Planning with Uncertain Action Models

Francesco Percassi[1], **Alessandro Saetti**[2], **Enrico Scala**[2]

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

Abstract

Uncertainty over model knowledge is a core challenge in planning and has been addressed through various approaches tailored to different scenarios. In this paper, we focus on scenarios where the agent does not initially know the exact outcome of its actions but gains knowledge upon execution, i.e., each action reveals its actual effect, removing uncertainty about future occurrences. We refer to this formulation as Planning with Uncertain Models of Actions (PUMA). We show that PUMA can be compiled in polynomial time in both Fully Observable Non-Deterministic planning and, perhaps more unexpectedly, classical planning, providing a constructive proof that PUMA remains PSPACE-complete despite its apparent exponential uncertainty. Finally, we experimentally evaluate both compilations with benchmark domains that capture the key aspects of the problem. The results show the practical feasibility of our approach and reveal a complementary behavior between the two compilations.

Code — https://github.com/Plutone/PUMA

Introduction

Planning is a well-known AI problem that involves synthesizing a sequence of actions to guide one or more agents from an initial world state to a desired goal. This problem can be extremely complex, particularly because agents operating in the real world must cope with uncertainty. Consequently, the planning community has devoted substantial effort to developing methods for handling uncertainty.

One approach is to anticipate all possible contingencies at planning time, for example, through conformant or contingent planning models (Smith and Weld 1998; Bonet 2010). These models extend the classical planning paradigm by allowing an action to produce different possible effects each time it is executed.

However, in several applications, this uncertainty is not related to the presence of an environment that continuously plays against the agent (Chrpa and Karpas 2024, 2025), but rather to incomplete information about the action model. For example, in a logistics domain where people and goods move among cities by train, domain experts might not know whether a given train is operating due to a possible strike. Similarly, in a robotic setting where robots move between rooms through doors, they might not know whether a door is open or whether the robot needs to carry its key. If the first attempt to move a given train fails, all subsequent attempts will fail; similarly, if the first attempt to open a door without the key fails, all subsequent attempts will fail. This form of uncertainty can also arise in action model learning (Walsh and Littman 2008), when the available data do not allow convergence to a unique action model.

In this paper, we target this specific kind of uncertainty. That is, we assume that the agent knows an uncertain model of its own actions, but once an action is applied, this model is revealed to the agent. In other words, we assume that the agent has full observability not only of the state obtained after executing an action but also of which effect is actually triggered by the action execution.

In Fully Observable Non-Deterministic Planning (FOND), actions may have multiple possible outcomes, the resulting state is fully observable, and nondeterminism persists across executions. Building on FOND syntax, we introduce a new semantics capturing the ideas discussed above. In our semantics, the first time an uncertain action is applied, the agent perceives which effect is triggered, acquires full knowledge of the action model, and can exploit this information in all subsequent steps up to the goal. We refer to the problem under our new semantics as Planning with Uncertain Models of Actions, PUMA for short.

We show that PUMA can be polynomially encoded both in FOND and, surprisingly, in classical planning. The compilation into FOND is straightforward and relies on making actions deterministic after their first application; the second compilation is substantially more complex, as its target formalism is simpler. The key idea behind the latter is that only a polynomial number of variables is required to keep track of the exponentially many models the agent may need to consider while solving the problem. Intuitively, executing an uncertain action generates a set of tasks that must be completed successively, one for each possible outcome. The compilation introduces variables that track all currently open tasks; since the number of simultaneously open tasks is polynomially bounded, the compilation requires only a polynomial number of variables. As a corollary, this provides a constructive proof that this form of planning with uncertainty is PSPACE-complete.

Beyond its theoretical contribution, the paper also investigates the practical viability of the proposed PUMA formulation through its compilation across four benchmark domains. By supplying both a FOND encoding and a classical planning encoding, we enable the use of a wide spectrum of off-the-shelf planners; in practice, the two approaches prove to be not only workable but complementary.

Problem Formulation

A PUMA problem Π is defined as a tuple $\langle F, A, I, G\rangle$, where F is a set of facts, A is a set of actions defined over F, $I \subseteq F$ is a state called the initial state, $G \subseteq F$ is a partial goal state. We denote by $\mathcal{L}it(F)$ the set of literals with facts in F. An action $a \in A$ is a pair $a = \langle \mathrm{pre}(a), \textsc{Effs}(a)\rangle$, where $\mathrm{pre}(a) \subseteq F$ denotes the precondition of a, while $\textsc{Effs}(a) = \{\mathrm{eff}_1(a), \ldots, \mathrm{eff}_{m_a}(a)\}$ with $\mathrm{eff}_i(a) \subseteq \mathcal{L}it(F)$ for $1 \leq i \leq m_a$ is a set of m_a alternative effects of a. In the following, with $\mathrm{eff}_i^+(a)$ and $\mathrm{eff}_i^-(a)$ we indicate the positive and negative effects in $\mathrm{eff}_i(a)$. When an action has only one effect, i.e., $m_a = 1$, it is called *certain*. Conversely, if $m_a > 1$, the action is called *uncertain*, as its execution can result in one of its multiple effects. With PUMA, alternative sets of effects can be used to encode the uncertainty about how the model of an action actually is.

A state s is a subset of F with the meaning that if $p \in s$ then p is true in s, otherwise it is false. An action a is applicable in s iff $\mathrm{pre}(a) \subseteq s$. The first time an agent executes an uncertain action a, it perceives an action in $\textsc{Dtmn}(a) = \{\langle \mathrm{pre}(a), \mathrm{eff}_i(a)\rangle \mid \mathrm{eff}_i(a) \in \textsc{Effs}(a)\}$. Let $\tau = \langle \hat{a}_1, \ldots, \hat{a}_n\rangle$ where $\hat{a}_i \in \textsc{Dtmn}(a_i)$ for all $1 \leq i \leq n$. We say that τ is well-formed when for every pair of actions $\hat{a}_i, \hat{a}_j \in \textsc{Dtmn}(a)$, if $\hat{a}_i \in \tau$, then $\hat{a}_j \notin \tau$. The execution of a perceived action $\hat{a}$ in a state s results in state $(s \setminus \mathrm{eff}^-(\hat{a})) \cup \mathrm{eff}^+(\hat{a})$, denoted by $s[\hat{a}]$ for short. Executing a sequence of perceived actions $\tau = \langle \hat{a}_1, \ldots, \hat{a}_n\rangle$ in a state s_0 results in the last state in $\langle s_0, \ldots, s_n\rangle$, where $s_i = s_{i-1}[\hat{a}_i]$ assuming $\mathrm{pre}(\hat{a}_i) \subseteq s_{i-1}$ for all $1 \leq i \leq n$. Such a last state is denoted by $s_0[\tau]$, for short. Given a well-formed sequence of perceived actions, the execution of an uncertain action a in $s_0[\tau]$ leads to a set of states defined as follows.

$$\begin{cases} \{s_0[\tau] \setminus \mathrm{eff}^-(\hat{a}) \cup \mathrm{eff}^+(\hat{a})\} & \text{if } \exists \hat{a} \in \textsc{Dtmn}(a) \text{ and } \hat{a} \in \tau, \\ \{s^i \mid s^i = (s_0[\tau] \setminus \mathrm{eff}_i^-(a)) \cup \mathrm{eff}_i^+(a),\ 1 \leq i \leq m_a\} & \text{otherwise.} \end{cases}$$

Intuitively, such a set is a singleton if the action has already been executed and, in this case, is the state resulting from the effects triggered for the action in the past; the set contains all the states resulting from the application of the alternative effects of the action otherwise.

A solution for a PUMA problem Π is an *action strategy* π that guarantees the achievement of the goal G from the initial state I, regardless of which actual effects of uncertain actions are revealed during execution.

Unlike a FOND planning policy, which maps each state to the next action to execute, a solution to our problem maps a sequence of perceived actions to the next action to apply. Formally, a strategy is a partial function $\pi : \hat{A}^* \to A \cup \{\epsilon\}$, where $\hat{A} = \bigcup_{a \in A} \textsc{Dtmn}(a)$. Hence, π associates each finite sequence of perceived deterministic actions with the next action to execute, or with ϵ if no further action is required. Given a strategy π, we denote by $T(\pi)$ the set of execution traces induced by π, namely all sequences of perceived actions that can result from following π starting from the initial state I. Formally, $T(\pi)$ is defined inductively as follows:

1. $\langle\rangle \in T(\pi)$;
2. Let $\tau = \langle \hat{a}_1, \ldots, \hat{a}_i\rangle \in T(\pi)$ the following holds
   (a) if $\pi(\tau) = a$ and $\exists \hat{a} \in \textsc{Dtmn}(a)$ such that $\hat{a} \in \tau$ then $\langle \hat{a}_1, \ldots, \hat{a}_i, \hat{a}\rangle \in T(\pi)$,
   (b) if $\pi(\tau) = a$ and $\nexists \hat{a} \in \textsc{Dtmn}(a)$ such that $\hat{a} \in \tau$ then $\forall \hat{a} \in \textsc{Dtmn}(a), \langle \hat{a}_1, \ldots, \hat{a}_i, \hat{a}\rangle \in T(\pi)$.

A strategy π is a solution to a PUMA problem if for every possible trace $\tau \in T(\pi)$, π is such that: (i) $\pi(\tau) = a \implies \mathrm{pre}(a) \subseteq I[\tau]$, and (ii) $\pi(\tau) = \epsilon \implies G \subseteq I[\tau]$.

From PUMA to FOND

The first compilation we study transforms a PUMA problem into a FOND planning problem. Syntactically, FOND is like PUMA, with the difference that the execution of an action is always non-deterministic, regardless of how many times it occurs. Roughly, a solution to a FOND planning problem is an action policy, i.e., a mapping from states to actions. Such a policy solves a given FOND problem if an agent using it reaches the goal from the initial state, regardless of how the non-deterministic effects unfold during execution. A policy, therefore, induces a set of action sequences, and when it is a solution, each such sequence leads to the goal from the initial state. In this work, we specifically focus on strong solutions (Cimatti et al. 2003). In strong FOND, there is no fairness assumption, so a cyclic policy cannot be a solution.

Intuitively, our compilation to FOND makes each uncertain action of PUMA deterministic after its first execution. We refer to our compilation COMP2FOND. Let $\Pi = \langle F, A, I, G\rangle$ be a PUMA problem. For convenience, we partition the set A into A_{U} and A_{C} where $A_{\mathrm{U}} = \{a \in A \mid m_a > 1\}$ denotes the set of uncertain actions, and $A_{\mathrm{C}} = A \setminus A_{\mathrm{U}}$ is the set of the remaining ones. COMP2FOND takes a PUMA problem Π and generates the FOND problem $\Pi_{\textsc{Comp2fond}} = \langle F', A', I', G\rangle$, where F', A' and I' are defined as follows.

$$F' = F \cup \{\mathsf{w}_a^i \mid a \in A_{\mathrm{U}}, 0 \leq i \leq m_a\},$$
$$A' = A_{\mathrm{C}} \cup A_{\mathrm{U}}^0 \cup A_{\mathrm{U}}^+, \text{where:}$$
$$A_{\mathrm{U}}^0 = \{a^0 \mid a \in A_{\mathrm{U}}\},$$
$$A_{\mathrm{U}}^+ = \{\langle \mathrm{pre}(a) \cup \{\mathsf{w}_a^i\}, \mathrm{eff}_i(a)\rangle \mid a \in A_{\mathrm{U}}, \mathrm{eff}_i(a) \in \textsc{Effs}(a)\},$$
$$I' = I \cup \{\mathsf{w}_a^0 \mid a \in A_{\mathrm{U}}\}.$$

With respect to A_{U}^0, for each uncertain action $a \in A_{\mathrm{U}}$, the compilation derives an action a^0 as follows.

$$\mathrm{pre}(a^0) = \mathrm{pre}(a) \cup \{\mathsf{w}_a^0\},$$
$$\textsc{Effs}(a^0) = \{\mathrm{eff}_i(a) \cup \{\neg\mathsf{w}_a^0, \mathsf{w}_a^i\} \mid \mathrm{eff}_i(a) \in \textsc{Effs}(a)\}.$$

The very first execution of an action a is emulated with a^0. Then, each eff_i of a^0 is extended in a way to enforce the planner, for each non-deterministic branch, to execute the

determinized action associated with eff_i for every successive time the agent executes a; each such action is what is contained in set A^+_{U}.

From PUMA to Classical Planning

This section presents a compilation scheme to transform a PUMA problem into Fully Observable Deterministic planning (FOD). The resulting problem is structured to emulate the solution of multiple (sub)tasks, each referred to as the current task. A "branching" occurs when an uncertain action is executed for the first time in the current task. Executing an action a with m_a effects results in one of the effects of a and a set of auxiliary effects opening of $m_a - 1$ tasks. Each task stores, using novel facts, the state resulting from an alternative effect of a and the identifiers of previously applied uncertain actions. Each open task must be solved subsequently. Solving and closing an open task requires reaching the problem goal. Open tasks are processed using a last-in, first-out strategy: the last open task is the first to be solved, forming a stack. Once the goals in the current task have been achieved, it is closed, and the most recently opened task becomes the new current task. This means that the truth values of the world state facts and the effect identifiers of the current task are overwritten with those previously stored for the last open task.

Let $\Pi = \langle F, A, I, G\rangle$ be a PUMA problem. In the rest of the paper, $N = \Sigma_{a\in A}(m_a - 1)$ denotes the maximum number of tasks that can remain open together when using a last-in, first-out strategy. For convenience, we present our compilation scheme using numeric variables and conditional effects as additional constructs for the target planning problem. Conditional effects can always be (polynomially) compiled away (Nebel 2000; Gerevini, Percassi, and Scala 2024); numeric variables can also be (polynomially) compiled away, provided that they are bounded and discrete (Gigante and Scala 2023). Before proceeding with the presentation, we briefly recall the concepts necessary to define a planning problem with numeric variables and conditional effects (Bonassi et al. 2025). Let $\langle F, A, I, G\rangle$ be a classical planning problem, where F, A, I and G are the same as those previously defined for the PUMA problem, except for the actions in A that are certain, i.e., they have only one set of effects, simply denoted by $\text{eff}(a)$ for an action $a \in A$. We extend the problem with numeric variables X taking values in $\mathbb{Z}$. These variables can be increased, decreased, or set directly through numeric effects of the form $(x \mathrel{+}= q)$, $(x \mathrel{-}= q)$, or $(x := q)$, where $x \in X$ and $q \in \mathbb{N}$. Numeric variables can appear in the preconditions of actions through numeric conditions of the form $(\xi \bowtie 0)$, where ξ is a linear expression over X and $\bowtie \in \{\geq, >, =, <, \leq\}$. Moreover, we extend classical planning with conditional effects $C \triangleright E$, where C is a set of conditions and E a set of effects. (When these sets are singletons, we omit set notation for brevity.)

The compilation COMP2FOD applied to a PUMA problem Π generates the FOD problem $\Pi_{\text{COMP2FOD}} = \langle F', X, A', I', G'\rangle$, where each component is defined below.

Facts and Numeric Variables The set of facts F is expanded as $F' = F \cup F_{\mathsf{z}} \cup \{\mathsf{plan}\}$, where $F_{\mathsf{z}} = \{\mathsf{z}^i_p \mid p \in F, i \in \{1, \ldots, N\}\}$ consists of N copies of the original facts in F. Fact z^i_p indicates that the original fact p must be set to true when the i-th task is opened, and plan is a fact used to alternate between the planning phase and the recovery of open tasks.

The set of (bounded) numeric variables is defined as $X = \{\mathsf{open}\} \cup X^{curr}_{\mathsf{k}} \cup X_{\mathsf{k}}$, where $X^{curr}_{\mathsf{k}} = \{\mathsf{k}^{curr}_a \mid a \in A\}$ and $X_{\mathsf{k}} = \{\mathsf{k}^i_a \mid a \in A, i \in \{1, \ldots, N\}\}$. The variable open memorizes the number of tasks currently open and takes values in the domain $\{0, \ldots, N\}$. For each action a, the variable k^{curr}_a memorizes the index of the effect selected for a in the current task and takes values in $\{0, 1, \ldots, m_a\}$, where 0 indicates that the action a has not yet been executed in the current task, while the values from 1 to m_a identify which among the m_a possible sets of effects has been applied. Similarly, k^i_a has the same meaning and domain as k^{curr}_a but refers to the i-th task in the open-task stack. Given an action a, the variables k^{curr}_a or k^i_a are referred to as the *effect identifier* of a.

Variables $F \cup X^{curr}_{\mathsf{k}}$ determine the state and the effect identifiers of the uncertain actions executed in the current task. Similarly, given $i \in \{1, \ldots, N\}$, the i-th task in the open-task stack is characterized by a set of facts $(\{z^i_p \mid p \in F\})$ representing a state, and the effect identifiers $(\{\mathsf{k}^i_a \mid a \in A_{\text{U}}\})$ of the uncertain actions executed in the task where the branch that generated task i occurred.

Actions The set of novel actions generated by the compilation is

$$
\begin{aligned}
A' &= A^{\oplus}_{\text{U}} \cup A^{\varnothing}_{\text{U}} \cup A'_{\text{C}} \cup \{\mathit{close}\} \cup A_{\mathit{res}}, \text{where:}\\
A'_{\text{C}} &= \{a' = \langle \text{pre}(a) \cup \{\mathsf{plan}\}, \text{eff}(a)\rangle \mid a \in A_{\text{C}}\},\\
A^{\oplus}_{\text{U}} &= \{a^{\oplus}_i \mid a \in A_{\text{U}}, i \in \{0, \ldots, N - (m_a - 1)\}\},\\
A^{\varnothing}_{\text{U}} &= \{a^{\varnothing} \mid a \in A_{\text{U}}\},\\
A_{\mathit{res}} &= \{\mathit{resume}^i \mid i \in \{1, \ldots, N\}\}.
\end{aligned}
$$

The set of certain actions of the PUMA problem is compiled into A'_{C} by simply augmenting their preconditions with plan. Each uncertain action a is compiled into two variants, giving rise to two sets: $A^{\oplus}_{\text{U}} \cup A^{\varnothing}_{\text{U}}$. In particular, the action $a^{\oplus}_i$, called a *branching action*, is used to model the situation in which action a is executed for the first time on a given trajectory when the number of open tasks is equal to i, and has to establish all (sub)tasks that have to be handled after its execution. We have $N - (m_a - 1)$ copies of this action, each of which can be used for a different number of currently open tasks. Conversely, the action $a^{\varnothing}$, called a *persist action*, is used to impose that any successive execution of the action a results in the same effects applied when a was executed for the first time. Formally:

$$
\begin{aligned}
\text{pre}(a^{\varnothing}) &= \text{pre}(a) \cup \{\mathsf{plan}, (\mathsf{k}^{curr}_a > 0)\},\\
\text{eff}(a^{\varnothing}) &= \{(\mathsf{k}^{curr}_a = j) \triangleright \text{eff}_j(a) \mid j \in \{1, \ldots, m_a\}\}.
\end{aligned}
$$

Action $a^{\varnothing}$ can be executed when (i) the precondition of a is satisfied, (ii) the planning phase is ongoing, and (iii) the effect identifier of a for the current task is greater than 0, meaning that a has already been executed in the current task.

Action $a^{\varnothing}$ has a conditional effect for each of the alternative effects of a, which, if fired by condition $(\mathsf{k}_a^{curr} = j)$, results in the same effects $\mathrm{eff}_j(a)$ as those previously applied for the current task. The conditions of these effects are mutually exclusive, meaning that exactly one of them can occur.

Let $a \in A$ and let $i \in \{0, \ldots, N - (m_a - 1)\}$ be the current number of open tasks, the corresponding branching action $a_i^{\oplus}$ is defined:

$$\mathrm{pre}(a_i^{\oplus}) = \mathrm{pre}(a) \cup \{\mathsf{plan}, (\mathsf{k}_a^{curr} = 0), (\mathsf{open} = i)\},$$
$$\mathrm{eff}(a_i^{\oplus}) = \mathrm{eff}_1(a) \cup \{(\mathsf{k}_a^{curr} := 1), (\mathsf{open} \mathrel{+}= m_a - 1)\} \cup \textsc{Save}(a, i).$$

In particular, $a_i^{\oplus}$ can be executed when (i) the precondition of a is satisfied, (ii) the planning phase is ongoing, (iii) the action a has never been executed in the current task (condition $(\mathsf{k}_a^{curr} = 0)$), and (iv) the number of open tasks is equal to i. Executing this action (i) updates the variables in F according to one of the non-deterministic effects of a, say $\mathrm{eff}_1(a)$, (ii) tracks the effect of a to 1 in the current task (effect $(\mathsf{k}_a^{curr} := 1)$) so that, if a is executed again in the current task, the next executions of a would result in the same effect, (iii) opens $m_a - 1$ alternative tasks by increasing the number of open tasks, and (iv) stores the information related to the $m_a - 1$ open tasks using $\textsc{Save}(a, i)$. The set of effects prescribed by $\textsc{Save}(a, i)$ is the core of the compilation and is defined as follows:

$$\bigcup_{j \in \{2,\ldots,m_a\}} \{(\mathsf{k}_a^{i+j-1} := j)\} \cup \quad (1)$$
$$\{(\mathsf{k}_{a'}^{i+j-1} := \mathsf{k}_{a'}^{curr}) \mid a' \in A_{\mathrm{U}} \setminus \{a\}\} \cup \quad (2)$$
$$\{\mathsf{z}_p^{i+j-1} \mid p \in \mathrm{eff}_j^{+}(a)\} \cup \quad (3)$$
$$\{\neg\mathsf{z}_p^{i+j-1} \mid p \in \mathrm{eff}_j^{-}(a)\} \cup \quad (4)$$
$$\{p \rhd \mathsf{z}_p^{i+j-1}, \neg p \rhd \neg\mathsf{z}_p^{i+j-1} \mid p \in F \setminus \mathcal{A}tm(\mathrm{eff}_j(a))\} \quad (5)$$

where $\mathcal{A}tm(L)$ indicates the facts in the set of literals L. $\textsc{Save}(a, i)$ is applied to each of the $m_a - 1$ tasks opened by the execution of a. Given that the number of tasks currently opened is i, the index of these new tasks is in the range $\ell = i+j-1$ for every $j \in \{2, \ldots, m_a\}$, that is $\ell \in \{i+1, \ldots, i+m_a-1\}$. Intuitively,

- Effects in Formula 1 assign to the effect identifier k_a^{ℓ} a value j, indicating which possible outcome of action a will be applied for task ℓ.
- Effects in Formula 2 ensure that, for every action $a' \neq a$, the current value of its effect identifier $\mathsf{k}_{a'}^{curr}$ is propagated to $\mathsf{k}_{a'}^{\ell}$, so that the possible next executions of a' in task ℓ will result in the same effect.
- Effects in Formula 3-4 apply the additive and deleting effects in $\mathrm{eff}_j(a)$ by setting z_p^{ℓ} for $p \in \mathrm{eff}_j^{+}(a)$ and $\neg\mathsf{z}_p^{\ell}$ for $p \in \mathrm{eff}_j^{-}(a)$.
- Finally, effects in Formula 5 propagate the truth value of all unaffected facts $p \in F \setminus \mathcal{A}tm(\mathrm{eff}_j(a))$ through conditional effects to preserve their value for an open task ℓ. Specifically, for each such p, the effect includes both $p \rhd \mathsf{z}_p^{\ell}$ and $\neg p \rhd \neg\mathsf{z}_p^{\ell}$.

Once the original goal G of Π has been achieved for the current task, it is possible to execute the action $close = \langle G \cup \{\mathsf{plan}\}, \{\neg\mathsf{plan}\}\rangle$. This action makes fact plan false, preventing the execution of any action in $A_{\mathrm{U}}^{\oplus} \cup A_{\mathrm{U}}^{\varnothing} \cup A'_{\mathrm{C}}$, and forcing the opening of the last generated task, indexed by the variable open, via the action $resume^i \in A_{res}$, where i is the number of tasks currently open. Let $i \in \{1, \ldots, N\}$, $resume^i$ is formally defined as:

$$\mathrm{pre}(resume^i) = \{\neg\mathsf{plan}, (\mathsf{open} = i)\},$$
$$\mathrm{eff}(resume^i) = \{\mathsf{plan}, (\mathsf{open} \mathrel{-}= 1)\} \cup \{\mathsf{z}_p^i \rhd p, \neg\mathsf{z}_p^i \rhd \neg p \mid p \in F\} \cup \{(\mathsf{k}_a^{curr} := \mathsf{k}_a^i) \mid a \in A_{\mathrm{U}}\}.$$

The action $resume^i$ can be executed when (i) the system is not in the planning phase, i.e., immediately after the closure of a task, and (ii) the number of open tasks is equal to i. As a result, the number of open tasks is reduced by one, and the planning phase is reactivated. The assignment of variables stored for the i-th open task, denoted by o, is retrieved from F_{z} and copied into the variables in F, thus restoring the state to the point at which it resulted from the application of the uncertain action that opened o. Similarly, the values of the effect identifiers stored for the i-th open task are retrieved from X_{k} and copied into the variables in X_{k}^{curr}. From this moment onward, the variables reserved for the i-th task can be safely reused and overwritten if a new task is opened, while the variables in F and k_a^{curr} contain all the information necessary to resume planning from the point where the branching for o occurred.

Initial State The initial state I is extended as follows:

$$I' = I \cup \{\mathsf{plan}, (\mathsf{open} := 0)\} \cup \{(\mathsf{k} := 0) \mid \mathsf{k} \in X_{\mathsf{k}}^{curr} \cup X_{\mathsf{k}}\}.$$

I' specifies that the planning problem begins with the planning phase and that no task has yet been opened. Moreover, all action identifiers in $X_{\mathsf{k}}^{curr} \cup X_{\mathsf{k}}$ are initialized to 0, because no action has yet been applied within the current task, as well as any other task.

Goal The compiled problem is solved whenever the current task is closed, that is, the planning phase is over, and there are no remaining tasks in the open-task stack to be closed, meaning that the value of open is 0. Formally, $G' = \{\neg\mathsf{plan}, (\mathsf{open} = 0)\}$.

Theoretical Properties

We now discuss the properties of the proposed compilation. In particular, we show that the resulting problem can be polynomially reduced to a classical planning instance. We then establish a relation between action strategies for PUMA problem and plans in the compiled classical planning problem, thereby proving the soundness and completeness of the compilation. This result shows membership of PUMA in PSPACE.

Theorem 1. *Given a* PUMA *problem* Π*, the size of* $\Pi_{\textsc{Comp2fod}}$ *is polynomial w.r.t. the size of* Π.

Proof Sketch. Just observe that we need a number of facts that is polynomial in the number of the original facts. We

indeed need a copy of each fact for every possible task that can be opened simultaneously, and they can be at most $N = \sum_{a \in A}(m_a - 1)$. Then we add a polynomial number of numeric variables, all with a bounded size. They can all be compiled into facts with a logarithmic encoding (Gigante and Scala 2023). The number of actions to be added is also polynomial in the number of input actions, and so is the number of conditional effects we produce. □

Theorem 2. *A* PUMA *problem* Π *is solvable iff* Π_{COMP2FOD} *is solvable.*

Proof Sketch. ($\Rightarrow$) First, we prove that if there exists a strategy π for Π, then there exists a plan for Π_{COMP2FOD}. It is easy to see that if the strategy exists, then there exists a tree where the nodes represent sequences of (perceived) deterministic actions, the root represents the empty sequence, each edge outgoing from a node η representing $\langle \hat{a}_1, \ldots, \hat{a}_i \rangle$ is labelled with $\pi(\hat{a}_1, \ldots, \hat{a}_i)$, each child of η represents $\langle \hat{a}_1, \ldots, \hat{a}_i, \hat{a}_x \rangle$ where $\hat{a}_x \in \text{DTMN}(\pi(\hat{a}_1, \ldots, \hat{a}_i))$, and each leaf of the tree represents $\langle \hat{a}_1, \ldots, \hat{a}_n \rangle$ such that $G \subseteq I[\langle \hat{a}_1, \ldots, \hat{a}_n \rangle]$. Given such a tree, we can construct a plan for Π_{COMP2FOD} as follows. Consider a leaf node η of the tree representing $\langle \hat{a}_1, \ldots, \hat{a}_n \rangle$ and such that each of these (deterministic) actions uses the first effect eff_1 of an (uncertain) action in A. We can show that there exists a plan such that its first n actions in A'_{C}, $A^{\oplus}_{\text{U}}$, and $A^{\varnothing}_{\text{U}}$ are derived from the actions in A_{C} and A_{U} labelling the edges that form the n-path ρ from the root of the tree to η, its $(n+1)$-th action is *close*, and that these actions are executable because the dynamics of the compiled actions is the same as for the original actions from which they derive for the facts in F, and because of the construction of the actions in Π_{COMP2FOD} for the facts that are not part of F. Assume that $\pi(\langle \hat{a}_1, \ldots, \hat{a}_i \rangle) \in A_{\text{C}}$ for $1 \leq i \leq n$. Then, the first n actions are in A'_{C}, i.e., they open no task, and the plan formed by these actions plus *close* is a solution as $G' = \neg\mathsf{plan} \wedge \langle \mathsf{open} = 0 \rangle$ holds in the state resulting from the execution of this plan. Indeed, open is 0 in I', the actions in A'_{C} that are in the plan do not change open value, and *close* sets plan to false.

Let j be the smallest natural number less than or equal to n such that $\pi(\langle \hat{a}_1, \ldots, \hat{a}_j \rangle) = a \in A_{\text{U}}$. Assume now that a is the only uncertain action labelling the edges that form path ρ, and $m_a = 2$. So, in the plan, there is only an action in $a^{\oplus}_0 \in A^{\oplus}_{\text{U}}$, which opens 1 new task o to solve. In this case, the $(n+1)$-th action in the plan after the *close* action is $resume^1$. This action is executable because the *close* action and the j-th action $a^{\oplus}_0$ in the plan, respectively, achieve the preconditions $\neg\mathsf{plan}$ and $(\mathsf{open} = 1)$ of $resume^1$. Then, it is easy to see that two edges are outgoing from node $\eta = \langle \hat{a}_1, \ldots, \hat{a}_j \rangle$ in the tree representing the solution strategy. Consider the path ρ' from η to a leaf of the tree representing $\langle \hat{a}_1, \ldots, \hat{a}_m \rangle$ with $m > j$ and such that $\hat{a}_{j+1}$ uses the second effect eff_2 of a. For the same reasons mentioned before, for each action a along this path, we can show that there is an action in A_{C}, $A^{\oplus}_{\text{U}}$, or $A^{\varnothing}_{\text{U}}$ in the plan, followed by another instance of *close* that closes task o, all these actions are executable and, if there is no uncertain action along ρ', the plan achieves the problem goals. In general, we can show that, for each open task, we can construct an executable portion of the plan that closes it; once no open task remains, the plan solves Π_{COMP2FOD}. Note that the number of tasks that can be open together is $N = \Sigma_{a \in A}(m_a - 1)$, and the compilation has enough facts and variables in F_{z} and X_{k} to save all the information required to restore each of these tasks. Moreover, the number of tasks that can be open is finite and equal to the number of paths from the root to any leaf of the tree representing the solution strategy. Thereby, a plan for Π_{COMP2FOD} is finite and can be constructed from the tree.

($\Leftarrow$) Assume now that there exists a plan for Π_{COMP2FOD}. We show that there must exist a solution strategy π for Π. First, we consider the case where the plan is $\langle a'_1, \ldots, a'_n, close \rangle$, where $a'_i \in A_{\text{C}}$ for $1 \leq i \leq n$, that is, they do not open tasks. We can define a solution strategy for Π as follows. $\pi(\langle\rangle) = a_1$, $\pi(\langle \hat{a}_1, \ldots, \hat{a}_i \rangle) = a_{i+1}$ for $1 \leq i < n$, $\pi(\langle \hat{a}_1, \ldots, \hat{a}_n \rangle) = \epsilon$, where a_i is the action in A from which the compilation scheme derives a'_i, and $\hat{a}_i$ is the action in $\text{DTMN}(a_i)$ with the same effects as $\text{eff}(a'_i) \cap F$. We can show that such a strategy is a solution, as the preconditions of the compiled actions include the preconditions of the original actions, and the dynamics of the actions is the same for the facts in F.

Now consider the case where the plan contains actions in $A^{\oplus}_{\text{U}}$. Assume now that the plan contains only an action derived from an uncertain action with two alternative effects. I.e., let $\langle a'_1, \ldots, (a_j)^{\oplus}_0, \ldots, a'_n, close, resume^1, a'_{n+1}, \ldots, a'_{n+m}, close \rangle$ be the plan that solves Π_{COMP2FOD}, where $a'_i \in A_{\text{C}}$ for $1 \leq i \leq n+m$, $i \neq j$, and $(a_j)^{\oplus}_0 \in A^{\oplus}_{\text{U}}$, that is, the j-th action is the only one that opens a task. From this plan, we can derive a solution strategy as follows. The first n actions in the plan induce the same strategy as in the previous case (considering that a_j is the PUMA action from which the compilation derives $(a_j)^{\oplus}_0$). Then $\pi(\langle \hat{a}_1, \ldots, \hat{a}^1_j \rangle) = a_{n+1}$, $\pi(\langle \hat{a}_1, \ldots, \hat{a}^1_j, \hat{a}_{n+1} \ldots \hat{a}_k \rangle) = a_{k+1}$ for $n < k < n+m$, and $\pi(\langle \hat{a}_1, \ldots, \hat{a}^1_j, \hat{a}_{n+1} \ldots \hat{a}_{n+m} \rangle) = \epsilon$, where $\hat{a}^1_j$ is the action in $\text{DTMN}(a_j)$ with the same effects as those in the ℓ-th set of effects of a_j and ℓ is the value assigned to k^{curr}_a by action $resume^1$; a_k is again the action in A from which the compilation scheme derives a'_k, and $\hat{a}_k$ is the action in $\text{DTMN}(a_k)$ with the same effects as $\text{eff}(a'_k) \cap F$. We can show that such a strategy is a solution for the same reasons mentioned before. This strategy can induce at most two traces; in other words, the strategy's tree has two paths from the root to a leaf. As shown, both these paths lead to executable actions and reach the goals. In general, we can use the same procedure to derive a strategy from a plan that opens more than one task, and use the same arguments to show that such a strategy is a solution of Π. Indeed, assume by contradiction that there exists an action trace induced by the strategy that is not executable or does not reach the problem goal. This would entail that the plan from which the strategy is derived opens a task that is not closed, that is, the plan execution results in a state where open is greater than zero and hence does not meet a goal, the actions executed for a task are not executable, or they do not reach the problem goal, that is, do not reach the precondition of

the *close* action closing the task. But this is not possible as the plan is a solution for Π_{COMP2FOD}. □

Corollary 3. PUMA *is* PSPACE-complete.

Proof. From Theorem 2, the compilation shows that PUMA is in PSPACE. The hardness follows from the observation that PUMA generalizes classical planning, which is PSPACE-complete (Bylander 1994). □

Related Work

In the broad literature on planning under uncertainty, a variety of models and methods have been proposed (Littman, Goldsmith, and Mundhenk 1998). A particularly relevant formulation is FOND planning (Kuter et al. 2008; Fu et al. 2011; Muise, McIlraith, and Beck 2012; Ramírez and Sardiña 2014; Geffner and Geffner 2018), which assumes complete observability and no probabilistic information about action outcomes. Our work preserves these assumptions but adds a key distinction. Unlike the classical adversarial FOND setting, where the environment may steer outcomes against the agent, we assume that any admissible effect may occur; once observed, however, it becomes known for the remainder of the plan. The problem, therefore, reduces to managing a finite set of contingencies, each corresponding to a possible realization of the action model.

When attention is restricted to bounded forms of uncertainty, the closest precedent is fault-tolerant planning (FTP) (Jensen, Veloso, and Bryant 2004; Domshlak 2013). In FTP, each action has a primary and a faulty effect, and the goal is to reach the target state despite a bounded number of faults. PUMA diverges from FTP in three key aspects: it treats all effects symmetrically, without distinguishing primary and faulty outcomes; it bounds the number of unexpected outcomes per action rather than imposing a global fault limit; and once an effect is observed, subsequent executions of that action yield the same outcome. Thus, in PUMA, the true effect of each action is revealed upon first execution, turning hidden non-determinism into determinism for the remainder of the plan.

Another relevant line of work is resilient planning (Aineto et al. 2023; Rovetta et al. 2025), which computes k-resilient plans that ensure goal achievement as long as no more than k action failures occur. Unlike resilient planning, PUMA does not rely on a failure model: the number of alternative outcomes can exceed two and differ across actions.

Related formulations include contingent planning and partially observable deterministic planning (POD) (Rintanen 2004; Palacios and Geffner 2009; Bonet and Geffner 2011). These approaches manage uncertainty through explicit sensing actions that reveal information about the world state. Although PUMA assumes full observability, it can be interpreted as a POD task in which the first execution of any uncertain action implicitly provides an observation of its own deterministic effect, thus resolving the uncertainty about that action for all future executions.

Finally, our work is related to action model learning (Juba, Le, and Stern 2021; Lamanna et al. 2021; Le, Juba, and Stern 2024; Aineto and Scala 2024; Lamanna et al. 2025). In particular, while action model learning may yield multiple action model hypotheses, we address the problem of planning given a finite set of possible models.

Experimental Evaluation

The goal of our evaluation is to assess the effectiveness and scalability of the two compilation strategies introduced for PUMA: one targeting deterministic planning (COMP2FOD) and the other targeting FOND planning (COMP2FOND). We are interested in how these approaches scale with increasing problem size and degree of uncertainty when combined with domain-independent planners.

Each run in our experiments consists of a compilation phase followed by a planning phase, both limited to a 3,600-second timeout. The classical planning problems generated by COMP2FOD were solved using Fast Downward (Helmert 2006) with the LAMA configuration (Richter and Westphal 2010) set to return the first solution found (denoted as LAMA-First). The implementation of COMP2FOD used in our experiments was developed within the Unified Planning framework (Micheli et al. 2025), preserves conditional effects, and compiles away numeric features by applying the so-called one-hot encoding compilation (Bonassi, Percassi, and Scala 2025), adapted to support direct assignments. The FOND planning problems generated by COMP2FOND were solved using PR2 (Muise, McIlraith, and Beck 2024), a state-of-the-art FOND planner.

We initially considered using existing domains and problems developed for FOND. Still, we realized that these problems are unsolvable under the semantics of PUMA, as they admit only strong cyclic solutions. Thereby, for our study, we developed three new benchmark domains, *TreasureIsland*, *LostAgent*, and a variant of the well-known *Blocksworld* domain. In *TreasureIsland*, a sailor navigates unexplored islands via predefined routes; a treasure coffer is located on one island, while the key needed to open it is hidden on one of the islands. For this domain, we define two variants. In the first variant, *Treasure-Key*, the uncertainty lies in the initial key location, while all the routes between islands are known. In the second variant, *Treasure-Routes*, the key location is known, but some navigation actions are uncertain and may lead to unexpected destinations. The key distinction between the two lies in the number of interpretations of the action model that the agent has to consider during the strategy generation: *Treasure-Key* induces a linear number of models to consider w.r.t. the number of uncertain actions, whereas *Treasure-Routes* leads to an exponential growth due to combinations of uncertain route outcomes. In the *LostAgent* domain, an agent operates in a grid environment where movement actions have uncertain effects, potentially causing transitions to any adjacent cell. The agent can rotate deterministically and has limited energy, which can be replenished only under specific conditions. The number of model interpretations to consider for this domain grows linearly with the number of possible outcomes of the uncertain movement action. Finally, the *Blocksworld* domain adds hidden block properties, such as slipperiness, that affect the outcome of the unstack action. Since these properties are

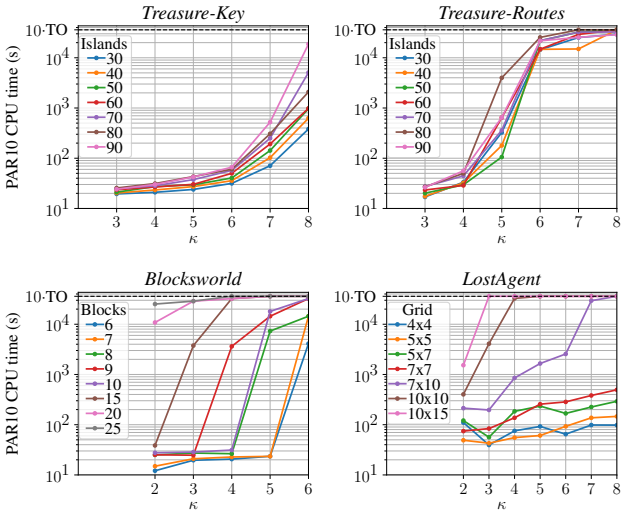

Figure 1: Average PAR10 CPU time of COMP2FOD plus LAMA-First, grouped by problem size and uncertainty degree κ; TO means timeout.

initially unknown and become observable only through the action execution, the number of interpretations of the model to consider grows exponentially with the number of blocks that exhibit such a property.

Each domain was evaluated using a two-dimensional benchmark that varies in problem size and degree of uncertainty κ, where problem size reflects structural complexity (e.g., number of islands, blocks, or grid size), and κ captures either the number of uncertain actions or the number of possible outcomes of an uncertain action. For *TreasureIsland*, κ (ranging from 3 to 8) denotes the number of uncertain actions affecting the key location or route outcomes. For *Blocksworld*, κ (from 2 to 6) is the number of blocks with hidden slipperiness, which is also the number of uncertain unstack actions. For *LostAgent*, κ (from 2 to 8) represents the number of possible outcomes of the movement action. For each combination of problem size and κ, we generated 10 random instances.

Deterministic Planning Compilation We first analyze the performance of the compilation COMP2FOD in combination with LAMA-First. Figure 1 shows the average CPU time (PAR10) as both problem size and degree of uncertainty κ increase. The results show that domains with linear growth of the model interpretations (*Treasure-Key*, *LostAgent*) scale more gracefully, while domains with exponential growth (*Treasure-Routes*, *Blocksworld*) lead to rapidly increasing CPU times. Among the latter, *Blocksworld* appears notably harder, as solving times increase significantly even for small instances with only a few slippery blocks. A possible explanation is that, in this domain, the compilation tends to obscure goal structure, thereby misleading the heuristics and hindering the planner's ability to identify promising paths to a solution.

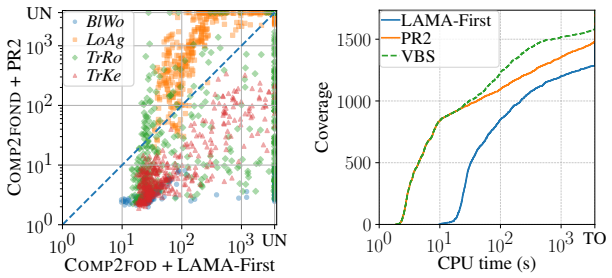

Figure 2: Left: CPU time per instance of COMP2FOD plus LAMA-First vs. COMP2FOND plus PR2; UN means unsolved, and domain names are abbreviated. Right: coverage over an increasing CPU time; VBS means Virtual Best Solver and TO means timeout.

Deterministic versus FOND Planning Pairwise comparison of the two approaches, COMP2FOD with LAMA-First and COMP2FOND with PR2, is shown in the left plot of Figure 2. The scatter plot shows, for each instance, the CPU time required by both approaches over the whole benchmark. The results indicate a clear dominance of the FOND-based approach in *Blocksworld* and *Treasure-Key*, where PR2 consistently outperforms the deterministic approach. In *Treasure-Routes*, the performance is mixed, with no approach consistently outperforming the other. In contrast, in *LostAgent*, the classical compilation with LAMA-First achieves better results overall. The plot on the right of Figure 2 shows the coverage over the CPU time limit for each planner and the Virtual Best Solver (VBS), which hypothetically selects the fastest approach for each instance. The VBS outperforms both individual planners, confirming the complementarity of the compilations. Overall, the VBS solves approximately 20% more instances than LAMA-First and about 6% more than PR2 within the time limit, quantifying the degree of complementarity between the two approaches.

Conclusions

This paper presents a new form of planning with uncertainty, PUMA, designed to capture scenarios in which actions initially have uncertain effects that become certain after their first execution. This formulation naturally captures domains in which uncertainty arises from incomplete knowledge of action models, rather than from ongoing stochasticity. We presented two polynomial-time compilations of PUMA problems: to FOND and deterministic planning. Of particular interest is the latter, as it provides a constructive proof that PUMA is a PSPACE-complete problem. The experimental results, obtained using off-the-shelf planners, indicate the complementarity of the approaches, although the FOND-based compilation is, as expected, more effective.

As future work, we plan to explore alternative uncertainty formulations, including a POD-based one to identify a PSPACE-complete fragment. We also aim to formally relate PUMA to bounded alternating Turing machines to clarify its theoretical properties and expressive power.

Acknowledgments

Francesco Percassi was supported by a UKRI Future Leaders Fellowship [grant number MR/Z00005X/1]. Alessandro Saetti and Enrico Scala were supported by the AI4WATER project within the European Union's PRIMA Programme.

References

Aineto, D.; Gaudenzi, A.; Gerevini, A.; Rovetta, A.; Scala, E.; and Serina, I. 2023. Action-Failure Resilient Planning. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 44–51.

Aineto, D.; and Scala, E. 2024. Action Model Learning with Guarantees. In *KR*, 801–811.

Bonassi, L.; Espasa, J.; Percassi, F.; and Scala, E. 2025. Conditional Effects in Numeric Planning Reloaded. In *ECAI*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, 4929–4936.

Bonassi, L.; Percassi, F.; and Scala, E. 2025. Towards Practical Classical Planning Compilations of Numeric Planning. In *AAAI*, 26472–26480.

Bonet, B. 2010. Conformant plans and beyond: Principles and complexity. *Artif. Intell.*, 174(3-4): 245–269.

Bonet, B.; and Geffner, H. 2011. Planning under Partial Observability by Classical Replanning: Theory and Experiments. In *IJCAI*, 1936–1941.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artif. Intell.*, 69(1-2): 165–204.

Chrpa, L.; and Karpas, E. 2024. On Verifying Linear Execution Strategies in Planning Against Nature. In *ICAPS*, 86–94.

Chrpa, L.; and Karpas, E. 2025. On Generating Robust Plans and Linear Execution Strategies in Planning Against Nature. In *ICAPS*, volume 35, 169–177.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*, 147(1-2): 35–84.

Domshlak, C. 2013. Fault Tolerant Planning: Complexity and Compilation. In *ICAPS*.

Fu, J.; Ng, V.; Bastani, F. B.; and Yen, I. 2011. Simple and Fast Strong Cyclic Planning for Fully-Observable Nondeterministic Planning Problems. In *IJCAI*, 1949–1954.

Geffner, T.; and Geffner, H. 2018. Compact Policies for Fully Observable Non-Deterministic Planning as SAT. In *ICAPS*, 88–96.

Gerevini, A. E.; Percassi, F.; and Scala, E. 2024. An Effective Polynomial Technique for Compiling Conditional Effects Away. In *AAAI*, 20104–20112.

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

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.

Jensen, R. M.; Veloso, M. M.; and Bryant, R. E. 2004. Fault Tolerant Planning: Toward Probabilistic Uncertainty Models in Symbolic Non-Deterministic Planning. In *ICAPS*, 335–344.

Juba, B.; Le, H. S.; and Stern, R. 2021. Safe Learning of Lifted Action Models. In *KR*, 379–389.

Kuter, U.; Nau, D. S.; Reisner, E.; and Goldman, R. P. 2008. Using Classical Planners to Solve Nondeterministic Planning Problems. In *ICAPS*, 190–197.

Lamanna, L.; Saetti, A.; Serafini, L.; Gerevini, A.; and Traverso, P. 2021. Online Learning of Action Models for PDDL Planning. In *IJCAI*, 4112–4118.

Lamanna, L.; Serafini, L.; Saetti, A.; Gerevini, A. E.; and Traverso, P. 2025. Lifted action models learning from partial traces. *Artif. Intell.*, 339: 104256.

Le, H. S.; Juba, B.; and Stern, R. 2024. Learning Safe Action Models with Partial Observability. In *AAAI*, 20159–20167.

Littman, M. L.; Goldsmith, J.; and Mundhenk, M. 1998. The Computational Complexity of Probabilistic Planning. *J. Artif. Intell. Res.*, 9: 1–36.

Micheli, A.; Bit-Monnot, A.; Röger, G.; Scala, E.; Valentini, A.; Framba, L.; Rovetta, A.; Trapasso, A.; Bonassi, L.; Gerevini, A. E.; Iocchi, L.; Ingrand, F.; Köckemann, U.; Patrizi, F.; Saetti, A.; Serina, I.; and Stock, S. 2025. Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29: 102012.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2024. PRP Rebooted: Advancing the State of the Art in FOND Planning. In *AAAI*, 20212–20221.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *ICAPS*.

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

Palacios, H.; and Geffner, H. 2009. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *J. Artif. Intell. Res.*, 35: 623–675.

Ramírez, M.; and Sardiña, S. 2014. Directed Fixed-Point Regression-Based Planning for Non-Deterministic Domains. In *ICAPS*.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.*, 39: 127–177.

Rintanen, J. 2004. Complexity of Planning with Partial Observability. In *ICAPS*, 345–354.

Rovetta, A.; Aineto, D.; Gerevini, A. E.; Scala, E.; and Serina, I. 2025. Improving Resilient Planning Through Landmarks and Regressed State Formulas. In *ECAI*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, 4645–4652.

Smith, D. E.; and Weld, D. S. 1998. Conformant Graphplan. In *AAAI/IAAI*, 889–896.

Walsh, T. J.; and Littman, M. L. 2008. Efficient Learning of Action Schemas and Web-Service Descriptions. In *AAAI*, 714–719.