS) *for $\mathcal{T}_\delta$ and Π_δ is defined as:*

$$\text{ICS}(\mathcal{T}_\delta, \Pi_\delta) = \{I \in \text{Asgn}(V_\mathbb{B} \cup V_\mathbb{R}) \mid \textsc{accepted}(\Pi_\delta, \mathcal{T}_\delta, I)\}.$$

[1] To simplify notation, we write, for example, $d \leq 4$ instead of $-d + 4 \geq 0$

The ICS for a trace log $\mathcal{T}_\delta$ is defined as the set of all states that, when used as initial states, allow for the trace to be correctly emulated and the goal to be achieved.

To determine acceptance, we simulate the trace log and check for valid plan execution by sequentially applying the recorded happenings in $\mathcal{T}_\delta$ to a state, thus mimicking the system's evolution over time. This process uses the transition function γ, which also constructs plan projections. Emulating a log entry $\mathcal{L} = \langle t, H \rangle$ from state s produces a new state s' defined as $s' = \gamma(\mathcal{L}, s) = \gamma(H, s)$. Emulating the entire trace log $\mathcal{T} = \langle \mathcal{L}_1, \ldots, \mathcal{L}_m \rangle$ from initial state s results in a final state s' given by $s' = \gamma(\mathcal{T}, s) = \gamma(\mathcal{L}_m, \gamma(\ldots, \gamma(\mathcal{L}_1, s)))$.

A trace log $\mathcal{T}_\delta$ for Π_δ emulated from I is accepted, denoted $\text{ACCEPTED}(\Pi_\delta, \mathcal{T}_\delta, s)$, if the preconditions for the logged actions, events, and processes are satisfied during emulation, and the goal is achieved in the final state.

We observe that, as the ICS is defined, it may include initial states that are not relevant, such as assigning numeric variables to values that would be impractical in real-world scenarios (e.g., a temperature below $0°C$ or above $100°C$). To address this, we assume that each numeric variable $v \in V_\mathbb{R}$ has a domain $\mathbb{D}_v = [\underline{v}, \overline{v}]$, where $\underline{v} \in \mathbb{R} \cup \{-\infty\}$ and $\overline{v} \in \mathbb{R} \cup \{+\infty\}$, with $\underline{v} \leq \overline{v}$. This defines the lower and upper bounds for the variable v. We denote the set of domains as $\mathbb{D} = \{\mathbb{D}_v \mid v \in V_\mathbb{R}\}$. This imposition of bounds aligns with recent studies in bounded numeric planning, which explore restrictions on numeric variables (Gigante and Scala 2023; Kuroiwa, Shleyfman, and Beck 2023). To incorporate this domain knowledge, we generalise Definition 1 to the *bounded initial condition space* $\text{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$, where we also require that the accepted states have numeric variables constrained within their specified bounds. Note that the definition provided so far is a special case of $\text{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ where we have, for each $v \in V_\mathbb{R}$, $\mathbb{D}_v = [-\infty, +\infty]$.

With the ICS defined, the next step is to formalise the *Initial Condition Retrieving Problem* (ICR): identifying the initial state within the ICS closest to a given partial or incorrect initial state and repairing it as needed.

Let $\hat{s}$ be a *partial assignment* of the variables in $V_\mathbb{B}$ and $V_\mathbb{R}$ to their respective domains, denoting with $\hat{s}(v) = \varnothing$ if the assignment to the variable $v \in V_\mathbb{B} \cup V_\mathbb{R}$ is undefined, and let $d(a, b)$ be a distance function that measures the difference between two values. The ICR problem seeks to find an initial state I_R within the ICS that minimises the distance from s.

Definition 2 (Initial Condition Retrieving Problem)**.** *Let $\mathcal{T}_\delta$ be a trace log for a* DH *planning task Π_δ, with an initial (possibly partial) state $\hat{I}$ and numeric variables bounded in $\mathbb{D}$. The* Initial Condition Retrieving Problem (ICR) *seeks to identify an initial state I_R such that:*

$$I_R = \underset{I \in \text{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})}{\arg\min} \sum_{v : \hat{I}(v) \neq \varnothing} d(I(v), \hat{I}(v)). \tag{3}$$

This formulation allows us to address three scenarios: (i) retrieving an arbitrary state from the ICS when all variables are undefined; (ii) filling the undefined variables while respecting partial assignments; (iii) repairing an initial state that fails to validate the trace log by finding the closest valid state. (Note that the undefined variables are excluded from the computation, and any assignment is valid as long as it satisfies the condition of belonging to a state within the ICS).

Various custom metrics can define the measure $d(a, b)$. In this work, we utilise the following specific distance:

$$d(a, b) = \begin{cases} (a-b)^2 & \text{if } a, b \in \mathbb{R}, \\ 0 & \text{if } a, b \in \{\top, \bot\} \text{ and } a = b, \\ \infty & \text{if } a, b \in \{\top, \bot\} \text{ and } a \neq b. \end{cases} \tag{4}$$

This measure ensures that the propositional partial assignments must be respected exactly. In this work, we chose this distance since we mostly focus on the retrieval of the numeric component. Other distances can be defined depending on the problem at hand.

To ensure that the ICS is well-defined, we assume it forms a closed set. We will discuss how to guarantee this property in a later section.

Constructing the Initial Condition Space

This section outlines a methodology for constructing a set of numeric constraints, expressed as numeric conditions, designed to capture all traces associated with the accepted runs of a trace log for a DH planning task. To do so, we introduce a constraint to ensure the goal is achieved, a constraint for each happening that guarantees both preconditions and effects are satisfied, and, possibly, constraints for enforcing the bounding of the numeric variables. This approach parallels planning as SAT (Kautz and Selman 1992; Rintanen 2012), drawing on encoding the planning task as a set of logical constraints. Further, it resembles the methodology proposed by Cashmore, Magazzeni, and Zehtabi (2020) for PDDL+ planning as SMT, but differently tailored for our purposes.

We start by defining the variables involved in the constraint set definition. Let $\mathcal{T}_\delta = \langle \mathcal{L}_1; \ldots; \mathcal{L}_m \rangle$ be a trace log of length m for a DH planning task having variables $V = V_\mathbb{B} \cup V_\mathbb{R}$. We define the set of numeric variables as

$$\Xi = \bigcup_{i=0}^{m} \Xi_i, \text{ with } \Xi_i = \{\xi_i^v \mid v \in V_\mathbb{B} \cup V_\mathbb{R}\}.$$

Each subset Ξ_i represents the numeric encoding of the variables V after applying the i-th log $\mathcal{L}_i$ in $\mathcal{T}_\delta$, where Ξ_0 corresponds to the initial state. For a numeric expression φ over $V_\mathbb{R}$, we denote $\Xi_i[\varphi]$ as the expression obtained by replacing each $v \in V_\mathbb{R}$ in φ with its corresponding variable ξ_i^v in Ξ_i. A *condition state* σ is a total assignment of values to the variables in Ξ, mapping them to $\mathbb{R}$. Given a condition state σ, a variable $\xi_i^v \in \Xi$, and a numeric expression ψ over Ξ, we use $\sigma(\xi_i^v)$ and $\sigma(\psi)$ with a semantics consistent with the one provided for V. Since the variables V and their copies Ξ_i have different domains, we define a mapping from condition states in $\text{Asgn}(\Xi_i)$ to states in $\text{Asgn}(V)$. Given a condition state σ, this mapping is denoted by $\mu_i(\sigma)$. Specifically, for $\{\xi_i^v \mid v \in V_\mathbb{B}\}$, positive values map to $\top$, and non-positive values map to $\bot$. Variables $\{\xi_i^v \mid v \in V_\mathbb{R}\}$ map directly to their assigned values.

The variables Ξ can be involved in the definition of *numeric constraints* of the form $\psi \trianglerighteq 0$. Let $\mathcal{C}(\Xi)$ be a set of such

constraints. We say that a condition state σ models $\mathcal{C}(\Xi)$, denoted $\sigma \models \mathcal{C}(\Xi)$, if for every constraint $\psi \trianglerighteq 0 \in \mathcal{C}(\Xi)$, the expression $\sigma(\psi) \trianglerighteq 0$ holds. The *mapping set* of a constraint set $\mathcal{C}(\Xi)$ is defined as:

$$M_i(\mathcal{C}(\Xi)) = \{\mu_i(\sigma) \mid \sigma \in \mathrm{Asgn}(\Xi), \sigma \models \mathcal{C}(\Xi)\}.$$

In essence, $M_i(\mathcal{C}(\Xi))$ contains the states obtained by mapping all condition states σ that satisfy $\mathcal{C}(\Xi)$, taking into account the subset Ξ_i of Ξ referring to the i-th happening.

With the formalism established, given $\mathcal{T}_\delta$, Π_δ, and $\mathbb{D}$, our objective is to construct a set of numeric constraints, denoted as $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ (*initial condition constraint set*), such that each model corresponds to an accepted run of the trace. This implies that for each condition state $\sigma \models \mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$, the value assumed by Ξ_0 of σ corresponds to a state belonging to $\mathrm{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$. Formally, we want to establish the following connection:

$$\mathrm{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D}) = M_0(\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})). \tag{5}$$

In the following, we present the formal definitions of the constraints that will be conjoined in the initial condition constraint set, modelling the goal state description, preconditions, and effects of the happenings in the trace, as well as any specified bounds.

Goal Constraint Set Let G be the goal of a DH planning task. Let $\Xi_m \subseteq \Xi$ be the variables referring to the last happening of the trace. We define the *goal constraint set of* G as $\mathcal{G}(\Xi_m, G) = G^{\mathbb{B}} \cup G^{\mathbb{R}}$, where

$$G^{\mathbb{B}} = \{\xi_m^v = \varrho(e) \mid v = e \in G, v \in V_{\mathbb{B}}\}$$
$$G^{\mathbb{R}} = \{\Xi_m[\varphi] \trianglerighteq 0 \mid \varphi \trianglerighteq 0 \in G\},$$

and $\varrho(e) = 1$ when $e = \top$, -1 otherwise.

The following lemma establishes the connection between the satisfaction of a goal G by a state s_m and its inclusion in the corresponding goal constraint set $M_m(\mathcal{G}(\Xi_m, G))$, modulo the mapping of variables.

Lemma 1. *Let G be a goal, s_m a state, and $\mathcal{G}(\Xi_m, G)$ the goal constraint set of G defined over Ξ_m. Then:*

$$s_m \models G \iff s_m \in M_m(\mathcal{G}(\Xi_m, G)).$$

Proof Sketch. ($\Rightarrow$) To prove the claim, we construct a condition state σ based on s_m such that $\sigma \models \mathcal{G}(\Xi_m, G)$. First, for each $v \in V_{\mathbb{B}}$: if $v = e \in G$, $s_m(v) = e$ since $s_m \models G$; additionally, by the definition of $\mathcal{G}(\Xi_m, G)$, $\xi_m^v = \varrho(e) \in G^{\mathbb{B}}$. Thus, we define $\sigma(\xi_m^v) = \varrho(s(v))$, which ensures that $\sigma \models G^{\mathbb{B}}$. For the goal numeric conditions, consider each $\varphi \trianglerighteq 0 \in G$: since $s_m \models G$, $s_m(\varphi) \trianglerighteq 0$; by definition of $\mathcal{G}(\Xi_m, G)$, we also have $\Xi_m[\varphi] \trianglerighteq 0 \in G^{\mathbb{R}}$. Since φ and $\Xi_m[\varphi]$ are the same formula but with different variables, setting $\sigma(\xi_m^v) = s_m(v)$ for each v in φ guarantees that $\sigma \models G^{\mathbb{R}}$. All variables not involved in G can take any value and we set $\sigma(\xi_m^v) = \varrho(s(v))$ if $v \in V_{\mathbb{B}}$, and $\sigma(\xi_m^v) = s(v)$ otherwise. Any variables Ξ_i with $i < m$ can be assigned freely, as they do not appear in $\mathcal{G}(\Xi_m, G)$. Combining these, we conclude that $\sigma \models \mathcal{G}(\Xi_m, G)$. By construction of σ, it is clear that $s_m = \mu_m(\sigma)$, proving the claim.

($\Leftarrow$) The reverse direction mirrors the previous proof with slight adjustments: we define a condition state $\sigma \models \mathcal{G}(\Xi_m, G)$ and map it through Ξ_m, showing that the resulting state meets the goal. □

To define the goal constraint set, we use variables Ξ_m to index the m-th subset of Ξ associated with the final state. For constraints on happenings, we use $\Xi_{i-1} \cup \Xi_i \subseteq \Xi$, where Ξ_{i-1} and Ξ_i represent the state before and after applying the happening h, respectively.

Happening Constraint Set Let h be the i-th happening logged in a trace log and $\Xi_{i-1} \cup \Xi_i \subseteq \Xi$. We define the *happening constraint set of* h as

$$\mathcal{R}(\Xi_{i-1}, h, \Xi_i) = h^{\mathbb{B}}_{\mathrm{pre}} \cup h^{\mathbb{R}}_{\mathrm{pre}} \cup h^{\mathbb{B}}_{\mathrm{eff}} \cup h^{\mathbb{R}}_{\mathrm{eff}} \cup h^{\mathbb{R}}_{\mathrm{frame}}, \text{ where}$$

$$h^{\mathbb{B}}_{\mathrm{pre}} = \{\xi^v_{i-1} = \varrho(e) \mid v = e \in \mathrm{pre}(h), v \in V_{\mathbb{B}}\}$$
$$h^{\mathbb{R}}_{\mathrm{pre}} = \{\Xi_{i-1}[\varphi] \trianglerighteq 0 \mid \varphi \trianglerighteq 0 \in \mathrm{pre}(h)\}$$
$$h^{\mathbb{B}}_{\mathrm{eff}} = \{\xi^v_i = \varrho(e) \mid v := e \in \mathrm{eff}(h), v \in V_{\mathbb{B}}\}$$
$$h^{\mathbb{R}}_{\mathrm{eff}} = \{\xi^v_i = \Xi_{i-1}[\varphi] \mid x := \varphi \in \mathrm{eff}(h), v \in V_{\mathbb{R}}\}$$
$$h^{\mathbb{R}}_{\mathrm{frame}} = \{\xi^v_i = \xi^v_{i-1} \mid v \in V_{\mathbb{B}} \cup V_{\mathbb{R}} \text{ s.t. } v := e \notin \mathrm{eff}(h)\}.$$

Intuitively, we are expressing how the happening h affects the numeric variables Ξ_{i-1} and Ξ_i by specifying the necessary conditions for h to occur ($h^{\mathbb{B}}_{\mathrm{pre}} \cup h^{\mathbb{R}}_{\mathrm{pre}}$) and its resulting effects ($h^{\mathbb{B}}_{\mathrm{eff}} \cup h^{\mathbb{R}}_{\mathrm{eff}}$), while also ensuring that any variables not directly affected by h remain unchanged ($h^{\mathbb{R}}_{\mathrm{frame}}$).

The previous definition only handles the case where we are constructing a constraint set for a single happening $h \in A$. However, processes and events affect the state in parallel (see the $A1$ and $A3$ rules for the plan projection). Then, to represent them as a constraint set, we introduce a *fictitious happening* to address situations where h is a subset of either E or P.

Let s be a state and $E(s) \subseteq E$ be the mutually exclusive applicable events in s. Since we know they are mutually exclusive, we create the fictitious happening

$$\mathrm{squash}(E(s)) = \langle \bigcup_{h \in E(s)} \mathrm{pre}(h), \bigcup_{h \in E(s)} \mathrm{eff}(h) \rangle.$$

We can then construct the happening constraint set of $E(s)$ as $\mathcal{R}(\Xi_{i-1}, \mathrm{squash}(E(s)), \Xi_i)$.

Let $P_\delta(s) \subseteq P$ be the subset of processes active in a state s, i.e., all those processes whose preconditions are satisfied in s. Similarly to $E(s)$, we can create the fictitious happening

$$\mathrm{squash}(P_\delta(s)) = \langle \bigcup_{h \in P_\delta(s)} \mathrm{pre}(h), \mathrm{change}\big(P_\delta(s)\big) \rangle,$$

where $\mathrm{change}(P_\delta(s))$ is, in accordance with Equation (1), the set of numeric effects for each $v \in V_{\mathbb{R}}$

$$v := v + \sum_{\varphi \in \Phi_\delta(v,s)} \varphi.$$

We can then compute the happening constraint set for $P_\delta(s)$ as $\mathcal{R}(\Xi_{i-1}, \mathrm{squash}(P_\delta(s)), \Xi_i)$.

The following lemma shows how two states, s_{i-1} and s_i, are linked by a happening h if and only if both states belong to models satisfying $\mathcal{R}_i$, modulo the mapping of variables.

Lemma 2. *Let h be a happening, s_{i-1} and s_i two states and $\mathcal{R}_i = \mathcal{R}(\Xi_{i-1}, h, \Xi_i)$ be the happening transition constraint of h. Then, $s_{i-1} \models \mathrm{pre}(h) \wedge s_i = \gamma(h, s_{i-1}) \Leftrightarrow s_{i-1} \in M_{i-1}(\mathcal{R}_i) \wedge s_i \in M_i(\mathcal{R}_i)$.*

Proof Sketch. ($\Rightarrow$) To prove this, we construct a condition state σ based on s_{i-1} and s_i that satisfies $\mathcal{R}_i$. Regarding the precondition, since $\mathrm{pre}(h)$ is a set of conditions, like G in Lemma 1, we follow the same methodology: for each variable v we set $\sigma(\xi^v_{i-1}) = \varrho(s_{i-1}(v))$ if $v \in V_{\mathbb{B}}$, $\sigma(\xi^v_{i-1}) = s_{i-1}(v)$ otherwise. This ensures $\sigma \models h^{\mathbb{B}}_{\mathrm{pre}} \cup h^{\mathbb{R}}_{\mathrm{pre}}$. Regarding the effects, for each $v \in V_{\mathbb{B}}$, if $v := e \in \mathrm{eff}(h)$, $s_i(v) = e$ by definition of γ, and $\xi^v_i = \varrho(e) \in h^{\mathbb{B}}_{\mathrm{eff}}$ by definition of $\mathcal{R}_i$. Thus, we set $\sigma(\xi^v_i) = \varrho(s_i(v))$, ensuring $\sigma \models h^{\mathbb{B}}_{\mathrm{eff}}$. For each numeric effect $v := \varphi \in \mathrm{eff}(h)$, we have $s_i(v) = s_{i-1}(\varphi)$ by definition of γ, and $\xi^v_i = \Xi_{i-1}[\varphi] \in h^{\mathbb{R}}_{\mathrm{eff}}$ by definition of $\mathcal{R}_i$. Therefore, for each $v \in V_{\mathbb{R}}$ affected by $\mathrm{eff}(h)$, we set $\sigma(\xi^v_i) = s_{i-1}(\varphi)$, ensuring $\sigma \models h^{\mathbb{R}}_{\mathrm{eff}}$. For each variable v which is not affected by h, $s_i(v) = s_{i-1}(v)$ by definition of γ, and to ensure $\sigma \models h^{\mathbb{R}}_{\mathrm{frame}}$, we set $\sigma(\xi^v_i) = \sigma(\xi^v_{i-1})$. Any variables in Ξ_j with $j \neq i-1$ and $j \neq i$ can be assigned freely, as they do not appear in $\mathcal{R}_i$. Combining these, we conclude that $\sigma \models \mathcal{R}_i$. By construction of σ, it is clear that $s_{i-1} = \mu_{i-1}(\sigma)$ and $s_i = \mu_i(\sigma)$, proving the claim.

($\Leftarrow$) The reverse direction follows by mirroring the previous proof with slight adjustments: we construct a condition state $\sigma \models \mathcal{R}_i$ from its definition. Then, mapping the subset of variables Ξ_{i-1} and Ξ_i onto their respective counterparts, we show that the resulting states s_{i-1} and s_i necessarily satisfy the preconditions and effects of the happening h. □

Bound Constraint Set Let $V_{\mathbb{R}}$ be the set of numeric variables of DH planning tasks bounded in $\mathbb{D}$. The *bound constraint set* for $V_{\mathbb{R}}$ is defined as

$$\mathcal{B}(\Xi, \mathbb{D}) = \{\xi^v_0 \geq \underline{v}, \xi^v_0 \leq \overline{v} \mid \mathbb{D}_v = [\underline{v}, \overline{v}] \in \mathbb{D}\}.$$

Once we have established the constraint sets for representing the goal condition, happenings of the trace log, and the possible bounds, we have all the necessary components to construct the initial condition constraint set as a conjunction of these elements.

Definition 3 (Initial Condition Constraint Set)**.** *Let $\mathcal{T}_\delta = \langle \mathcal{T}_1; \ldots; \mathcal{T}_m \rangle$ be a trace log for a* DH *planning task Π_δ having goal G and numeric variables bounded in $\mathbb{D}$. We define the* initial condition constraint set *as*

$$\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D}) = \mathcal{B}(\Xi, \mathbb{D}) \cup \bigcup_{i=1}^{m} \mathcal{R}(\Xi_{i-1}, H_i, \Xi_i) \cup \mathcal{G}(\Xi_m, G).$$

We now prove that $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ encloses all and only the initial conditions from which, emulating a trace log, we induce an accepted run, proving Equation (5).

Theorem 1 (Correctness and completeness)**.** *Let $\mathcal{T}_\delta$ be a trace log of a* DH *planning task Π_δ having goal G and numeric variables bounded in $\mathbb{D}$. Then:*

$$I \in \mathrm{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D}) \iff I \in M_0(\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})). \quad (6)$$

Proof. We discuss the case where the numeric variables are unbounded, i.e., for each $v \in V_{\mathbb{R}}$, $\mathbb{D}_v = [-\infty, +\infty]$. Let $s_0 = I$ be a (initial) state. Let $\mathcal{R}_i$ be a shortening of $\mathcal{R}(\Xi_{i-1}, H_i, \Xi_i)$ and $\mathcal{G}$ a shortening of $\mathcal{G}(\Xi_m, G)$. We expand Lemma 1 and Lemma 2 over the logs in $\mathcal{T}_\delta$ obtaining

$$s_m \models G \wedge \textstyle\bigwedge_{i=1}^{m} s_{i-1} \models \mathrm{pre}(h) \wedge s_i = \gamma(H_i, s_{i-1}) \iff$$
$$s_m \in M_m(\mathcal{G}) \wedge \textstyle\bigwedge_{i=1}^{m} s_{i-1} \in M_{i-1}(\mathcal{R}_i) \wedge s_i \in M_i(\mathcal{R}_i).$$

From the left-hand side, we recognise that this coincides with emulating $\mathcal{T}_\delta$ from s_0, obtaining an accepted run. On the right-hand side, based on the definition of mapping set, there must exist a condition state σ such that

$$\sigma \models \mathcal{G}(\Xi_m, G) \wedge \bigwedge_{i=1}^{m} \sigma \models \mathcal{R}(\Xi_{i-1}, H_i, \Xi_i). \quad (7)$$

We recall that $\mathcal{G}(\Xi_m, G)$ and each $\mathcal{R}(\Xi_{i-1}, H_i, \Xi_i)$ are sets of constraints in the form $\psi \trianglerighteq 0$ over the variables in $\Xi_0 \cup \ldots \cup \Xi_m$. By Equation (7), the condition state σ must satisfy these constraints. Notably, satisfying all sets of constraints is equivalent to satisfying their union. Thus, we can rewrite Equation (7)

$$\sigma \models \bigcup_{i=1}^{m} \mathcal{R}(\Xi_{i-1}, H_i, \Xi_i) \cup \mathcal{G}(\Xi_m, G) = \mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D}).$$

On the right-hand side of Equation (6) we imposed that $s_0 \in M_0(\mathcal{R}_1)$ and, being $\mathcal{R}_1 \subseteq \mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ it is true that $M_0(\mathcal{R}_1) = M_0(\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D}))$. We conclude that $I = s_0 \in M_0(\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D}))$. In the case of bounded variables, there is a restriction on the accepted states within the ICS, and the constraint set $\mathcal{B}(\Xi, \mathbb{D})$ added to the initial condition constraint set ensures this. □

Closedness of the ICS. As discussed previously, to solve the ICR problem, the ICS must be a closed set. For space reasons, we decided to show all the properties of ICS without restricting it to be close. Now, we close $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$, as standard for mathematical programming approaches, by replacing each $\psi \trianglerighteq 0$ in it with (i) $\psi \geq 0$ if $\trianglerighteq$ is $\geq$, (ii) two formulas $\psi \geq 0$ and $-\psi \geq 0$ if $\trianglerighteq$ is $=$, and (iii) $\psi' \geq 0$ with $\psi' = \psi - \varepsilon$ and $\varepsilon \in \mathbb{Q}$ if $\trianglerighteq$ is $>$. For this reason, $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ is now a closed set of equations in the form $\psi \geq 0$.

Example 2. *We continue our example on the Linear Car planning task Π^{LC}_δ. Consider the trace log $\mathcal{T}^\pi_\delta$ presented in Equation (2). We notice how the processes move and speed can be squashed together as $ms = \langle \{on = \top\}, \{v := v + a \cdot \delta, d := d + v \cdot \delta\} \rangle$. The trace log has length $m = 9$ and then we produce 10 sets of copy variables, i.e., $\Xi_0 \cup \ldots \cup \Xi_9$, such that $\Xi_i = \{\xi^{on}_i, \xi^d_i, \xi^v_i, \xi^a_i, \xi^A_i, \xi^D_i\}$. For the goal condition and each (squashed) happening in the trace, we have:*

$$\mathcal{G}(\Xi_9, G) = \{\xi^{on}_9 = -1\} \cup \{\xi^d_9 \geq 2, \xi^d_9 \leq 4\}$$
$$\mathcal{R}(\Xi_8, \{idle\}, \Xi_9) = \{\xi^{on}_8 = 1\} \cup \{\xi^v_8 < 0.1, \xi^a_8 < 0.1\}$$
$$\cup \{\xi^{on}_9 = -1\} \cup \{\xi^a_9 = \xi^a_8, \xi^v_9 = \xi^v_8, \xi^d_9 = \xi^d_8,$$
$$\xi^A_9 = \xi^A_8, \xi^D_9 = \xi^D_8\}$$
$$\mathcal{R}(\Xi_7, \{ms\}, \Xi_8) = \{\xi^{on}_7 = 1\} \cup \emptyset$$
$$\cup \{\xi^d_8 = \xi^d_7 + \xi^v_7, \xi^v_8 = \xi^v_7 + \xi^a_7\}$$
$$\cup \{\xi^a_8 = \xi^a_7, \xi^A_8 = \xi^A_7, \xi^D_8 = \xi^D_7, \xi^{on}_8 = \xi^{on}_7\}$$
$$\mathcal{R}(\Xi_6, \{brake\}, \Xi_7) = \{\xi^{on}_6 = 1\} \cup \{\xi^a_6 > \xi^D_6\} \cup \{\xi^a_7 = \xi^a_6 - 1\}$$
$$\cup \{\xi^{on}_7 = \xi^{on}_6, \xi^v_7 = \xi^v_6, \xi^d_7 = \xi^d_6, \xi^A_7 = \xi^A_6, \xi^D_7 = \xi^D_6\}$$
$$\ldots$$
$$\mathcal{R}(\Xi_1, \{gas\}, \Xi_2) = \{\xi^{on}_1 = 1\} \cup \{\xi^a_1 < \xi^A_1\} \cup \{\xi^a_2 = \xi^a_1 + 1\}$$
$$\cup \{\xi^{on}_2 = \xi^{on}_1, \xi^v_2 = \xi^v_1, \xi^d_2 = \xi^d_1, \xi^A_2 = \xi^A_1, \xi^D_2 = \xi^D_1\}$$

$$\mathcal{R}(\Xi_0, \{turnOn\}, \Xi_1) = \{\xi_0^{on} = -1\} \cup \emptyset \cup \{\xi_1^{on} = 1\}$$
$$\cup \{\xi_1^a = \xi_0^a, \xi_1^v = \xi_0^v, \xi_1^d = \xi_0^d, \xi_1^A = \xi_0^A, \xi_1^D = \xi_0^D\}$$

We have omitted the other constraint sets for brevity, as they follow the same structure but have different subscripts. We notice that the initial condition I specified in Π_δ^{LC} is such that $I \in \mathrm{ICS}(\mathcal{T}_\delta, \Pi_\delta)$ but so does the initial condition $I' = \{d := 49, v := -12, a := 0, on := \bot, A := 1, D := -1\}$, which allows generating an accepted run but is not desirable since it starts with a negative velocity. For this reason, we impose the bounds $\mathbb{D}_d = [0, \infty]$, $\mathbb{D}_v = [0, \infty]$ and $\mathbb{D}_a = [0, \infty]$, which can be enforced through

$$\mathcal{B}(\Xi_0, \mathbb{D}) = \{\xi_0^d \geq 0, \xi_0^v \geq 0, \xi_0^a \geq 0\}$$

and thus $I \in \mathrm{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ and $I' \notin \mathrm{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$.

ICR as Mathematical Programming

We now show how to practically solve the ICR problem, casting it into a mathematical programming problem and using all the elements provided above.

Let Π_δ be a DH planning task characterised by a flawed initial state s. In this context, a flaw may refer to a state that is either incomplete or one from which it is impossible to generate an accepted run by emulating the given trace log $\mathcal{T} = \langle \mathcal{T}_1; \ldots; \mathcal{T}_m \rangle$. We consider the numeric variables $V_\mathbb{R}$ of Π_δ bounded within $\mathbb{D}$.

To define the initial condition constraint set $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$, we use the copy variables $\Xi = \Xi_0 \cup \ldots \cup \Xi_m$ where $\Xi_i = \{\xi_i^v \mid v \in V_\mathbb{B} \cup V_\mathbb{R}\}$, alongside $\mathcal{T}_\delta$, Π_δ, and $\mathbb{D}$ as outlined in Definition 3. The task of finding the state I_R according to Definition 2 can be formulated as the following mathematical programming problem aimed at determining an appropriate assignment of the variables Ξ such that:

$$\begin{aligned}
\text{minimize} \quad & \sum_{\substack{v \in V_\mathbb{R}: \\ s(v) \neq \varnothing}} \lambda_v \cdot (\xi_0^v - s(v))^2 \\
\text{subject to} \quad & \psi \geq 0 && \text{foreach } \psi \geq 0 \in \mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D}) \\
& \xi_0^v = \varrho(s(v)) && \text{foreach } v \in V_\mathbb{B} \text{ s.t. } s(v) \neq \varnothing,
\end{aligned}$$

where $\lambda_v \in \mathbb{Q}^+$. The objective function mirrors the distance employed in the ICS definition by considering all the numeric variables $V_\mathbb{R}$, ignoring the undefined ones, i.e., $s(v) = \varnothing$. Each term in the sum is weighted by a scaling factor λ_v, which can be used to normalise the numerical values if they have different orders of magnitude. Since, according to the distance function d provided in Equation (4), violating the assignments of $V_\mathbb{B}$ results in an infinite penalty, each ξ_0^v for $v \in V_\mathbb{B}$, is constrained to match the value from the given initial state, i.e., $\varrho(s(v))$. Moreover, the constraints ensure that only the assignments of Ξ satisfying $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ are considered.

Worst-Case Size of ICS. From the construction of the $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ set, and the presented example, one can easily notice that the number of conditions inside of it is in the worst-case proportional to $|\mathcal{T}_\delta| \cdot |V_\mathbb{B} \cup V_\mathbb{R}|$ since, for each log $\mathcal{L}_i$ inside the trace, we have at least one numeric condition for each $\xi_i^v \in \Xi_i$ with Ξ_i being a copy of $V_\mathbb{B} \cup V_\mathbb{R}$. Looking at the Linear Car example, however, we notice how many conditions are equalities among variables in Ξ and can thus be substituted away. We will see in the experimental analysis how the worst-case size is never reached.

Experimental Analysis

The experimental analysis has been performed on well-known hybrid benchmarks (Fox and Long 2006; Capitanelli et al. 2018; Piotrowski et al. 2016; Scala et al. 2016). We have also chosen numeric tasks from the latest IPC (Taitler et al. 2024) with the most interesting numeric effects. Table 1 shows the list of domains and their category, namely *non-linear* (NL) if the numeric effects contain multiplications between the variables in $V_\mathbb{R}$, and *linear* (L) otherwise. To generate the trace log $\mathcal{T}_\delta$, for each planning task we employed the solver ENHSP (Scala et al. 2016) for the hybrid domains and PATTY (Cardellini, Giunchiglia, and Maratea 2024) for the numeric domains, with a time limit of 30 min, to find a valid plan starting from an initial condition $I_\star$, and we collected the generated traces. Table 1 shows, for each planning task, the number of traces found (#), the average number of variables ($|V_\mathbb{B} \cup V_\mathbb{R}|$), and the average length of the trace log ($|\mathcal{T}_\delta|$). Using the traces found by the planners as input, we run two experimental benchmarks. In the first analysis, we *partialised* the original initial condition $I_\star$ by different amounts, and we solved the ICR problem by searching for a solution that covers as much as possible the assigned values while filling the unassigned ones. In the second analysis, we perturbed the initial condition $I_\star$ with some noise and analysed some properties of the retrieved initial conditions. To solve the mathematical programming problem associated with the ICR problem, we used the general-purpose optimiser SCIPY (Virtanen et al. 2020) which can solve non-linear programming problems employing the interior point algorithm (Byrd, Hribar, and Nocedal 1999). To close the ICS set, we employed $\varepsilon = 10^{-6}$. All the experiments have been run on an Intel Xeon Platinum 8000 3.1GHz with 8 GB of RAM.[2]

Partialised Initial Condition. In this experimental setting, for each planning task, we first computed the $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta)$ under the form of a set of constraints using the trace log found by the planners. Table 1 shows the average size of the initial condition constraint set $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta)$, and the average time to construct it for each domain. This confirms the claim made in the previous section about the worst case never being reached; even in the domains WATERING, FARMLAND-FO, HYDROPOWER and HVAC, where either the trace log is very long, there are many variables or the non-linear dynamic makes it difficult to perform the substitutions of the equalities, the size of $\mathcal{I}(\mathcal{T}_\delta, \Pi_\delta)$ remains limited, paying the price of employing more time in its generation. We then solved the ICR problem, searching for the closest initial condition in the ICS which respects a partial assignment as much as possible. We obtained the partial assignment for each domain by partialising the original initial condition $I_\star$, which

[2]The experimental benchmarks are available at https://github.com/maurovallati/ICR-ICAPS2025. The ICR solver is part of the PATTY solver (http://pattyplan.com)

Domain	Domain Stats #	$\lvert V_b \cup V_n \rvert$	$\lvert \mathcal{T}_\delta \rvert$	ICS stats $\lvert \mathcal{I}(\mathcal{T}_\delta, G) \rvert$	Time	ICR stats $\beta = 1$	$\beta = 0.5$	$\beta = 0$
Hybrid								
BAXTER (L)	19	105.4	17.6	5.8	0.1s	0.1s	0.3s	0.4s
DESCENT (NL)	20	17.0	36.5	52.7	0.4s	1.0s	5.2s	15.3s
HVAC (NL)	18	37.5	136.1	43.2	24.9s	7.8s	26.0s	60.4s
LINEARCAR (L)	10	8.0	39.9	14.2	0.1s	0.1s	0.1s	0.1s
SOLARROVER (L)	20	20.0	533.0	2.0	2.1s	0.1s	0.1s	0.1s
Numeric								
COUNTERS (L)	11	12.3	106.4	114.8	0.2s	0.2s	1.5s	12.8s
COUNTERS-FO (L)	19	25.0	219.4	219.2	1.8s	0.6s	6.1s	15.9s
DELIVERY (L)	9	173.9	50.8	13.6	0.4s	0.2s	0.3s	0.5s
DRONE (L)	16	132.2	99.6	155.9	0.8s	3.6s	22.1s	21.3s
EXPEDITION (L)	2	66.0	214.0	108.5	0.9s	0.1s	0.7s	10.0s
FARMLAND (L)	20	42.3	290.6	290.1	1.3s	0.2s	4.0s	57.8s
FARMLAND-FO (L)	18	56.2	961.5	958.0	15.6s	4.5s	32.2s	43.8s
HYDROPOWER (L)	20	4059.0	97.1	66.1	15.7s	0.5s	1.6s	4.1s
MPRIME (L)	14	368.6	126.4	66.1	2.8s	0.2s	0.5s	0.6s
ROVER (L)	15	472.8	86.5	77.9	2.3s	0.1s	0.6s	0.6s
SAILING (L)	18	16.6	648.4	23.6	1.9s	0.1s	1.6s	1.4s
SAILING-FO (L)	20	15.0	581.8	16.4	2.6s	0.1s	1.1s	2.2s
SUGAR (L)	19	178.8	61.2	52.6	0.5s	0.8s	1.0s	0.8s
WATERING (L)	18	50.9	1085.4	201.2	13.8s	1.6s	7.3s	13.3s

Table 1: Solving times required to build the ICS and solve the ICR on hybrid and numeric planning domains. The domains can be linear (L) or non-linear (NL) depending on the form of the numeric effects. The value β represents the percentage of known initial condition.

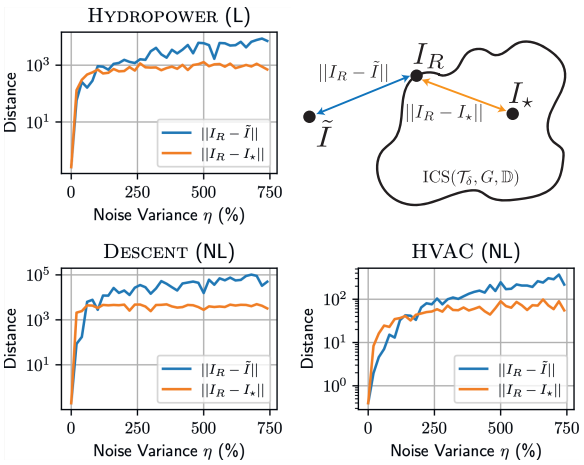

Figure 1: Line chart showing how the distance between the retrieved initial condition I_R, the perturbed initial condition $\tilde{I}$ and the original initial condition $I_\star$ changes based on the noise introduced.

serves as the starting point for generating the trace log computed by the planners. The value β indicates the ratio between the number of assigned values and $|V_\mathbb{B} \cup V_\mathbb{R}|$. (i) When $\beta = 0$, solving the ICR problem involves finding any valid condition within the ICS. (ii) For values of $0 < \beta < 1$, the objective shifts to finding an initial condition within the ICS that fills all unassigned values while adhering as closely as possible to the assigned values of the partial state. (iii) When $\beta = 1$, s is identical to $I_\star$, and the task reduces to verifying that $I_\star$ resides within the ICS.

The ICR statistics columns present the times required to retrieve an initial condition within the ICS for the three scenarios. In all cases, the solutions were found by SCIPY with an associated cost of 0. This is because in cases (i) and (ii) the original initial condition $I_\star$ can always be retrieved, while in case (iii), the cost associated with each condition in the ICS is also 0. Furthermore, all returned initial conditions have been validated against the trace, ensuring that the trace log can be emulated from the corresponding initial condition and lead to a state satisfying the goal. From Table 1, we observe that the problem becomes more challenging as the value of β decreases, indicating that more variables remain unassigned. When information is removed from the objective function, SCIPY has less guidance on where the initial conditions may lie, resulting in increased time to ensure that all constraints are satisfied.

Noisy Initial Condition. Here, we perturbed the initial condition $I_\star$ with Gaussian noise on all the numeric variables, obtaining a perturbed initial condition $\tilde{I}$. More formally, we constructed $\tilde{I}$ such that, for each $x \in V_\mathbb{R}$, we have $\tilde{I}(x) \sim \mathcal{N}(I_\star(x), I_\star(x) \cdot \eta)$ where $\mathcal{N}$ is the Gaussian distribution and $\eta \in \mathbb{R}_{\geq 0}$ adjusts the noise variance. Then, we solved the ICR problem, asking to find a solution inside the bounded ICS $\mathrm{ICS}(\mathcal{T}_\delta, \Pi_\delta, \mathbb{D})$ as close as possible to the perturbed $\tilde{I}$. For each planning task, we specified the domain $\mathbb{D}_x$ for each numeric variable, based on the semantics of the variable. Fig. 1 shows, for a single planning task of three representative domains, due to their interesting numeric effects, how, by varying the noise variance η, the ICR can retrieve an initial condition lying in the ICS. The orange line denotes the distance between the retrieved function I_R and the original condition $I_\star$, while the blue line denotes the distance between I_R and the perturbed condition $\tilde{I}$. It can be noted how, when the noise is low, both lines increase, with the orange line dominating the blue line. When the noise becomes large, the orange line stops growing and remains constant, while the blue line keeps diverging. This indicates that the retrieved I_R has reached the bounds of the ICS, as depicted in Fig. 1 (top-right). By validating each time the retrieved initial condition, this experiment verifies experimentally that all the I_R inside the ICS of it are valid.

Conclusions and Future Work

In this paper, we formally defined the Initial Condition Retrieving problem, which allows us to identify an initial state from which a given plan trace can reliably reach a goal state.

Our contributions are multifaceted. First, we established a formal method for constructing the ICS, with a focus on discretised hybrid planning. Second, we proposed an efficient strategy to tackle the ICR problem by reframing it as a mathematical programming task. Third, we provided empirical evidence of the method's efficacy across a variety of benchmarks. Finally, we demonstrated the adaptability of our approach to numeric planning tasks.

In our future work, we aim to relax some assumptions of this study by enabling the handling of partial traces that may lack events or processes and accommodating noisy data. Additionally, we plan to extend our framework to address more complex formulae for preconditions and goals.

Acknowledgements

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/Z00005X/1]. Matteo Cardellini is partially supported by project FAIR (PE00000013) under the NRRP MUR program funded by the EU - NGEU.

References

Alaboud, F. K.; and Coles, A. 2019. Personalized Medication and Activity Planning in PDDL+. In *ICAPS*, 492–500. AAAI Press.

Bezrucav, S.-O.; Canal, G.; Coles, A.; and Cashmore, M. 2022. Towards Automatic State Recovery for Replanning. In *IntEx Workshop*.

Bogomolov, S.; Magazzeni, D.; Minopoli, S.; and Wehrle, M. 2015. PDDL+ Planning with Hybrid Automata: Foundations of Translating Must Behavior. In *ICAPS*, 42–46.

Boreale, M. 2020. Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial odes. *Sci. Comput. Program.*, 193: 102441.

Byrd, R. H.; Hribar, M. E.; and Nocedal, J. 1999. An Interior Point Algorithm for Large-Scale Nonlinear Programming. *SIAM J. Optim.*, 9(4): 877–900.

Capitanelli, A.; Maratea, M.; Mastrogiovanni, F.; and Vallati, M. 2018. On the manipulation of articulated objects in human-robot cooperation scenarios. *Robotics Auton. Syst.*, 109: 139–155.

Cardellini, M.; Giunchiglia, E.; and Maratea, M. 2024. Symbolic Numeric Planning with Patterns. In *AAAI*, 20070–20077. AAAI Press.

Cardellini, M.; Maratea, M.; Percassi, F.; Scala, E.; and Vallati, M. 2024. Taming Discretised PDDL+ through Multiple Discretisations. In *ICAPS*, 59–67. AAAI Press.

Cardellini, M.; Maratea, M.; Vallati, M.; Boleto, G.; and Oneto, L. 2021. In-Station Train Dispatching: A PDDL+ Planning Approach. In *ICAPS*, 450–458. AAAI Press.

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

Chiari, M.; Gerevini, A. E.; Percassi, F.; Putelli, L.; Serina, I.; and Olivato, M. 2023. Goal Recognition as a Deep Learning Task: The GRNet Approach. In *ICAPS*, 560–568. AAAI Press.

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

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

Gigante, N.; and Scala, E. 2023. On the Compilability of Bounded Numeric Planning. In *IJCAI*, 5341–5349. ijcai.org.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming Up With Good Excuses: What to do When no Plan Can be Found. In *ICAPS*, 81–88.

Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *AIPS*, 44–53. AAAI.

Henzinger, T. A.; and Kopke, P. W. 1999. Discrete-Time Control for Rectangular Hybrid Automata. *Theor. Comput. Sci.*, 221(1-2): 369–392.

Henzinger, T. A.; Kopke, P. W.; Puri, A.; and Varaiya, P. 1998. What's Decidable about Hybrid Automata? *J. Comput. Syst. Sci.*, 57(1): 94–124.

Herzig, A.; de Menezes, M. V.; de Barros, L. N.; and Wassermann, R. 2014. On the revision of planning tasks. In *ECAI*, volume 263, 435–440.

Hoffmann, J. 2015. Simulated Penetration Testing: From "Dijkstra" to "Turing Test++". In *ICAPS*, 364–372.

Kautz, H. A.; and Selman, B. 1992. Planning as Satisfiability. In *ECAI*, 359–363. John Wiley and Sons.

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.

Kuroiwa, R.; Shleyfman, A.; and Beck, J. C. 2023. Extracting and Exploiting Bounds of Numeric Variables for Optimal Linear Numeric Planning. In *ECAI*, 1332–1339.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *AAAI*, 12022–12031.

Parkinson, S.; Khan, S.; and Chrpa, L. 2020. Automated Planning for Administrating Role-Based Access Control. In *SPARK Workshop*.

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

Percassi, F.; Scala, E.; and Vallati, M. 2023a. A Practical Approach to Discretised PDDL+ Problems by Translation to Numeric Planning. *J. Artif. Intell. Res.*, 76: 115–162.

Percassi, F.; Scala, E.; and Vallati, M. 2023b. Fixing Plans for PDDL+ Problems: Theoretical and Practical Implications. In *ICAPS*, 324–333. AAAI Press.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2020. Landmark-based approaches for goal recognition as planning. *Artif. Intell.*, 279.

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic Planning for PDDL+ Domains. In *IJCAI*, 3213–3219.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artif. Intell.*, 193: 45–86.

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

Scala, E.; McCluskey, T. L.; and Vallati, M. 2022. Verification of Numeric Planning Problems Through Domain Dynamic Consistency. In *AI*IA*, volume 13796, 171–183.

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fiser, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Mag.*, 45(2): 280–296.

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 *AAAI*, 3188–3194.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.