Over All, PDDL Semantics is Simultaneously Simple and Hard to Get Right

Nicola Gigante[1] , **Andrea Micheli**[2] , **Enrico Scala**[3] , **Alessandro Valentini**[2]
[1]Free University of Bozen-Bolzano, Italy
[2]Fondazione Bruno Kessler, Trento, Italy
[3]University of Brescia, Italy
nicola.gigante@unibz.it, amicheli@fbk.eu,
enrico.scala@unibs.it, alvalentini@fbk.eu

Abstract

PDDL 2.1 is the community standard for specifications of *temporal planning* problems, involving actions that have a duration and can overlap in time. Recent work has shown that some modelling features, such as *intermediate* and *conditional* effects, can be expressed in PDDL 2.1 by means of specific encodings. At the core of these encodings is a construction that *requires* two events to happen *simultaneously*. However, in practice, almost none of the state-space heuristic search planners known in the literature are capable of finding plans exhibiting this *required simultaneity*, suggesting that the search approach they use is actually *incomplete* with regards to the official PDDL 2.1 semantics. In this paper, we explore this issue both theoretically and experimentally. On the theoretical side, we define two different notions of *required simultaneity*, and we isolate which features of the semantics of PDDL 2.1 allow for such behaviors and how to possibly change the semantics to forbid each of them. In particular, we prove that the crucial detail is how the *over-all* conditions interact with the *mutex* relation. From these observations we isolate the reason why most search-based planners cannot find plans with required simultaneity, and provide an updated search strategy that recovers semantic completeness at the cost of a larger branching factor which, however, can be suitably pruned thanks to an application of our results. On the experimental side, we compare the proposed search strategies, showing that our pruning criterion allows us to recover semantic completeness without significant overhead.

1 Introduction

Automated planning is a field of artificial intelligence whose purpose is to build autonomous agents capable of achieving their *goals* by reasoning how to *act* on their environment. In the subfield of *temporal planning*, actions have a duration and can possibly overlap in time. One of the most common languages used to model temporal planning domains and problems is PDDL 2.1 (Fox and Long 2003), which extends the STRIPS tradition of its predecessor (Ghallab et al. 1998) to the temporal paradigm (and to numeric planning, which we do not consider here).

Recent work (Gigante, Micheli, and Scala 2022) has investigated how some modeling features could be supported by PDDL 2.1 by means of suitable encodings. Such features include *intermediate conditions and effects* (ICE), *i.e.*, the ability of testing conditions or applying effects in the middle of the execution of an action, and *conditional effects*, *i.e.*, the support for actions whose effects depend on some function of the current state. While ICE is not natively supported in PDDL 2.1 at all, conditional effects are, but are not universally supported by planners and search procedures. The encodings found to support these features depend crucially on some particular constructions that enforce some endpoints of specific actions to *happen simultaneously*.

However, when trying to evaluate how such encodings would perform in practice, we discovered that these simultaneity constructions are *not* uniformly supported by the temporal planners most known in literature, including heuristic search planners (such as OPTIC (Benton, Coles, and Coles 2012), TAMER (Valentini, Micheli, and Cimatti 2020), NextFLAP and TFLAP (Sapena, Onaindia, and Marzal 2024)), constraint-based planners (ARIES (Bit-Monnot 2023)), sat-based planners (ITSAT (Rankooh and Ghassem-Sani 2015)) and based on satisfiability modulo theories (Patty (Cardellini and Giunchiglia 2025)). The tested planners either claim no solutions exist for the tested problems, or do not terminate, or exhibit stability bugs. This finding leads to the conclusion that the search strategies used by these planners are accidentally *incomplete* with respect to the official PDDL 2.1 semantics.

In this paper, we explore the issue both theoretically and experimentally. On the theoretical side, we introduce and study the notion of *required simultaneity*, that is, the situation where simultaneous events in a plan cannot be executed sequentially instead. We identify the two distinct notions of *causal* and *temporal* simultaneity, representing distinct ways of forcing simultaneity in a planning problem. Notably, it turns out the tested planners support the latter but not the former. Asking which features of the PDDL 2.1 semantics allow for such behaviors, we find the crucial detail revolves around how the *over-all* conditions of actions interact with the *mutex* relation. By suitably extending the latter, we introduce two semantic restrictions that allow one to forbid each of the two kinds of simultaneity.

Then, we turn to an algorithmic point of view. We model the observed behavior of the tested planners by describing an abstract search procedure which exhibits the same bias against causal simultaneity, explaining the reason of the observed incompleteness. Then, we observe that an easy fix exists which, however, would cause the search branching

factor to grow exponentially. Nevertheless, thanks to the semantic results mentioned above, we can define a pruning criterion that considers the simultaneity of events only when strictly necessary, significantly reducing the branching factor, while maintaining semantic completeness.

Experimentally, we ask how much overhead we need to pay to recover semantic completeness. We evaluate an implementation of the incomplete baseline with the two complete search strategies on a number of standard and novel benchmarks. The results show the pruned strategy basically performs as well as the baseline, introducing almost no overhead, on most benchmarks. This shows our solution may allow the tested planners to be improved to recover semantic completeness with almost no cost.

The paper is structured as follows. After introducing basic notions and background in Section 2, we introduce the notion of required simultaneity in Section 3. Then, Section 4 explains the connection between simultaneity, over-all conditions and the mutex relation, providing semantic restrictions that allow one to forbid each of the two types of simultaneity. Section 5 describes the incomplete search strategy that captures the behavior observed in the tested planners, and the improved strategies that recover completeness, which are experimentally evaluated in Section 6. Section 7 concludes with final remarks and pointers to future work.

2 Preliminaries

We start by introducing *temporal planning* problems as defined in the PDDL 2.1 specification (Fox and Long 2003) (without numeric variables). Given a set P of *propositions*, let $\mathbb{B}_P$ be the set of *Boolean formulas* over P.

Definition 1 (Snap action)**.** *A snap (instantaneous) action is a tuple $h = \langle \mathrm{pre}(h), \mathrm{eff}^+(h), \mathrm{eff}^-(h) \rangle$, where:*

1. $\mathrm{pre}(h) \in \mathbb{B}_P$ *is the* precondition*;*
2. $\mathrm{eff}^+(h), \mathrm{eff}^-(h) \subseteq P$ *are the two* disjoint *sets of* positive *and* negative *effects of h, respectively.*

We write $\mathrm{eff}(h)$ *for* $\mathrm{eff}^+(h) \cup \mathrm{eff}^-(h)$.

Definition 2 (Durative action)**.** *A durative action $a \in A$ is a tuple $\langle a_\vdash, a_\dashv, \mathrm{pre}^{\leftrightarrow}(a), [L_a, U_a] \rangle$, where:*

1. $a_\vdash$ *and* $a_\dashv$ *are the* start *and* end *snap actions, respectively;*
2. $\mathrm{pre}^{\leftrightarrow}(a) \in \mathbb{B}_P$ *is the* over-all condition*;*
3. $L_a \in \mathbb{Q}_{>0}$ *and* $U_a \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$ *are the duration bounds.*

Definition 3 (Temporal planning problem)**.** *A temporal planning problem is a tuple $\mathcal{P} = \langle P, A, I, G \rangle$, where:*

1. P *is a set of* propositions*;*
2. A *is a set of* durative actions*;*
3. $I \subseteq P$ *is the* initial state*;*
4. $G \in \mathbb{B}_P$ *is the goal condition.*

In our presentation, states are subsets s of P which list the true atoms and excludes the false ones (closed-world assumption). Conditions are Boolean formulas over P. The execution starts at the initial state I, and actions in A specify how a state changes when an action is applied and which conditions must be true for the action to be applicable in the first place. The *snap actions* at the start/end events of a durative action, which can be seen as a pair of instantaneous classical planning actions linked together. The duration of an action, *i.e.*, the distance in time between its snap actions, has to belong to the interval $[L_a, U_a]$. Moreover, the *over-all* condition ($\mathrm{pre}^{\leftrightarrow}(a)$) of an action tells what has to hold during the *open interval* of the execution of the action.

What follows is an exposition of the official PDDL 2.1 semantics faithful to those employed also by Gigante et al. (2022) and Gigante, Micheli, and Scala (2022). A collection of action tuples from A, together with a starting time and a duration, is called a plan.

Definition 4 (Plan)**.** *Let $\mathcal{P} = \langle P, A, I, G \rangle$ be a temporal planning problem. A plan for $\mathcal{P}$ is a set $\pi = \{\pi_1, \ldots, \pi_n\}$ of tuples $\pi_i = \langle a_i, t_i, d_i \rangle$, where:*

1. $a_i \in A$ *is a durative action;*
2. $t_i \in \mathbb{Q}_{\geq 0}$ *is its* start time*;*
3. $d_i \in \mathbb{Q}_{>0}$ *is its* duration*.*

In a *valid* plan, also called a *solution* plan, a state must be reached which satisfies the goal condition G. To formally define this concept we need some machinery. Intuitively, an *induced parallel plan* represents a plan as the ordered sequence of all the sets of snap actions that happen together.

Definition 5 (Induced parallel plan)**.** *Let π be a plan for a temporal planning problem $\mathcal{P} = \langle P, I, A, G \rangle$. The* induced parallel plan *for π is the sequence $\pi^{ind} = \langle b_1, \ldots, b_k \rangle$ of pairs $b_i = \langle t_i, S_i \rangle$, where $t_i \in \mathbb{Q}^+$ with $t_i < t_{i+1}$ for all $1 \leq i < k$, and S_i is a set of* snap actions *from A, such that $h \in S_i$ if and only if either:*

1. $h = a_\vdash$ *for some $a \in A$ such that $\langle t_i, a, d \rangle \in \pi$; or*
2. $h = a_\dashv$ *for some $a \in A$ such that $\langle t, a, t_i - t \rangle \in \pi$ for some $t \geq 0$;*

In order for a plan to be realistic and non-contradictory, we need to exclude that two snap actions affecting the same variables can happen at the same time. This is expressed by a *mutual exclusion* (mutex) relation between snap actions.

Definition 6 (Mutex snap actions)**.** *Two snap actions h and z are* mutually exclusive *(mutex) if either:*

1. $\mathrm{pre}(h)$ *mentions any proposition mentioned by* $\mathrm{eff}(z)$*;*
2. $\mathrm{pre}(z)$ *mentions any proposition mentioned by* $\mathrm{eff}(h)$*;*
3. $\mathrm{eff}^+(h) \cap \mathrm{eff}^-(z) \neq \varnothing$*; or*
4. $\mathrm{eff}^+(z) \cap \mathrm{eff}^-(h) \neq \varnothing$.

Note that Definition 6 does *not* involve the propositions mentioned by the *over-all condition* of the actions. This is because over-all conditions are enforced on the *open interval* of the action execution, so the propositions they mention are not *read* and *written to* at the same time.

Given a set of propositions $s \subseteq P$, called a *state*, and a condition $\phi \in \mathbb{B}_P$, we write $s \models \phi$ if s satisfies ϕ under the classical semantics of Boolean formulas. Given a state $s \subseteq P$ and a set S of *non-mutex* snap actions, we define the *application* of S to s, written $S(s)$, as:

$$S(s) = (s \setminus \bigcup_{h \in S} \mathrm{eff}^-(h)) \cup \bigcup_{h \in S} \mathrm{eff}^+(h)$$

In the following we define a *valid plan*, *i.e.*, a solution for a temporal planning problem. The definition is standard but we split it into two parts, which will come useful later.

Definition 7 (Causally valid plans)**.** *Let $\mathcal{P} = \langle P, A, I, G\rangle$ and $\pi = \{\langle a_1, t_1, d_1\rangle, \cdots, \langle a_n, t_n, d_n\rangle\}$ be a temporal planning problem and a plan for $\mathcal{P}$. Consider the induced plan $\pi^{ind} = \langle\langle t'_1, S_1\rangle, \cdots, \langle t'_m, S_m\rangle\rangle$ of π. Then, π is a* causally valid *plan for $\mathcal{P}$ if the following statements hold:*

1. there are no simultaneous mutex snap actions, i.e., *there are no $h, z \in S_i$, with $h \neq z$, for some $1 \leq i \leq m$ such that h and z are* mutex*;*
2. executed snap actions are applicable and reach the goal, i.e., *if $s_0 = I$ and $s_i = S_i(s_{i-1})$ for $1 \leq i \leq m$, we have:*
   (a) $s_i \models \bigwedge_{h \in S_i} \mathrm{pre}(h)$*;*
   (b) $s_m \models G$*,*
3. over-all conditions are respected, i.e., *for all $a \in A$, if $1 \leq i < j \leq m$ are such that $a_\vdash \in S_i$ and $a_\dashv \in S_j$ with no other snap action of a in the middle, then for each $i \leq k < j$ we have that $s_k \models \mathrm{pre}^{\leftrightarrow}(a)$.*
4. actions do not self-overlap, i.e., *there are no $1 \leq i, j \leq n$, with $i \neq j$, such that $a = a_i = a_j$ and $t_i \leq t_j \leq t_i + d_i$.*

Definition 8 (Valid plans)**.** *Let $\mathcal{P} = \langle P, A, I, G\rangle$ and $\pi = \{\langle a_1, t_1, d_1\rangle, \cdots, \langle a_n, t_n, d_n\rangle\}$ be a temporal planning problem and a plan for $\mathcal{P}$. We say π is* valid *if it is* causally valid *and the duration bounds of the actions are respected,* i.e., *for all $1 \leq i \leq n$ we have $L_{a_i} \leq d_i \leq U_{a_i}$.*

The semantics defined above assumes a *dense* time model. In this case, Gigante et al. (2022) proved that, to keep the plan-existence problem decidable, actions must be forbidden to overlap with other executions of themselves, so this is the assumption we make here (Item 4 of Definition 7). Finally, in the terminology of Gigante et al. (2022), we assume the >0-separation semantics, that is, we only require mutually exclusive snap actions to be separated by a non-zero amount of time (Item 1 of Definition 7). The results of this paper apply specifically to >0-separation semantics. If, instead, a minimum required $\epsilon > 0$ distance is given (ϵ-separation semantics), then simultaneity is uncontroversial and quite simple to achieve by either employing *clip actions* (Fox, Long, and Halsey 2004) or *container actions* (Smith 2003).

3 Required Simultaneity

Recent work (Gigante, Micheli, and Scala 2022) has shown that some modeling features, such as *intermediate conditions and effects* (ICE) and *conditional effects*, can be simulated in PDDL 2.1 by ad-hoc encodings, at which core is a construction used to *require* two snap actions to happen *at the same time*. Here, we focus on the concept of *simultaneity* that arise from these constructions, which was only treated as auxiliary in previous works.

To reason about simultaneity we first need some terminology and notation. In what follows, we need to consider each different occurrence of a snap action as a different entity, independently from its timestamp.

Definition 9 (Item)**.** *An* item *of a plan π is a specific occurrence of a snap action appearing in π^{ind}, considered different from each other occurrence. We denote as $M(\pi)$ the set of all its items.*

Two plans can be considered to have the same set of items, if the same snap actions appear with the same multiplicity. A plan π induces two pieces of information about its items.

Definition 10 (Schedule)**.** *The* schedule *of a plan π is a function $\sigma_\pi : M(\pi) \to \mathbb{Q}$ mapping items to their timestamp in π^{ind},* i.e., *if $\langle t, S\rangle \in \pi^{ind}$ with $h \in S$, then $\sigma_\pi(h) = t$.*

Definition 11 (Causal order)**.** *Given two items h and z of a plan π, we write $h <_\pi z$ when $\sigma_\pi(h) < \sigma_\pi(z)$.*

Note that $<_\pi$ is a strict partial order, because two simultaneous items are not comparable. To *linearize* a plan, we separate simultaneous items compatibly with the original causal order of the plan.

Definition 12 (Linearization)**.** *Given a plan π, a* linearization *of π is a plan π' such that:*

1. *the set of items is the same,* i.e., *$M(\pi) = M(\pi')$; and*
2. *the causal order $<_{\pi'}$ of π' is a strict* total *order compatible with $<_\pi$.*

Intuitively, if a plan contains some simultaneous items but it cannot be linearized without breaking plan validity, the simultaneity is not incidental, but *required.*

Definition 13 (Required simultaneity)**.** *Let $\mathcal{P}$ be a temporal planning problem and let π be a valid plan for $\mathcal{P}$. Then:*

1. *π* requires simultaneity *if there is no linearization of π that is a valid plan for $\mathcal{P}$.*

In particular, we have two cases:

2. *π* requires causal simultaneity of *S if there is no linearization of π that is even a* causally valid *plan for $\mathcal{P}$.*
3. *π* requires temporal simultaneity of *S if it requires simultaneity but not causally,* i.e., *linearizations of π that are causally valid plans exist but none of them is a valid plan.*

We say $\mathcal{P}$ requires (causal/temporal) simultaneity *if there exist valid plans that require (causal/temporal) simultaneity.*

One way to obtain plans and problems that require simultaneity is the constructions employed by Gigante, Micheli, and Scala (2022). A slightly simplified version of these constructions is shown in Figure 1. The left side shows how to make the start of two actions happen at the same time. The key idea is to have the *over-all* condition of action a be supported by the effects of the start of action b, and *vice versa*. In this way, $a_\vdash$ cannot happen before $b_\vdash$ because the over-all condition of a would not be satisfied, and similarly, $b_\vdash$ cannot happen before $a_\vdash$. The only solution is for the two snap actions to happen at the same time. Note that this holds without mentioning the duration bounds of a and b, so this construction requires *causal* simultaneity. The construction to link the *end* of two actions is specular (center), while the one to link the end of a and the start of b (right) is a bit more involved. It requires a third auxiliary action, whose over-all condition constrains $a_\dashv$ and $b_\vdash$ to happen together. This

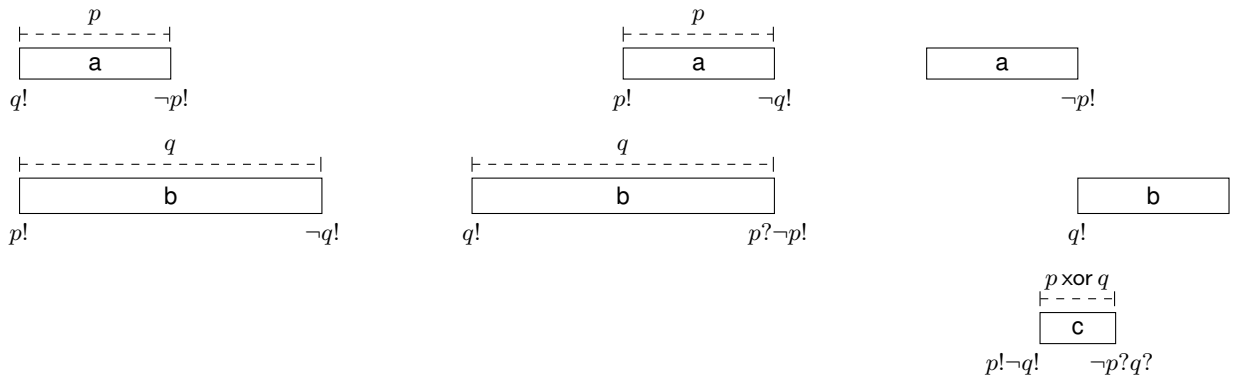
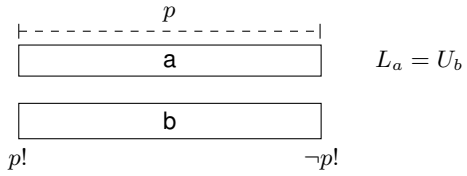


Figure 1: Construction of planning problems requiring the simultaneity of $\{a_\vdash, b_\vdash\}$ (left), $\{a_\dashv, b_\dashv\}$ (center), and $\{a_\dashv, b_\vdash\}$ (right). Question marks denote conditions, exclamation marks denote effects, dashed lines denote over-all conditions. All the mentioned propositions are assumed to be fresh and false at the initial state.



$L_a = U_b$

Figure 2: A construction obtaining plans with required *temporal* simultaneity of $\{a_\vdash, b_\vdash\}$ and $\{a_\dashv, b_\dashv\}$. The proposition p is assumed to be fresh and false at the initial state.

third construction is the most general and can be used to require the simultaneity of any pair of snap actions, not only an end/start pair as in the figure.

A different method to require simultaneity is shown in Figure 2. In this construction the action a is constrained to be executing entirely inside b, but the duration bounds are crucial to enforce simultaneity of the actions extrema. Indeed, any plan where a is executed inside b would be *causally valid*, but not valid in general, therefore this construction requires *temporal* simultaneity.

When trying to evaluate how the encodings of ICE mentioned above would perform in practice, we observed that the constructions of Figure 1 are *not present* in any of the standard planning benchmarks and are *not supported* by most known temporal planners. In particular, we tested OPTIC, TAMER, ARIES, NextFLAP, TFLAP, ITSAT and Patty: all the planners either incorrectly report the problems as unsolvable or diverge in at least one of the constructions. Hence, the search strategy employed by such planners can be considered *semantically incomplete* with regards to the official PDDL 2.1 semantics. In contrast, most of the tested planners (*i.e.*, OPTIC, TAMER, ARIES and Patty) *do work* correctly over the construction in Figure 2, suggesting they support temporal simultaneity, but not causal simultaneity.

It is worth to note that some constructions requiring simultaneity have been observed by Cushing (2012), who pointed at them as evidence of a flaw in the PDDL 2.1 semantics, reasoning about the fact that exact simultaneous execution is not possible in the physical world.

We take a neutral stance on this debate. If required simultaneity is considered a useful feature (for example to simulate ICE), then the results in Section 5 explain the incompleteness of most planners and provide a way to recover completeness without slowing down the search too much. If, instead, simultaneity is considered harmful, then the results in Section 4 point at the specific feature of PDDL 2.1 semantics that makes it possible and provide a syntactic necessary condition for a problem to require simultaneity, possibly helping the debugging of planning domains.

4 Simultaneity and Over-all Conditions

A simple observation worth mentioning now is that *over-all* conditions are crucial to require any form of simultaneity. This is because simultaneous snap actions must be non-mutex, *i.e.*, their conditions and effects cannot talk about each other, so over-all conditions are the only tool available to constrain any ordering relationship between them.

4.1 Forbidding temporal simultaneity

A crucial detail is the fact that over-all conditions are not involved in the definition of the mutex relation. We prove here that if this was not the case, requiring *temporal* simultaneity would be impossible, although *causal* simultaneity would still be an option.

Definition 14 (Eager mutex relation)**.** *Let b be an action of a temporal planning problem $\mathcal{P}$, and let h be a snap action of a different action. If $\mathrm{eff}(h)$ mentions any proposition mentioned by $\mathrm{pre}^{\leftrightarrow}(b)$ we say that:*

1. *h is* eagerly mutex *with $b_\vdash$, written $h \sqsubset b_\vdash$; and*
2. *$b_\dashv$ is* eagerly mutex *with h, written $b_\dashv \sqsubset h$.*

Definition 15 (Eager plan validity)**.** *Given a temporal planning problem $\mathcal{P}$, a (causally) valid plan π for $\mathcal{P}$ is* (causally) eagerly valid *if it contains no simultaneous eagerly mutex snap actions.*

Definition 16 (Eager semantics)**.** *The* eager semantics *of PDDL 2.1 only allows eagerly valid plans.*

We can already observe that the construction in Figure 2 does not work if we restrict ourselves to eagerly valid plans, because $b_\vdash$ is eagerly mutex with $a_\vdash$, *i.e.*, it writes to proposition p which is mentioned by $\mathrm{pre}^{\leftrightarrow}(a)$. A detail that will be important later is that the eager mutex relation is *asymmetric*: in Figure 2, $b_\vdash$ is eagerly mutex with $a_\vdash$, but not *vice versa*. The constructions in Figure 1 on the left and center would not work either, for the same reason as above, but the one on the right works well even when focusing only on eagerly valid plans, since $a_\dashv$ and $b_\vdash$ do not talk about each other at all. Therefore the eager mutex relation only forbids the requirement of temporal simultaneity. To prove that, we need a first lemma.

Lemma 1. *Given an* eagerly valid *plan π for a temporal planning problem $\mathcal{P}$, if one* causally valid *linearization π' exists, then* every *linearization of π is causally valid.*

Proof. Let π be an eagerly valid plan for $\mathcal{P}$ and suppose there exists a causally valid linearization π' of π. Then, consider any other linearization π'' of π. Observe that π'' may differ from π' in two ways:

1. $\sigma_{\pi'}(h) \neq \sigma_{\pi''}(h)$ for some item h but the *causal order* is unchanged, *i.e.*, $h <_{\pi'} z$ iff $h <_{\pi''} z$ for any item h and z; in this case it is straightforward to check that since π' is causally valid, so is π'', by checking that Items 1 to 4 of Definition 7 stay satisfied;
2. the *causal order* is different, *i.e.*, we have h and z such that $h <_{\pi'} z$ but $z <_{\pi''} h$; however, π'' is still a linearization of π, so h and z can only possibly be some items that are simultaneous in π; since π is eagerly valid, h and z are not eagerly mutex with one another; from this observation, it follows that since π' is causally valid, so is π'', which can again be easily shown by checking that Items 1 to 4 of Definition 7 stay satisfied. □

In the statement of Lemma 1, note that π' does not have any simultaneous item by definition, so eager (causal) validity and standard (causal) validity coincide.

Theorem 1. *In the eager semantics, a temporal planning problem cannot require temporal simultaneity.*

Proof. Let $\mathcal{P}$ be a temporal planning problem and let π be eagerly valid plan for $\mathcal{P}$. Suppose there exists a linearization of π that is causally valid but not valid. Then, we show that there exists a linearization of π that is valid.

Because a causally valid linearization is assumed to exist, Lemma 1 shows that every linearization of π is causally valid. Among these, we show that we can find a *rigid* linearization, *i.e.*, a linearization of π where the *durations* of all actions are unchanged, but actions have only been rigidly moved back or forth. By this property, Definition 8 will be satisfied as well, proving that we found a valid plan.

To start, let us define $\delta > 0$ as the minimum distance between non-simultaneous items in π, *i.e.*, the minimum of $t' - t$ among all $t' > t$ such that some $\langle t, S\rangle, \langle t', S'\rangle \in \pi^{ind}$. Furthermore, let $\epsilon = \frac{\delta}{2}$.

Since one causally valid linearization of π is supposed to exist, thanks to Lemma 1 we can start from a specific linearization π' of π chosen appropriately such that $|\sigma_{\pi'}(h) - \sigma_\pi(h)| < \epsilon$ for all items h. Such a linearization π' of π is guaranteed to exist: if an item h is not simultaneous with any other in π, we set $\sigma_{\pi'}(h) = \sigma_\pi(h)$. Otherwise, it is always possible to change $\sigma_{\pi'}(h)$ to decrease the difference with $\sigma_\pi(h)$ while maintaining the relative order with all the other items, thanks to the density of time and the >0-separation semantics.

Now, let a be an action whose duration differs between π' and π, and let h and z be its starting and ending items, respectively. Let us also denote the duration of a in the two plans as $d_\pi(a) = \sigma_\pi(z) - \sigma_\pi(h)$ and $d_{\pi'}(a) = \sigma_{\pi'}(z) - \sigma_{\pi'}(h)$. We can suppose *w.l.o.g.* that the duration decreased (the other case being specular), so $d_{\pi'}(a) < d_\pi(a)$, *i.e.*, $\sigma_{\pi'}(h) > \sigma_\pi(h)$ or $\sigma_{\pi'}(z) < \sigma_\pi(z)$, or both. For the way we chose π' above, we know $\sigma_{\pi'}(h) - \sigma_\pi(h) < \epsilon$ and $\sigma_\pi(z) - \sigma_{\pi'}(z) < \epsilon$, so $d_\pi(a) - d_{\pi'}(a) < 2\epsilon = \delta$. Hence, we can now define a plan π'' equal to π' except that $\sigma_{\pi''}(z) = \sigma_{\pi'}(z) + (d_\pi(a) - d_{\pi'}(a))$, *i.e.*, we increase again the duration of a by adding the necessary length at the end. Since this increase is less than δ, for any other item w, if $z <_\pi w$, then $z <_{\pi''} w$ as well. Therefore π'' is a linearization of π, which is causally valid for Lemma 1. Furthermore, the number of actions whose duration differs in π'' with regards to π has decreased. By iterating this procedure we find a rigid linearization of π, which is a valid plan for $\mathcal{P}$. □

4.2 Forbidding causal simultaneity

Since Theorem 1 tells us which semantic condition may be applied to avoid temporal simultaneity, one may wonder whether a similar condition may be defined to avoid causal simultaneity. Looking at Figure 1, we can identify two different scenarios. The first scenario is exemplified by the constructions on the left and center of Figure 1. There, we have snap actions which are *reciprocally* eagerly mutex, in contrast to Figure 2 where only one snap action was eagerly mutex with the other but not *vice versa*. We formalize and generalize this intuition as follows.

Definition 17 (Cyclically mutex sets)**.** *Given a set of non-mutex snap actions S for a temporal planning problem $\mathcal{P}$, we say that S is* cyclically mutex *if the eager mutex relation forms a* cycle *through all the elements of S.*

Note that the eager mutex relation is antireflexive so a singleton set is never cyclically mutex.

The second scenario is exemplified by the construction on the right of Figure 1, where $a_\dashv$ and $b_\vdash$ are forced to happen at the same time not because they talk about each other, but because happening together is the only way to avoid violating the over-all condition of the c action. The crucial observation here is that, to avoid causal simultaneity in this case, one may forbid to execute together two snap actions that affect *the same overall condition* of another action. This intuition is formalized as follows.

Definition 18 (Externally mutex sets)**.** *Let S be a non-singleton set of snap actions for a temporal planning problem $\mathcal{P}$. We say S is* externally mutex *if there is an action c not involved in S such that* $\mathrm{eff}(h)$ *mentions some proposition mentioned in* $\mathrm{pre}^{\leftrightarrow}(c)$ *for all $h \in S$.*

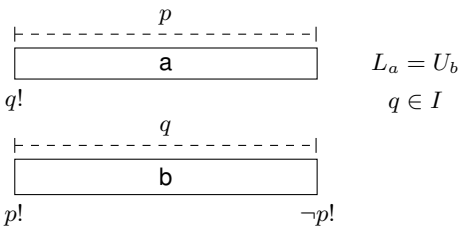


Figure 3: Construction of a plan requiring temporal simultaneity which is not independently valid, as discussed in Theorem 3.

Forbidding both cyclically mutex and externally mutex sets of simultaneous snap actions is what we need to avoid causal simultaneity.

Definition 19 (Independent set)**.** *A set S of snap actions for a temporal planning problem $\mathcal{P}$ is* independent *if none of its subsets are either cyclically or externally mutex.*

Definition 20 (Independent plan validity)**.** *Given a temporal planning problem $\mathcal{P}$, a (causally) valid plan π for $\mathcal{P}$ is (causally)* independently valid *if it contains no independent sets of simultaneous snap actions.*

Definition 21 (Independent semantics)**.** *The* independent semantics *of PDDL 2.1 only allows independently valid plans.*

We can prove now that the independent semantics forbids causal simultaneity as intended.

Theorem 2. *In the independent semantics, a temporal planning problem cannot require causal simultaneity.*

Proof. Let $\mathcal{P}$ be a temporal planning problem and let π be an independently valid plan for $\mathcal{P}$. Then, we must prove that there exists a causally valid linearization of π.

We define such a linearization π' as follows. With $\pi^{ind} = \langle\langle t_1, S_1\rangle, \ldots, \langle t_n, S_n\rangle\rangle$, let some S_i be non-singleton. Since π is independently valid, S_i is independent, *i.e.*, none of its subsets are either externally or cyclically mutex. Then, by the acyclicity there is a total order between the elements of S_i compatible with the eager mutex relation, *i.e.*, there is a linearization π' such that $\sigma_{\pi'}(h) < \sigma_{\pi}(z)$ implies $h \sqsubset z$.

We now show that π' is indeed causally valid, by checking each condition for causal validity from Definition 7.

1. there are no simultaneous mutex snap actions because independent validity requires standard validity, which excludes simultaneous mutex snap actions;
2. since any set $S = \{h_1, \ldots, h_n\}$ of simultaneous snap actions in π must not be mutex, applying them together or sequentially in any order lead to the same state, *i.e.*, for each state s, $S(s) = h_n(\ldots(h_1(s)))$; therefore the same final goal state is reached by π and π'.
3. the satisfaction of the over-all conditions is verified by two observations. For any $\langle t, S\rangle \in \pi^{ind}$ with a non-singleton S:
   (a) for any state s, any subset $S' \subseteq S$, and any running action *a not involved in S'*, since $S(s) \models \mathrm{pre}^{\leftrightarrow}(a)$ and S' is not externally mutex, no pair of elements of S' mention together any propositions mentioned in $\mathrm{pre}^{\leftrightarrow}(a)$; therefore, applying the snap actions in S' in any order keeps satisfying $\mathrm{pre}^{\leftrightarrow}(a)$;
   (b) for any state s, any subset $S' \subseteq S$, any running action *a involved in S'*, and any pair $h, z \in S'$ we have either:
      i. $h \sqsubset z$ because $z = a_{\vdash}$ and h mentions a proposition mentioned in $\mathrm{pre}^{\leftrightarrow}(a)$; by construction we have $\sigma_{\pi'}(h) < \sigma_{\pi'}(z)$, and since $S'(s) \models \mathrm{pre}^{\leftrightarrow}(a)$, h can only affect $\mathrm{pre}^{\leftrightarrow}(a)$ by contributing to its satisfaction, so executing h before z does not violate $\mathrm{pre}^{\leftrightarrow}(a)$;
      ii. $h \sqsubset z$ because $h = a_{\dashv}$ and z mentions a proposition mentioned in $\mathrm{pre}^{\leftrightarrow}(a)$; by construction we have $\sigma_{\pi'}(h) < \sigma_{\pi'}(z)$, so a ends before the execution of z, whose effects are therefore unrelated to $\mathrm{pre}^{\leftrightarrow}(a)$;
      iii. h and z do not mention each other's over-all condition, and they are non-mutex because they were simultaneous, therefore they can be executed in any order without violating the over-all condition of a;
4. when two extrema of two instances of the same action happen at the same time, the action is considered to be self-overlapping, which is forbidden; therefore, if two snap actions are simultaneous in π they must be of different actions, so applying them sequentially in any order does not risk to create any self-overlap. □

Note that the plan in Figure 2 is independently valid, because no external action is involved and the eager mutex relation does not form a cycle between $a_{\vdash}$ and $b_{\vdash}$ (nor between their opposite endpoints). Therefore, the independent semantics only forbids causal simultaneity, and can accept plans requiring temporal simultaneity. However, we cannot say that the independent semantics supports temporal simultaneity either, because some of such plans may not be independently valid. That is because the eager mutex relation is *state-independent*, but if we look at some particular state reached during the execution of a plan, some cyclically mutex set of snap actions *may* still be causally valid when applied sequentially *to that state*, as shown in the following.

Theorem 3. *There are plans requiring temporal simultaneity which are not independently valid.*

Proof. Consider the plan depicted in Figure 3, which is a variant of Figure 2 where $a_{\vdash}$ now mentions the over-all condition of b. In this example, $a_{\vdash}$ and $b_{\vdash}$ are each eagerly mutex with one another, so the set $\{a_{\vdash}, b_{\vdash}\}$ is not independent, and the plan is not independently valid. However, if we assumed that q is true at the initial state as marked in the figure (or, in general, before these actions are executed), the effect of $a_{\vdash}$ is redundant. Therefore, there is still no causal reason why $a_{\vdash}$ should not be scheduled after $b_{\vdash}$, so a causal linearization of the plan exists, *i.e.*, the plan requires temporal simultaneity. □

Theorem 3 shows that a semantic definition precisely separating plans requiring temporal and causal simultaneity requires state-dependent definitions, unlike the eager and cyclic mutex relations defined here. Exploring this semantic path is left as future work.

5 Search Strategies

In this section, we go back to the observation that most of the tested planners are not capable of finding plans that require causal simultaneity (*e.g.*, the constructions in Figure 1), while some of them appear to work well with temporal simultaneity (*e.g.*, the construction in Figure 2). Here, we define an abstract search strategy that exhibits the same behavior, *i.e.*, it is *semantically incomplete* by systematically missing plans requiring causal simultaneity. This search strategy therefore models the heuristic search algorithms implemented by most of the tested planners. We propose an extended search strategy that is semantically complete for the full semantics of PDDL 2.1, including causal simultaneity. We observe that, implemented naively, the complete search strategy would be too slow, having a too large branching factor, so we propose a pruning strategy that shrinks the branching factor again while keeping full semantic completeness.

Algorithm 1 shows the skeleton of an almost standard search-based semi-decision procedure for the plan existence problem. For simplicity, we do not consider the use of search heuristics, and the presented procedure may not terminate when no valid plans exist for a problem (a sound-and-complete instantiation of this algorithm can be found in (Panjkovic, Micheli, and Cimatti 2022)). These simplifications do not affect anything of what follows, though.

The procedure initializes a queue of to-be-visited states starting from the initial one, and pops one state at a time from the queue. When a state satisfying the goal is found, the sequence $\overline{S} = \langle S_1, \ldots, S_m \rangle$ of sets of snap actions that led to that state is given to a constraint satisfaction procedure which obtains a plan π by scheduling the snap actions under the following constraints:

1. the order in the sequence: $\sigma_\pi(h_i) \leq \sigma_\pi(h_j)$ for $i \leq j$;
2. the mutex relation: $\sigma_\pi(h_i) \neq \sigma_\pi(h_j)$ if h_i, h_j are mutex;
3. the duration bounds: $L_a \leq \sigma_\pi(h_j) - \sigma_\pi(h_i) \leq U_a$ if $h_i = a_\vdash$ and $h_j = a_\dashv$.

The scheduling step can commonly be performed using a *simple temporal network* (Dechter, Meiri, and Pearl 1991).

If the current state is not a goal state, it is expanded to find its successors. This is the point where the different search strategies that we consider here differ. The loop at Algorithm 1 ranges over all the *relevant* subsets of non-mutex snap actions of the problem (*i.e.*, the set R), which are collected once at Algorithm 1. The definition of *relevant* subset changes depending on the strategy we are applying, as defined below. Note that the selection of *subsets* of snap actions instead of single ones is crucial for what follows.

Once a subset S has been selected, it is tested for applicability on the current state s.

Definition 22 (Applicability)**.** *A subset S of snap actions is* applicable *to a state s if the following hold:*

1. *all preconditions hold in s,* i.e.*, $s \models \mathrm{pre}(h)$ for all $h \in S$;*
2. *no action already running is started again;*
3. *the over-all conditions are satisfied,* i.e.*, $S(s) \models \mathrm{pre}^{\leftrightarrow}(a)$ for each running action a (including those just started).*

Algorithm 1 Plan existence semi-decision procedure

Require: $\mathcal{P} = \langle P, A, I, G \rangle$ temporal planning problem
1: *queue* $\leftarrow \{I\}$
2: $R \leftarrow$ **relevant subsets** S of non-mutex snap actions
3: **while** *queue* $\neq \varnothing$ **do**
4: $s \leftarrow$ pop from *queue*
5: **if** $s \models G$ **then**
6: $\overline{S} \leftarrow$ seq. of sets of snap actions leading to s
7: **if** $\overline{S}$ is schedulable into a valid plan π **then**
8: **return Yes**
9: **end if**
10: **end if**
11: **for each** $S \in R$ applicable in s **do**
12: $s' \leftarrow S(s)$
13: insert s' in *queue*
14: **end for**
15: **end while**
16: **return No**

Definition 22 assumes the procedure keeps track of running actions on top of the states in the queue, which is not shown in Algorithm 1 for simplicity. If the selected subset is applicable, it is applied to the current state to obtain a successor, which is added to the queue for later processing.

As mentioned above, the different strategies we consider differ for the definition of *relevant subsets*. In the simplest strategy, which is incomplete for causal simultaneity, snap actions are considered singularly.

Definition 23 (Singleton strategy)**.** *In the* singleton strategy*, a subset of snap actions is relevant iff it is a singleton.*

The singleton strategy expands states applying only one snap action at the time. To see how this choice breaks causal simultaneity, consider Figure 1. In the construction on the left, the strategy can choose *e.g.*, to apply $a_\vdash$ first, but it is not applicable because p would not hold after applying it. Similarly, $b_\vdash$ cannot be applied because q would not hold on the resulting state. The state where both p and q hold, resulting from applying $a_\vdash$ and $b_\vdash$ together, can never be visited. A similar situation happens in the construction on the right of Figure 1. Each of $a_\dashv$ and $b_\vdash$ would violate the over-all condition of c which would be running at that time. The state obtained by applying them together can never be reached. Therefore, the singleton strategy is *incomplete*, because it cannot find plans requiring causal simultaneity.

However, note that the singleton strategy successfully finds a plan for the construction in Figure 2. Indeed, the search can choose to apply $b_\vdash$, which sets p, enabling the over-all condition of a, which can be started then. Since $a_\vdash$ and $b_\vdash$ are not mutex, the temporal scheduling step can (and will) choose to schedule them at the same time. We can prove that plans requiring temporal simultaneity are indeed found by the singleton strategy.

Theorem 4. *Let $\mathcal{P}$ be a temporal planning problem whose all plans require temporal simultaneity. The singleton search strategy finds such plans.*

Proof. If a plan π for $\mathcal{P}$ requires temporal simultaneity, by

definition there are linearizations which are causally valid (but none of them is valid). Any such linearization π induces a total order between items, and each item in the total order is applicable by Definition 22 to the state reached by applying the previous items. Therefore, the snap actions can be selected by Algorithm 1 with the singleton strategy in any such total order. Since π requires temporal simultaneity, some items are simultaneous in π. Those items cannot be mutex because π is valid, therefore the are no constraints in the temporal scheduling step enforcing them to be separate, so the original π can be scheduled. □

Theorem 4 proves that the singleton strategy accepts temporal simultaneity, and indeed one can observe that it can find the plan depicted in Figure 3, for the same argument applied in Theorem 3. Therefore, the singleton strategy closely models the intended incomplete behavior: it is incomplete for causal simultaneity but supports temporal simultaneity.

We can now provide a different strategy that is semantically complete for the full semantics of PDDL 2.1, including causal simultaneity.

Definition 24 (Exhaustive strategy)**.** *In the* exhaustive strategy *every subset of snap actions is relevant.*

It is clear that the exhaustive strategy can find all plans, regardless of their required simultaneity, because any possible applicable subset of non-mutex snap actions is tried at each step. Any possible plan π can then be found by applying in order exactly the subsets appearing in π^{ind} and then scheduling them appropriately. This strategy *could* be used to recover semantic completeness in an actual planner but, in practice, it would be unfeasible, because considering all the subsets, of which there can be an exponential number, leads to a huge branching factor.

However, the problem can be mitigated by observing that many of those subsets are useless and can be pruned, obtaining a complete search strategy with a potentially quite lower branching factor.

Definition 25 (Pruned strategy)**.** *In the* pruned strategy*, a subset of snap actions is relevant if and only if it is either a singleton or* independent*.*

Intuitively, considering subsets of independent snap actions is crucial to construct plans requiring causal simultaneity. If a non-independent subset is applied to a state, its result can also be reached by applying its snap actions separately, or grouped into smaller independent subsets. We can therefore prove that the pruned strategy is semantically complete.

Theorem 5. *Suppose a temporal planning problem $\mathcal{P}$ has a unique solution π that require causal simultaneity. Then, Algorithm 1 with the pruned search strategy finds π.*

Proof. Suppose that $\mathcal{P} = \langle P, A, I, G\rangle$ has (only) some plan requiring causal simultaneity, and let π be one such plan with $\pi^{ind} = \langle\langle t_0, S_0\rangle, \ldots, \langle t_n, S_n\rangle\rangle$ and $\overline{S} = \langle S_0, \ldots, S_n\rangle$ be the sequence of its sets of snap actions. Suppose by contradiction that Algorithm 1 with the pruned search strategy does not find any plan for $\mathcal{P}$, including π. Since the strategy considers as relevant all *independent* subsets of snap actions, there is $\langle t_i, S_i\rangle \in \pi^{ind}$ for some set S_i that is *not* independent. Now, because S_i is, in particular, not cyclically mutex, it can be partitioned into a sequence of disjoint maximal subsets $\langle S_i^0, \ldots, S_i^m\rangle$ such that each S_i^j is either a singleton or independent and there are no $h_j \in S_i^j$ and $h_k \in S_i^k$ with $j < k$ such that $h_k \sqsubset h_j$. That is, $\langle S_i^0, \ldots, S_i^m\rangle$ are the topologically ordered strongly connected components of the eager mutex relation between the elements of S.

Now, let $\overline{S'} = \langle S_0, \ldots, S_{i-1}, S_i^0, \ldots, S_i^m, S_{i+1}, \ldots, S_n\rangle$ be the sequence of sets of snap actions of π with $\langle S_i^0, \ldots, S_i^m\rangle$ injected instead of S_i. The number of non-independent sets in $\overline{S'}$ decreased with respect to $\overline{S}$, so we can iterate and suppose *w.l.o.g.* that $\overline{S'}$ does not have any non-independent set anymore.

Let $\overline{S'} = \langle S'_0, \ldots, S'_{m'}\rangle$. Now, one can check that, because π is valid, by construction, starting from I, each S'_{i+1} is applicable to s_i, where $s_0 = I$ and $s_{i+1} = S_{i+1}(s_i)$. Moreover, all the S'_i are independent and non-mutex. Therefore, Algorithm 1 with the pruned strategy is able to find this sequence of sets of snap actions. Moreover, at Algorithm 1 the sequence will be schedulable, with a possible solution being exactly π. To see this, recall that each S_i^j above was non-mutex with all others $S_i^{j'}$ subsets of S_i, so the temporal scheduling step does not have any reason to schedule them separately, and therefore π is a possible solution. This shows π can be found by the pruned strategy. □

6 Experimental Comparison

To empirically validate the findings of Section 5, we implemented the three strategy variations of Algorithm 1 in a temporal heuristic-search planner, similar to OPTIC (Benton, Coles, and Coles 2012) or TAMER (Valentini, Micheli, and Cimatti 2020). Our planner is written in Python and uses the weighted-A^* algorithm, meaning that the queue of states is ordered according to the function $f(s)$, defined as $f(s) = g(s) + w \times h(s)$, where $g(s)$ is the depth of the state s, $h(s)$ is the heuristic function and w is a weight hyperparameter. In our planner we use the h_{ff} heuristic (Hoffmann and Nebel 2001).

We experimented on four temporal planning benchmarks taken from the 2018 International Planning Competition (which is the latest competition with a temporal planning track). Moreover, since none of the IPC benchmarks exhibits required simultaneity, we created three synthetic benchmarks with this feature. Out of the IPC domains, we considered the MATCHCELLAR, DRIVERLOG, SATELLITE and PARKING (each featuring 20 instances), because on these benchmarks our simple implementation was able to solve more than a few instances. We remark that the purpose of this experimental analysis is not to propose a planner that beats the state of the art, but rather to show the performance impact needed to recover semantic completeness with respect to required simultaneity. As for the synthetic benchmarks, we created three domains (each with 18 instances) inspired by the three constructions shown in Figure 1: SIMULTANEITYATSTART requires a number of sets of actions to start simultaneously with some additional actions required to be performed at least once to reach the goal.

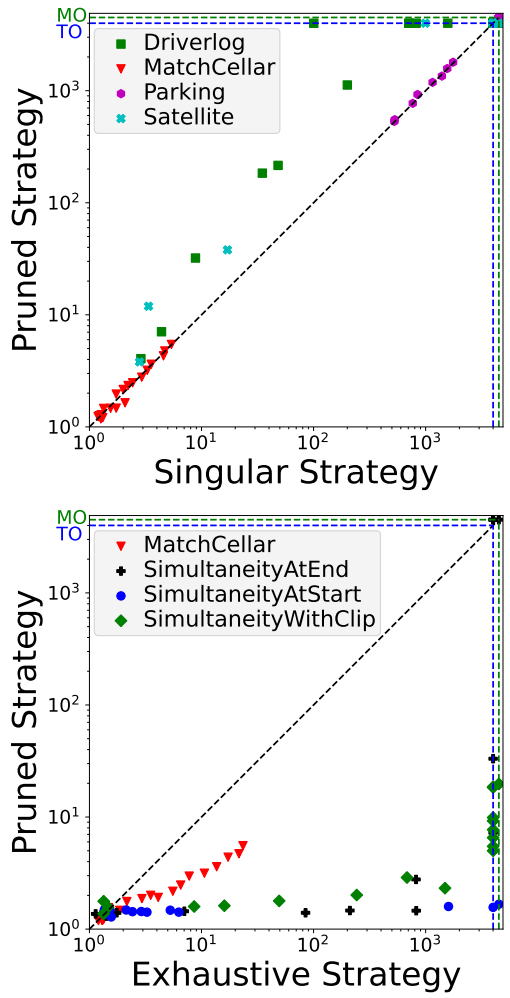


Figure 4: Run-time scatter plot comparing the singleton and pruned strategies on IPC instances (left), and the pruned and exhaustive strategies (right). In the right plot, we omitted all domains in which the exhaustive strategy was unable to solve any instance.

SIMULTANEITYATEND does the same requiring the actions to terminate simultaneously and SIMULTANEITYWITHCLIP requires some pairs of actions to have simultaneous starting and ending (as shown in the right part of Figure 1).

We fixed the weight hyper-parameter to 4: we also experimented with other values, obtaining similar comparative results among techniques, but we reported here the value that was able to solve the higher number of instances. We ran all the experiments on an AMD EPYC 7413 with a 1800s timeout and 20 GB memory limit. All the implementation code and the benchmarks are available in the additional material of this submission.

Figure 4 reports the results of our experiments. The left plot clearly shows that on MATCHCELLAR and PARKING, the pruned and singleton strategies coincide. This is because in these domains there are no independent subsets of snap actions, resulting in the same search space. In DRIVERLOG and SATELLITE, instead, while the problem does not exhibit required simultaneity, there are sets of independent snap actions, causing an increase in the search space size and a consequent slowdown. We note however that the slowdown still permits to solve many of the instances.

We experimentally confirmed that the singleton strategy is incomplete for the synthetic benchmarks: it declares all the problems as unsolvable. The right part of Figure 4 compares the exhaustive strategy and the pruned one. As expected, and as clearly shown by the plot, the pruned strategy is consistently much superior to the exhaustive one. Note that for PARKING, DRIVERLOG and SATELLITE, the pruned strategy is able to solve several instances (as shown on the left plot), but the exhaustive strategy cannot solve any instance on these domains.

Refining the Pruned Strategy Implementation As mentioned in the previous section, a detailed analysis of state-dependent rules and syntactical fragments for refining the subsets of simultaneous actions to be explored is left for future work. However, we noted that the all IPC instances only allow for conditions (both overall or otherwise) expressed as conjunctions of positive literals. In this very specific subcase, we can refine the eager mutex relation as follows, reducing the number of cyclically mutex sets.

1. $h \sqsubset b_{\vdash}$ additionally requires that $\mathrm{eff}(h)$ only has positive effects on propositions mentioned in $\mathrm{pre}^{\leftrightarrow}(b)$;
2. $b_{\dashv} \sqsubset h$ additionally requires that $\mathrm{eff}(h)$ only has negative effects on propositions mentioned in $\mathrm{pre}^{\leftrightarrow}(b)$.

Moreover, we do not need to consider any externally mutex set as per Definition 18 for overall conditions that are conjunctions of literals, because in this syntactical fragment, the guarantees of Theorem 2 are maintained: Item 3a can be adapted because all the conditions that are conjunctions of positive literals trivially satisfy the proof and the refined eager mutex relation above still satisfy the proof for Item 3b. We implemented this specialized variation of the pruned strategy by computing the cyclical mutex sets using the refined relation and only considering non-conjunctive conditions for the externally mutex set computation. We confirmed that in all the IPC cases, this strategy collapses to the singleton strategy and in the synthetic domains it collapses to the pruned strategy, retaining the best performance in all the cases and solving the highest number of instances.

7 Conclusions

This paper introduces and studies the concept of *required simultaneity* in plans for temporal planning problems, that arose from constructions employed in the literature to encode in PDDL 2.1 modeling features such as *intermediate conditions and effects* and *conditional effects* (Gigante, Micheli, and Scala 2022).

When tested, these constructions turned out to be not uniformly supported by most of the temporal planners known in the literature, understandably since they do not appear in any IPC benchmark. Starting from this observation, we studied the situation both theoretically and experimentally.

On the theoretical side, we identified two distinct notions of required simultaneity, and identified several con-

ditions that allows the definition of stricter semantics for PDDL 2.1 where either type of required simultaneity is forbidden. Then, we approached the problem algorithmically, providing an abstract search strategy that exhibits the incomplete behavior found in most of the tested planners. We then proposed a complete search strategy that, by applying our semantic results, can sufficiently prune the branching factor to minimize the performance penalty to pay to recover semantic completeness.

Experimentally, we implemented a prototype heuristic search planner including the discussed search strategies and confirmed experimentally that semantic completeness with regards to required simultaneity can be recovered with an acceptable performance overhead. In particular, we measured the overhead on selected IPC domains where simultaneity is never required, showing that our pruned search strategy performs competitively with the incomplete one.

The conditions applied in the pruned search strategy are *state-independent* and therefore necessarily an overapproximation. Looking for state-dependent conditions that capture exactly all and only the subset of events that necessarily needs to happen simultaneously is left for future work.

Acknowledgements

Andrea Micheli and Alessandro Valentini have been partially supported by the STEP-RL project funded by the European Research Council under GA n. 101115870.

Enrico Scala has been supported by the Italian Ministry of University and Research within the PRIMA 2024 programme project "Optimizing Water Resources in Coastal Areas using Artificial Intelligence" (AI4WATER – D53C25000510006).

References

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *ICAPS 2012*.

Bit-Monnot, A. 2023. Enhancing hybrid CP-SAT search for disjunctive scheduling. In Gal, K.; Nowé, A.; Nalepa, G. J.; Fairstein, R.; and Radulescu, R., eds., *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 255–262. IOS Press.

Cardellini, M., and Giunchiglia, E. 2025. Temporal numeric planning with patterns. In Walsh, T.; Shah, J.; and Kolter, Z., eds., *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, 26481–26489. AAAI Press.

Cushing, W. A. 2012. *When is Temporal Planning Really Temporal?* Ph.D. Dissertation, Arizona State University. https://rakaposhi.eas.asu.edu/cushing-dissertation.pdf.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20:61–124.

Fox, M.; Long, D.; and Halsey, K. 2004. An investigation into the expressive power of pddl2.1. In *ECAI 2004*.

Ghallab, M.; Howe, A.; Knoblock, C.; Mcdermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL—The Planning Domain Definition Language.

Gigante, N.; Micheli, A.; Montanari, A.; and Scala, E. 2022. Decidability and complexity of action-based temporal planning over dense time. *Artif. Intell.* 307:103686.

Gigante, N.; Micheli, A.; and Scala, E. 2022. On the expressive power of intermediate and conditional effects in temporal planning. In Kern-Isberner, G.; Lakemeyer, G.; and Meyer, T., eds., *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022*.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Panjkovic, S.; Micheli, A.; and Cimatti, A. 2022. Deciding unsolvability in temporal planning under action non-self-overlapping. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 9886–9893. AAAI Press.

Rankooh, M. F., and Ghassem-Sani, G. 2015. ITSAT: an efficient sat-based temporal planner. *J. Artif. Intell. Res.* 53:541–632.

Sapena, Ó.; Onaindia, E.; and Marzal, E. 2024. A hybrid approach for expressive numeric and temporal planning with control parameters. *Expert Syst. Appl.* 242:122820.

Smith, D. E. 2003. The case for durative actions: A commentary on pddl2.1. *Journal of Artificial Intelligence Research*.

Valentini, A.; Micheli, A.; and Cimatti, A. 2020. Temporal planning with intermediate conditions and effects. In *AAAI 2020*.