Domain Concretization from Examples: Addressing Missing Domain Knowledge via Robust Planning

Akshay Sharma, Piyush Rajesh Medikeri and Yu Zhang

***Abstract*— The assumption of complete domain knowledge is unwarranted for robot planning and decision-making in the real world. Incompleteness in domain knowledge may come from design flaws or arise from domain ramifications or qualifications. In such cases, traditional planning methods can produce highly undesirable behaviors. Addressing the planning problem under incomplete domain knowledge is challenging since the agent has no clue about what information is missing. This is a type of unknown unknowns, which differs significantly from partial observability, a type of known unknowns. In this work, we assume that the missing information is encoded in a set of examples or teacher demonstrations. We formulate the problem of *domain concretization* with these examples as an inverse problem to domain abstraction. Given a domain model provided initially, when the model does not conform with the teacher demonstrations, our method searches for a candidate model set that refines the initial model under a minimalistic and deterministic model assumption. For new problems, it generates a robust plan with the maximum probability of success under the set of candidate models. Together with a standard search formulation in the model-space, we propose a heuristic-based search method and also an online version of it to reduce the search time. We evaluated our approach with several International Planning Competition (IPC) domains and a simulated robotics domain where incompleteness was introduced by removing domain features from the complete models. Results show that our methods increase the success rate of planning without significantly impacting the plan cost.**

I. INTRODUCTION

Most planning agents rely on the assumption of complete domain knowledge for decision-making. In the case when the domain knowledge is incomplete, these agents will continue with decision-making assuming the knowledge is complete, often resulting in undesirable behaviors. For example, missing state features implies that the agent would perceive different states as the same state if they differ only in those features. In such cases, traditional planning agents can generate the same plan under very different scenarios. For similar reasons, standard learning methods, such as reinforcement learning (RL) [1] and inverse reinforcement learning (IRL) [2], [3], and inference methods, such as intention recognition [4], [5], can be easily misled when complete domain knowledge is assumed.

Consider a packing domain depicted in Fig. 1. The robot is tasked to pack items into different boxes. The items are made of different materials. Under the initial model provided by the domain designer (not the user), the robot would simply stack items into a box until it is full and continue to fill the next box. However, items made from certain materials could be fragile (colored yellow in Fig. 1). A fragile item would break when stacked on. There are various reasons why such domain knowledge may be missing in the initial model. For example, the designer may be unaware that items made from certain materials are fragile, or have simply ignored the problem. A robot that assumes complete domain knowledge would inevitably fail in action with its users. Our work aims to allow the robot to use teacher (user) demonstrations to infer missing information to refine its model after the user detects problems. It has a variety of applications, such as automated manufacturing (similar to Fig. 1), where the domain model must be refined over time to adapt to unmodeled complexities in the real world.

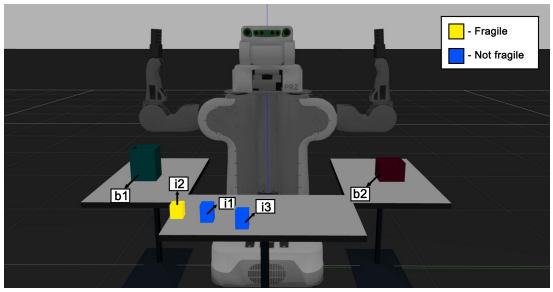

Fig. 1: Scenario for packing items (i_1-i_3) into boxes (b_1-b_2).

In this paper, we introduce the concept of *domain concretization* to address incomplete domain knowledge. The planning problem under incomplete domain knowledge is more challenging than planning under partial observability. Since the agent does not know what information is missing, it is a type of *unknown unknowns* [6]. To address such a challenging setting, we assume that the missing knowledge is encoded by a set of teacher demonstrations for a set of planning problems (referred to as training cases). Given the domain model provided initially, when the model does not conform with the teacher demonstrations, our method searches for a candidate model set that refines the initial model to conform with the teacher demonstrations. For new planning problems (referred to as test cases), we can generate a robust plan to satisfy as many models as possible in the candidate model set. In our motivating scenario in Fig. 1, a teacher demonstration may switch to a new box after packing a yellow item even though the current box is not full. Our method would search for ways to refine the initial model to *explain* this demonstration. For example, it can impose a new precondition on the stacking action requiring the

This work was supported by NSF grants 1844524, 2047186, and AFOSR grant FA9550-18-1-0067.

Akshay Sharma, Piyush Rajesh Medikeri and Yu Zhang are with the School of Computing and Augmented Intelligence (SCAI), Arizona State University, Tempe, AZ. {ashar204, pmediker, yu.Zhang.442} @asu.edu.

item to be stacked on to be non-fragile, and simultaneously expect a subset of items (i.e., the blue items) to satisfy the fulfilling property (i.e., being non-fragile). Note that we may arbitrarily interpret the additional information and the robot merely needs to understand that the property is a prerequisite for the action. Furthermore, there may be multiple ways to explain the demonstrations (e.g., all items are fragile vs. only the yellow items are fragile), resulting in different candidate models. A robust plan should work under as many such models as possible.

Our solution is first formulated as a search problem in the model space. To expedite the search, we then present a heuristic-based method by restricting the set of possible models considered. Additionally, we present an online version that is more practical when the teachers demonstrations are provided incrementally. It also has a computational advantage by using only one demonstration in each iteration and maintaining a small set of candidate models for the next iteration. To address a test case, our method generates a robust plan that achieves the goal of the planning problem associated with the test case under the maximum number of candidate models. This is achieved by transforming the planning problem into a Conformant Probabilistic Planning problem [7]. We evaluated our method on various IPC domains and a simulated robotics domain where incompleteness was introduced by removing certain features from the complete domain model. Results show that the robust plans generated by our method can increase the success rate of planning without significantly impacting the plan cost (i.e., the sum of its action costs). Additional technical discussions and complete proofs can be found in the full version [8].

Our contribution in this work is three-fold: 1) introducing the problem of domain concretization from examples, which addresses a type of unknown unknowns that are little studied in robot task planning; 2) proposing model-search and heuristics based methods, combined with probabilistic conformant planning, to address the problem with formal analyses; 3) evaluating the proposed methods in IPC domains and a simulated robotics domain in an automated manufacturing setting to demonstrate the potential in real-world domains.

II. RELATED WORK

The idea of learning or refining action models using plan demonstrations or examples has been studied extensively in prior work. While some have considered refining incomplete domain models [9], [10], others have focused more on learning action models from scratch [11], [12], [13], [14], [15]. For example, authors in [13] have used transfer learning to learn the action model using a small amount of training data. In [14], authors have moved one step further by developing an algorithm that can learn action models under noisy demonstrations. In [15], authors propose to learn models under stochastic domains. One common assumption in all these methods is the complete knowledge of the state space, which does not hold in our problem. While it is possible for users to directly refine the robot's model [16], it requires deep understanding of planning and is impractical for regular users. Our model refinement process may appear similar to predicate invention [17] that is often used to describe novel high-level concept in inductive logic programming. In contrast, we invent predicates to represent unknown knowledge in planning domain models.

The robot's initial model in our problem may be considered an approximate domain model and hence planning in such a case becomes a type of model-lite planning [18]. Many existing approaches have considered planning with such approximate (or incomplete) domain models [19], [20], [21]. For example, authors in [20] introduce a planning system that can generate plans for an incomplete domain where actions may be missing certain preconditions or effects. Conformant planning [22] may also be considered as a special type of model-lite planning where information about the initial state is missing. Our problem can be viewed as a more general problem where the information about what could be missing is not available.

The ongoing research on abstraction has focused on ensuring certain properties when creating abstractions. For example, the authors in [23] have studied abstractions for producing optimal behaviors that are similar to those in the ground domain. In [24], the authors have investigated and categorized several abstraction mechanisms that retain properties of the ground domain like the Markov property. The fact that the robot's initial model is missing some domain features in our work makes the model an abstraction of the ground model. The relationship of domain concretization to domain abstraction is analogous to that of RL to IRL: domain concretization reverses domain abstraction by refining a given abstract model. While domain abstraction is known to benefit planning, prior work has already started to pay attention to the appropriateness of such abstractions [25], [24]. Our work here can be viewed as addressing unsound abstractions that are introduced *unintentionally* or *unknowingly* in the design phase, due to design flaws or arising from domain ramifications [26] and qualifications [27], [28].

It may be tempting to solve the domain concretization problem based on partially observable markov decision processes (POMDPs) [29], or reinforcement learning methods with POMDPs [30], where the state is partially observable and the robot uses observations to update its belief about the state of the world. However, note that a POMDP still requires complete knowledge about the ground state space, which is not available in our problem formulation due to missing domain knowledge. More specifically, this means that, neither the belief state nor the observation function would be complete, unlike that in POMDP. While it is possible to include the model space as part of the state space of the POMDP, formulating it will be largely intractable.

III. DOMAIN CONCRETIZATION

We first list the assumptions made in our work: 1) Deterministic domains: actions have deterministic effects. Such an assumption is commonly made in robot task planning [31], [32], [33]. 2) Rational teachers: the teacher demonstrations are optimal in the complete domain model, which is a

common assumption made in the literature for learning from examples or demonstrations [34], [35]. 3) Missing information only in the robot's action models and state observations, excluding the goal. This is because goals of the robot are given by the user (teacher) who has access to the complete domain knowledge.

A. Planning Background

We use the Planning Domain Definition Language (PDDL) [36] to define our domain model and problem. Here, a planning problem is defined by a triplet $P = \langle s_0, g, M \rangle$, where s_0 is the start state, g is the set of goal propositions that must be *true* in the goal state and M is the domain model. $M = \langle O, R \rangle$ where R is the set of predicates and O is the set of operators. The set of propositions F and the set of actions A are generated by instantiating all the predicates in R and all the operators in O, respectively. Hence, we can also define $M = \langle A, F \rangle$. A state is either the set of propositions $s \subseteq F$ that are true or $s_\perp = \{\perp\}$. The state $s_\perp$ is a dead state and once it is reached, the goal can never be achieved. The actions change the current state by adding or deleting some propositions. Each action $a \in A$ is specified by a set of preconditions $Pre(a)$, a set of add effects $Add(a)$ and a set of delete effects $Del(a)$, where $Pre(a)$, $Add(a)$ and $Del(a) \subseteq F$. For a model M, the resulting state after executing plan π in state s is determined by the transition function γ, which is defined as follows:

$$\gamma(\pi, s) = \begin{cases} s & \text{if } \pi = \langle\rangle; \\ \gamma(\langle a \rangle, \gamma(\pi', s)) & \pi = \pi' \circ \langle a \rangle. \end{cases} \quad (1)$$

In our problem, we use the *Generous Execution (GE)* semantics as defined in [20] where if an action a is not executable, it does not change the world state s. Hence, the transition function γ for an action sequence with a single action a and state s under GE semantics is defined as follows:

$$\gamma(\langle a \rangle, s) = \begin{cases} (s \setminus Del(a)) \cup Add(a) & \text{if } Pre(a) \subseteq s; \\ s & \text{otherwise.} \end{cases} \quad (2)$$

A plan π is a valid plan for a problem $P = \langle s_0, g, M \rangle$ iff $\gamma^M(\pi, s_0) \sqsupseteq$ (entails) g. The cost of a plan π (denoted by $cost(\pi)$) is the cumulative cost of all the actions in π. For simplicity, we assume uniform-cost actions throughout.

B. Problem Setting and Analysis

Let us revisit the motivating example in Fig. 1. The complete domain model (denoted by M^*) using PDDL is provided in Fig. 2, which includes 4 operators: `open_box`, `grasp`, `place` and `stack`. `place` is used when the box is empty and `stack` is used when the box already contains other items. The goal is to pack all items. The items made of glass are fragile. Note the predicate `not_fragile` used.

In our problem setting, the user (teacher) is assumed to have access to M^*. The robot's initial model (denoted by $\widetilde{M}$), however, may not have knowledge about fragility: the predicate `not_fragile` would be missing (highlighted in Fig. 2). The teacher demonstrations are generated using M^* and *projected* onto $\widetilde{M}$ when observed by the robot, with the missing information removed. More specifically, the robot would observe all actions in a teacher's demonstration but a different state trajectory from the user's perspective. Let us look at an example based on Fig. 1. To simplify the discussion, assume that we concern with only the two items `i1` and `i2` (fragile). Given a training case with the initial state that `i2` is on top of `i1`:

- A robot's would-be plan π based on $\widetilde{M}$ (5 actions): ⟨`open_box b1`, `grasp i2`, `place i2 b1`, `grasp i1`, `stack i1 i2 b1`⟩;
  Initial state *observed* by the robot: `i2` on top of `i1`
- A teacher's demonstration z based on M^* (6 actions): ⟨`open_box b1`, `grasp i2`, `place i2 b1`, `open_box b2`, `grasp i1`, `place i1 b2`⟩;
  Initial state *observed* by the teacher: `i2` on top of `i1` and **`i2` is fragile**

In the teacher demonstration z, the teacher opens another box to pack `i2` since `i1` cannot be stacked on a fragile item. Since this information is unknown to the robot, the robot's plan would ignore such a constraint, resulting in a shorter (and less costly) plan. However, note that the teacher demonstration provide hints about the hidden issue. In particular, from the robot's perspective, z does not conform with its model $\widetilde{M}$ since z and π have different costs for the same problem, thus violating the assumption of rational teachers if $\widetilde{M} = M^*$. The process of *domain concretization* is hence hinged on addressing such nonconformities. Given a teacher demonstration z as shown above, a nonconformity is introduced when $cost(z) \neq cost(\pi)$: the teacher has chosen a more or less costly plan than the optimal plan in the robot's model $\widetilde{M}$, which should not occur if $\widetilde{M}$ is complete. In fact, given our assumptions, the only possibility is $cost(z) > cost(\pi)$ since abstractions relax planning constraints.

When nonconformity is detected, $\widetilde{M}$ must be concretized. In our approach, we generate new models by adding new predicates to the preconditions and/or effects of the actions and the initial state. Each such model is then tested for conformity. Models passing the test become candidate models. When multiple teachers demonstrations are available, we require a candidate model to conform with all the demonstrations. Addressing the detected nonconformities does not automatically guarantee recovery of the complete model.

C. Problem Formulation

A domain model $\widetilde{M} = \langle \widetilde{O}, \widetilde{R} \rangle$ is incomplete in that it is missing a set of predicates, denoted by $\widehat{R}$ (unknown), that are present in the complete domain $M^* = \langle O^*, R^* \rangle$.

- $\widetilde{R} \subseteq R^*$ and $\widetilde{R} = R^* \setminus \widehat{R}$
- $\widetilde{Pre(o)} = Pre(o) \setminus \widehat{R}$
- $\widetilde{Add(o)} = Add(o) \setminus \widehat{R}$
- $\widetilde{Del(o)} = Del(o) \setminus \widehat{R}$

Definition 1. *The problem setting of Domain Concretization is a setting where the agent has access to an initial domain model $\widetilde{M}$ and a set of **observed** teacher demonstrations under M^* (which are projected onto $\widetilde{M}$).*

We denote the set of teacher demonstrations as ζ^*. Each demonstration $z^* \in \zeta^*$ is a tuple $\langle s_0^*, g, \tau \rangle$ where s_0^* and g

```
(:types box item - object
       metal - item glass - item ...)
(:action open_box
:parameters (?b - box)
:precondition (and (handempty))
:effect (and (box_open ?b)))

(:action grasp
:parameters (?m - item)
:precondition (and (on_shelf ?m) (handempty))
:effect (and (holding ?m) (not (on_shelf ?m))
      (not (handempty))))

(:action place
:parameters (?m1 - item ?b - box)
:precondition (and (holding ?m1) (box_open ?b)
      (box_empty ?b))
:effect (and (item_packed ?m1) (handempty)
      (on_top ?m1 ?b) (not (holding ?m1))
      (not (box_empty ?b))))

(:action stack
:parameters (?m1 - item ?m2 - item ?b - box)
:precondition (and (holding ?m1) (box_open ?b)
      (not_fragile ?m2) (on_top ?m2 ?b))
:effect (and (item_packed ?m1) (handempty)
      (on_top ?m1 ?b) (not (on_top ?m2 ?b))
      (not (holding ?m1))))
```

Fig. 2: Packing domain description in PDDL. The predicate in bold (not_fragile) is present in the teacher's model M^* but is missing from the robot's model $\widetilde{M}$.

are the initial state and goal of the associated training case, respectively. τ is an action sequence. The robot observes a *projected* demonstration $\langle \widetilde{s_0},\, g,\, \tau \rangle \in \widetilde{\zeta}$, where $\widetilde{s_0} = s_0^* \setminus \widehat{R}$.

Definition 2. *Planning under Incomplete Domain Knowledge (PIDK) is defined as $\widetilde{P} = \langle \widetilde{s_0}, g, \widetilde{M}, \widetilde{\zeta}, \rho' \rangle$, which is the problem of generating a plan that has at least ρ' probability of success for $P^* = \langle s_0^*, g, M^* \rangle$ (test case).*

In the training phase, our method identifies a set of candidate domain models that explain the teacher's demonstrations for the training cases. In the testing phase, our method generates robust plans for new planning problems (i.e., test cases as initial state and goal pairs (s_0^*, g)) under the set of candidate models. Note that the robot observes only $\widetilde{s_0} = s_0^* \setminus \widehat{R}$ in test cases as well.

D. Candidate Model Generation via Model Search

To reduce the set of candidate models, we make a minimalistic assumption: we search only for those models that require the minimum number of new predicates and changes to be introduced into $\widetilde{M}$. Without this assumption, the candidate model space would be infinite as we can always introduce dummy predicates to a model without affecting its candidacy. The motivation for minimum model changes can be attributed to the principle of *Occam's Razor* [37].

We transform the problem of generating candidate models to a search problem in the model space. When nonconformities are detected, we gradually increase the number of predicates to be added until a candidate model can be identified. Denote the set of possible predicates as X, which is generated from all possible combinations of typed arguments (e.g., box and item in Fig. 2) in the domain with a maximum arity. Each search state in the model space is a domain model (denoted by M) generated by adding some predicates from X to their possible missing positions in $\widetilde{M}$. Since these new predicates may also be present in the initial state (as propositions), each candidate model M must also be checked against an initial state s_0 whose projection onto $\widetilde{M}$ is $\widetilde{s_0}$. s_0 may not be unique. A model M is accepted as a candidate model if it passes the model test below for at least one s_0. Our model-space search is defined as follows:

- Initial State: $M = \widetilde{M}$

- Action Set Λ**:** $\{\alpha_\chi^{Pre(o)}\} \cup \{\alpha_\chi^{Add(o)}\} \cup \{\alpha_\chi^{Del(o)}\}$

$$\forall \chi \in X \text{ and } \forall o \in O \text{ where } O \in M.$$

An action α in the model-space search represents a predicate χ being added to $Pre(o)$, $Add(o)$ or $Del(o)$ of an operator o in the current model M. Each action thus represents a unit change to the model.

- Successor Function T**:** $T(M, \alpha_\chi^{Pre(o)}) = M'$. T produces new model M' where $R' = R \cup \chi$ and $Pre(o') = Pre(o) \cup \chi$ where $o' \in O'$ and $O' \in M'$. Similarly, we can define $T(M, \alpha_\chi^{Add(o)})$ and $T(M, \alpha_\chi^{Del(o)})$.

- Model (Goal) Test: $C_1(M) \wedge C_2(M) \wedge C_3(M)$:

1) $C_1(M)$: Plan Validity Test returns *true* if:

$$\forall \langle \widetilde{s_0}, g, \tau \rangle \in \widetilde{\zeta}, \gamma^M(\tau, s_0) \sqsupseteq g \tag{3}$$

This ensures that all the teacher demonstrations are executable and achieve the goal under M.

2) $C_2(M)$: Well-Justification Test returns *true* if:

$$\forall \langle \widetilde{s_0}, g, \tau \rangle \in \widetilde{\zeta}, \forall a_i \in \tau, \gamma^M(\tau \setminus \{a_i\}, s_0) \not\sqsupseteq g \tag{4}$$

This ensures that the demonstrations are well-justified [38] in M, which means that if any action is removed from the demonstration the goal will not be achieved.

3) $C_3(M)$: Plan Optimality Test returns *true* if:

$$\forall \langle \widetilde{s_0}, g, \tau \rangle \in \widetilde{\zeta}, cost(\pi^M) = cost(\tau) \tag{5}$$

where π^M is the optimal plan for the problem $P = \langle s_0,\, g,\, M \rangle$. This condition ensures that the teacher demonstrations are optimal under M. Note that C_3 and C_1 imply C_2 but C_2 is easier to test than C_3. When C_1 succeeds but C_2 fails, C_3 is guaranteed to fail so its test is no longer needed.

Claim 1. *C_1 and C_3 are necessary and sufficient to ensure model M can generate τ, $\forall \langle \widetilde{s_0}, g, \tau \rangle \in \widetilde{\zeta}$.*

Claim 2. *C_2 is a necessary condition for a demonstration τ to be optimal, where $\langle \widetilde{s_0}, g, \tau \rangle \in \widetilde{\zeta}$.*

We solve the problem by uniform cost search. For any search state in the model space, let X' ($X' \subseteq X$) be the set of predicates added to the current domain model M. We generate U as the set of possible propositions that can be instantiated from X'. The possible initial state s_0 satisfies $s_0 = \widetilde{s_0} \cup \mu$, where $\mu \in 2^U$. We gradually increase $|\mu|$ until a candidate model can be found that passes the model test and return all candidate models for that $|\mu|$. Each candidate model is associated with a set of s_0's for which the model

tests were passed for the set of training cases. When the same model passes the tests with different sets of s_0's, they will be considered as different candidate models.

In our packing domain in Fig. 2 discussed in Sec. III-B, we start with $\widetilde{M}$ as the model M and one predicate missing. Denote the missing predicate as `pred_1`. Since M fails $C_3(M)$, the search is started by generating X. X includes all the possible predicates like `(pred_1 ?b - box)`, `(pred_1 ?b - box ?m - item)`, `(pred_1 ?b - box ?m - metal)`, etc. We order them first based on the number of arguments, and then from the most to the least general argument types. Using X, we generate the set of actions Λ, such like $\alpha_\chi^{Pre(stack)}$ $\forall \chi \in X$, which means χ will be added to the preconditions of operator `stack`.

E. Heuristic-based Model Generation

To contain the computational complexity, we further present a heuristic-based search method. The idea is to identify model changes that (partially or fully) address the nonconformity with the teacher demonstrations, instead of checking all possible models. For the packing domain in Fig. 2, consider a case where $\widetilde{M}$ is missing the predicate `(box_open ?b)` so that any teacher demonstration with the `open_box` action will create a nonconformity: C_2 (*unjustified action*) will fail because the goal will be achieved even after deleting the `open_box` actions. In such as case, we can generate the next set of models such that C_2 returns true. For example, we can generate a model by inserting a new predicate to the add effects of action `open_box` *and* to the preconditions of the action `place`.

The action set Λ' now is a set of actions where each encodes multiple model changes (instead of a unit change). The search process is similar as before except that if a model fails the model test on a teacher demonstration $\langle s_0,\, g,\, \tau\rangle$, instead of returning *false*, it returns a set of actions $B \subseteq \Lambda'$, which is used to generate the possible models for the next step. Furthermore, instead of checking the model M for all possible s_0's, we check only for those that are returned along with the action set. The action set B is returned as follows:

- *Unsatisfied Precondition*: Here, the demonstration is not executable in M because of some unsatisfied precondition. This means $C_1(M)$ returns *false* for an action $a_i \in \tau$, $1 \leq i \leq |\tau|$, such that $Pre(a_i) \not\subseteq \gamma^M(\langle a_1, a_2, ...a_{i-1}\rangle, s_0)$. In such a case, we return the action set $B = \{(\alpha_\chi^{Add(o_j)})\}$ $\forall a_j \in \tau$ and for $\chi \in \Delta$, where Δ is the set of preconditions missing for a_i and o_j is the operator corresponding to action a_j and $1 \leq j \leq i-1$. Alternatively, χ (after instantiation according to $\langle s_0,\, g,\, \tau\rangle$) may be added to the initial state s_0. This is because a missing proposition could either be present in the add effects of any previous action or in the initial state.

- *Unjustified action*: This happens when some action in the demonstration is not well-justified in M. This means $C_2(M)$ returns *false* for some action a_i such that $\gamma^M(\tau \setminus \{a_i\}, s_0) \sqsupseteq g$. In such a case, we return the action set $B = \{(\alpha_\chi^{Add(o_i)}, \alpha_\chi^{Pre(o_j)})\} \cup \{(\alpha_\chi^{Add(o_i)}, \alpha_\chi^{Pre(o_j)}, \alpha_\chi^{Del(o_j)})\}$ $\forall a_j \in \tau$ and $\forall \chi \in X$. o_i and o_j are operators corresponding to actions a_i and a_j respectively and $i+1 \leq j \leq |\tau|$. Intuitively, this generates B such that a_i cannot be removed from τ, which makes it well-justified under the updated M.

- *Sub-optimal demonstration*: This happens when the optimal plan under M is less costly (shorter) than a teacher's demonstration. In such a case, $C_3(M)$ would return *false*, which means there exists an action that is not possible in τ under M^* but is possible in π^M under M. Hence, the operator corresponding to that action is missing some precondition. In such a case, $\exists a_i, a_i'$ such that $a_i \in \tau$ and $a_i' \in \pi^M$ and $a_i \neq a_i'$, $1 \leq i \leq n$ $(n = |\pi^M|)$. We return the action set $B = \{(\alpha_\chi^{Pre(o_j)})\} \cup \{(\alpha_\chi^{Pre(o_j)}, \alpha_\chi^{Del(o_j)})\}$ $\forall a_j' \in \pi^M$ and $\forall \chi \in X$, where o_j is the operator corresponding to a_j' and $i \leq j \leq n$.

For the example mentioned where `box_open` is missing, if τ = $\langle$`open_box b1, grasp i1, place i1 b1`$\rangle$ while the action `open_box` is considered not needed in M, then $B = \{(\alpha_\chi^{Add(open\_box)}, \alpha_\chi^{Pre(grasp)}), (\alpha_\chi^{Add(open\_box)}, \alpha_\chi^{Pre(place)})\} \cup \{(\alpha_\chi^{Add(open\_box)}, \alpha_\chi^{Pre(grasp)}, \alpha_\chi^{Del(grasp)}), (\alpha_\chi^{Add(open\_box)}, \alpha_\chi^{Pre(place)}, \alpha_\chi^{Del(place)})\}$ $\forall \chi \in X$. This will generate 4 new models for each predicate $\chi \in X$.

Theorem 3. *(Soundness) The candidate models found by the heuristic-based search can generate all the teacher demonstrations, with the minimum number of changes to $\widetilde{M}$.*

Proof. While generating the action set B in the heuristic-based search, the process checks to see if the model test is satisfied. If so, it accepts the model as a candidate. Under Claim 1, it can be concluded that a candidate model will be able to generate all the teacher demonstrations if the heuristic-based search finds it. A uniform cost search ensures that the number of changes made to $\widetilde{M}$ is minimum. □

Theorem 4. *(Completeness) The heuristic-based search finds all the models satisfying the model test with the minimum number of changes to the incomplete model $\widetilde{M}$.*

Proof Sketch. We can prove this by induction. The basic idea is to show that in each iteration, B includes all the possibilities in which a predicate may be added to make the model satisfy the model test for the training case (teacher demonstration) considered. The uniform cost search ensures that the number of changes is minimal. □

F. Online Model Generation

In the real world, it is desirable to have an online search method that consider teacher demonstrations as they arrive. The search procedure is similar to the heuristic-based method above and starts with the incomplete model $\widetilde{M}$. The difference being that search process is performed against only one teacher demonstration at a time, referred to as an iteration. Within each iteration, the search finds a set of candidate models $\mathcal{M}$ that conform to the demonstration considered. The next iteration begins with these models when a new teacher demonstration is provided. In terms of computation, the online heuristic-based search is expected to perform better. This is because it greedily searches for a set of candidate models at each iteration with the minimal changes.

Hence, the solution produced by the online method may be dependent on the order of the teacher demonstrations considered. It is not guaranteed to recover a candidate model for all training cases even when one exists. This is because the online search considers only a single demonstration at a time. Hence, it can stop prematurely at shallower levels when the true model resides at deeper levels.

G. Robust Planning

Given the set of candidate models $\mathcal{M}$, when given a new task $(\widetilde{s_0}, g)$, we find a robust plan such that it has the highest probability of achieving the goal under the weighted set of candidate models $\mathcal{M}$. Similar to [20], we compile the problem of generating a robust plan into a Conformant Probabilistic Problem (CPP). A Conformant Probabilistic Problem [7] is defined as $P' = \langle I,\ g,\ D,\ \rho\rangle$ where I is the belief over the initial state, D is the domain model, and ρ is the acceptable goal satisfaction probability. The domain model $D = \langle A', F'\rangle$, where F' is the set of propositions and A' is the set of actions. Each $a' \in A'$ has the set of preconditions $Pre(a') \subseteq F'$ and $E(a')$, the set of conditional effects. Each $e' \in E(a')$ is a pair of $con(e')$ and $\dot{o}(e')$, where $con(e') \subseteq F'$ enables e' and $\dot{o}(e')$ is a set of outcomes ϵ. The outcome ϵ is a triplet $\langle Pr(\epsilon),\ add(\epsilon),\ del(\epsilon)\rangle$, where $add(\epsilon)$ adds the proposition ϵ to and $del(\epsilon)$ deletes it with probability $Pr(\epsilon)$. A compilation that translates the PIDK problem $\widetilde{P} = \langle \widetilde{s_0}, g, \widetilde{M}, \widetilde{\zeta}, \rho'\rangle$, when given $\mathcal{M}$, to a conformant probabilistic planning problem P' is defined as follows:

For each candidate model $M_i \in \mathcal{M}$, a proposition m_i is introduced. Let the set of these propositions be $\widehat{M}$. Further, a set $\widehat{F} = \bigcup_{i=1}^{n} F_i$ is introduced, where F_i is the set of propositions instantiated by predicates $R_i \in M_i$, where $M_i \in \mathcal{M}$ and n is the total number of models in $\mathcal{M}$. In the compiled problem, the set of propositions $F' = \widehat{M} \cup \widehat{F}$. For each model $M_i \in \mathcal{M}$, a set U_i' is created, which is the set of new propositions that are not present in $\widetilde{M}$. The domain model D for P' is created from $\widetilde{M}$ as follows:

- A new action a_0' that initializes the initial state with the missing propositions introduced for each candidate model M_i. The action a_0' has $Pre(a_0') = \emptyset$; furthermore, $\forall m_i \in \widehat{M}$, a conditional effect $e_i' \in E(a_0')$ is created such that $con(e_i') = \{m_i\}$ and each outcome $\epsilon_j \in \dot{o}(e_i')$ has $add(\epsilon_j) = \mu'$ and $del(\epsilon_j) = \emptyset$ where $\mu' \in 2^{U_i'}$. For each outcome, $Pr(\epsilon_j) = 1/|2^{U_i'}|$. This initializes the initial state for each model M_i, considering all the possibilities equally likely (since we have no information about them in the test case).
- For each action $a \in \widetilde{A}$ in $\widetilde{M}$, if model $M_i \in \mathcal{M}$ adds a proposition $u_i' \in U_i'$ to $\widetilde{Pre(a)}$, a conditional rule $e_i \in E(a)$ is created, such that $con(e_i) = \widetilde{Pre(a)} \cup \{m_i\} \cup \{u_i'\}$, $add(e_i) = \widetilde{Add(a)}$ and $del(e_i) = \widetilde{Del(a)}$. For example, if in model M_i, action `place i1` has the new proposition `(pred_0 i1)` in its preconditions, then the action in the compiled domain will have a conditional rule where $con(e_i) = \widetilde{Pre(\texttt{place i1})} \cup \{m_i\} \cup \{\texttt{(pred\_0 i1)}\}$; $add(e_i)$ and $del(e_i)$ will remain the same.
- For each action $a \in \widetilde{A}$ in $\widetilde{M}$, if model $M_i \in \mathcal{M}$ adds a proposition $u_i' \in U_i'$ to $\widetilde{Add(a)}$, a conditional rule $e_i \in E(a)$ is created, such that $con(e_i) = \widetilde{Pre(a)} \cup \{m_i\}$, $add(e_i) = \widetilde{Add(a)} \cup \{u_i'\}$, $del(e_i) = \widetilde{Del(a)}$.
- For each action $a \in \widetilde{A}$ in $\widetilde{M}$, if model $M_i \in \mathcal{M}$ adds a proposition $u_i' \in U_i'$ to $\widetilde{Del(a)}$, a conditional rule $e_i \in E(a)$ is created, such that $con(e_i) = \widetilde{Pre(a)} \cup \{m_i\}$, $add(e_i) = \widetilde{Add(a)}$, $del(e_i) = \widetilde{Del(a)} \cup \{u_i'\}$.

The initial belief state $I = (\bigwedge_{f \in \widetilde{s_0}} f) \wedge (oneof(m_1, m_2, ...m_n))$ where $oneof$ returns *true* when exactly one of its input is *true*. In the compiled problem, we set $\rho = \rho'$. A conformant plan is calculated for the given problem using a conformant probabilistic planner. The plan so obtained is a robust plan for problem $\widetilde{P} = \langle \widetilde{s_0},\ g,\ \widetilde{M}, \widetilde{\zeta}, \rho'\rangle$ with success probability of at least ρ' (see below). To search for the most robust plan, we can start with $\rho' = 1$ and gradually decrease it until a plan is found.

Theorem 5. *If $\pi' = \langle a_0', a_1', a_2', a_3', ...a_n'\rangle$ is a plan for the complied problem P' with goal satisfaction probability ρ, then ρ is also the (lower bound of the) probability of success of the plan $\pi = \langle a_1, a_2, a_3, ...a_n\rangle$ for the problem $\widetilde{P}$.*

Proof Sketch. The action a_0' initializes all possible initial states for each model $M_j \in \mathcal{M}$ in $\mathcal{M}$, resulting in a set of new candidate models (denoted by $\mathcal{M}'$) that also specify the initial state. Given the set of all such models $\mathcal{M}'$, it is easy to see that there is a bijective mapping between the initial state in the conformant planning problem and the candidate models in $\mathcal{M}'$. We can use induction to prove that a solution to any given initial state in the conformant planning problem is also a solution to a planning problem under the corresponding candidate model. Therefore, we can conclude that if π' achieves the goal with probability ρ in P', then the probability of success of π in the problem $\widetilde{P}$ is also ρ. □

IV. EVALUATION

We first evaluate our methods on various IPC domains to show their effectiveness under incomplete domain knowledge. Then, we create a more complex version of our packing domain to show the practical benefits. Plans are generated with Fast-Downward [39]. For solving conformant probabilistic planning problems, probabilistic-FF [7] is used.

A. IPC Domains

For this evaluation, we used two IPC domains. In the *rover* domain, there are multiple rovers each equipped with capabilities like sampling soil, rock, and capturing images at different waypoints. The second domain is a slightly modified version of the *gold-miner* domain. In this domain, we have a grid-world and the task is to pick up gold from a particular cell and deposit it in another. Some cells have a laser or a bomb that could be used to clear the blocked cells. We introduced incompleteness by removing predicates that

Doms	# T	Model Searched				Candidate Models				Time(secs)				Plan Success				Avg Cost Inc		
		BF	HS	HS+	OS	BF	HS	HS+	OS	BF	HS	HS+	OS	HS	HS+	OS	Baseline	HS	HS+	OS
One predicate missing at a time																				
Rovers	3	1500	400	70	400	2	2	1	2	205.78	20.11	4.95	37.80	8/8	8/8	8/8	0/8	+0.25	+0.25	+0.25
	3	1500	200	40	200	2	2	1	2	47.62	10.51	3.8	10.98	8/8	5/8	8/8	0/8	+0.63	+0.8	+0.63
	5	10300	840	70	300	3	3	1	3	520.29	80.06	6.05	184.52	11/11	8/11	11/11	0/11	+0.64	+0.75	+0.64
Miner	2	700	30	10	30	1	1	1	1	8.74	0.90	0.82	1.12	12/12	12/12	12/12	0/12	+1.25	+1.25	+1.25
	4	2520	430	30	70	1	1	1	1	517.69	156.14	12.83	31.07	10/10	10/10	10/10	0/10	+1.00	+1.00	+1.00
	2	140	10	3	10	1	1	1	1	2.42	0.49	0.45	0.62	12/12	12/12	12/12	0/12	+1.25	+1.25	+1.25
Two predicates missing at a time																				
Rovers	3	-	26500	5900	26600	-	4	1	4	-	645.11	142.8	484.96	7/7	3/7	7/7	0/7	+0.57	1.33	+0.57
	6	-	179620	14470	31900	-	3	1	3	-	7873.6	753.18	1484.72	8/8	5/8	8/8	0/8	+0.25	+0.4	+0.25
	6	-	185200	14620	32000	-	3	1	3	-	7456.40	741.49	1671.37	8/8	8/8	8/8	0/8	+0.00	+0.00	+0.00
Miner	4	-	10140	620	3600	-	1	1	1	-	560.09	55.05	192.62	10/10	10/10	10/10	0/10	+1.00	+1.00	+1.00
	4	-	3920	610	620	-	1	1	1	-	298.35	41.17	41.72	10/10	10/10	10/10	0/10	+1.00	+1.00	+1.00
	2	121300	260	110	260	1	1	1	1	1610.92	3.29	2.69	2.83	12/12	12/12	12/12	0/12	+1.25	+1.25	+1.25
Three predicates missing at a time																				
Rovers	6	-	-	127940	1175000	-	-	1	3	-	-	3172.64	25610.00	-	8/8	8/8	0/8	-	+0.25	+0.25
Miner	4	-	227110	3900	11420	-	1	1	1	-	6441.61	95.98	364.22	10/10	10/10	10/10	0/10	+1.00	+1.00	+1.00

TABLE I: Comparison results of the proposed methods with a baseline (assuming complete models) for the IPC domains

can be generated by some actions and are simultaneously preconditions of other actions. For example, in the rover domain, the precondition to capture an image is that the camera should be calibrated, which can be produced by the $calibrate$ action. For each domain, we created multiple incomplete domains by removing a different number of predicates (up to 3) randomly from the set of possible missing predicates under the given domain. For this experiment, we removed only those predicates that were not present in the initial state. Using the complete domain model, optimal plans were generated for training cases and used as the teacher demonstrations. These demonstrations were then projected onto the incomplete model and given to the agent.

Table I shows the results of our experiments. For each incomplete domain, the results are presented for 4 methods: Brute Force (BF), Heuristic-based Search (HS), and Online heuristic-based Search (OS); For HS, we also implemented a random select strategy with parallel instances that maintains only a fixed percentage of the children nodes when expanding any node, which we referred to as HS+. For our experiments we used 20%. Here, each incomplete domain was given multiple training cases (#T column in table) with teacher demonstrations. It turned out only a few cases (around 2-6) were needed for most methods in this experiment. For HS+, we ran 12-15 instances in parallel and chose the one that found candidate models the fastest. After identifying the candidate models, we then generated 8-12 test cases of different problems and tested our robust planning method. Blank cells ("-") in the table represents the situation where the model search time exceeded a predefined limit (180 mins for scenarios missing 2 predicates and 480 mins for scenarios missing 3 predicates). The brute-force approach timed-out in almost every domain tested when multiple predicates were missing. Heuristic-based search (either HS, HS+ or OS) reduced the number of models searched by a considerable amount. We can also observe that, as the number of missing predicates increased, computational time increased by a significant amount. This is due to the exponential growth of the search space. The online search method performed better than the heuristic-based search method especially when multiple predicates were missing. HS+ is the fastest among all but it does not always cover the complete model (and hence its plan is not always successful). Note that had we used $\widetilde{M}$ to generate the plans, they would often fail (see the *Baseline* column in the table). Results also show that the plans generated by our methods only slightly increased the plan cost (i.e., the average length for successful plans), when compared to the plan in the complete domain.

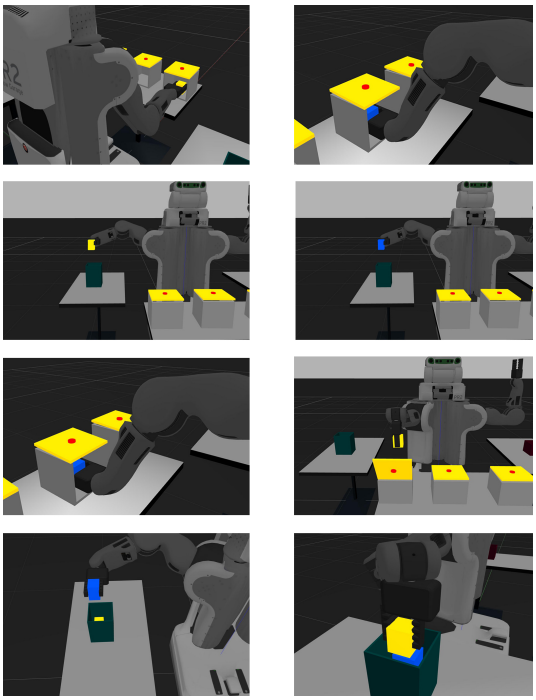

Fig. 3: Comparison of the action sequences generated with a standard planning method (left) and our method provided with a few teacher demonstrations (right). A video for comparing the different behaviors is provided as an attachment.

B. Simulated Robotics Domain

In this experiment, we have created a simulated robotics domain which is a more complex version of our packing domain. In this domain, we now have two constraints for packing items into boxes. As before, the first constraint is that a fragile item cannot be stacked. The second constraint is that a fragile item cannot be dropped into the box (via `drop`) and instead must be placed carefully (via `place`). We also introduced two grasping actions: `horizontal_grasp` and `vertical_grasp`. For `horizontal_grasp`, the surroundings of the item to be picked up should be clear. For `vertical_grasp`, clear surroundings are not a necessity.

Some of the items may be stored in containers (see Fig. 3). In such cases, when using `vertical_grasp`, the container must be opened first by pressing a button on the top. On the other hand, this is not needed for `horizontal_grasp` since the contains are directly accessible from the side. Furthermore, if the robot picks up an item horizontally, it must use `drop` action instead of `place` since it cannot twist its wrist (assumed). For vertical grasp, both `place` and `drop` are possible. The goal is the same as before.

Fig. 3 shows the setup of our simulated experiments. The left sequence (top to bottom) is the one that used $\widetilde{M}$ to plan, which is missing knowledge about the fragility of items. As expected, the robot was not able to distinguish between fragile items (in yellow) and non-fragile items (in blue). Hence, it used `horizontal_grasp` to pick up the fragile item and `drop` to put the fragile item into the box, which could damage the item. Furthermore, in the subsequent actions, the robot stacked an item over the fragile item, which was also undesirable. On the other hand, the actions executed using our method (HS) is shown on the right of Fig. 3, after providing the teacher demonstrations for a few training cases (different from the scenario shown in Fig. 3). The robot first picked up the non-fragile item using `horizontal_grasp` and then put it into the box using the `drop` action. Then it used `vertical_grasp` followed by `place` to stack the fragile item carefully over non-fragile item.

V. CONCLUSIONS & FUTURE WORK

In this paper, we have formally introduced the problem of *Domain Concretization* and discussed its prevalence to robot planning. We have presented a solution that uses teacher demonstrations and an initial model to generate a set of candidate models and then search for a robust plan that achieves any test case under the maximum number of candidate models. We have formulated the model search process and developed a heuristic-based search to make the search more efficient. For practical use, we have also presented an online version of this search method. Our methods were tested on IPC domains and a simulated robotics domain where our methods significantly increased the success rate of planning. This work opens up many research directions. For example, the conditions under which domain concretization will converge to include the complete model would be useful to study. Furthermore, this work is limited to deterministic domains and rational teachers. Extending our approach to relax these assumptions would be meaningful next steps.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[2] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *ICML*, 2000.

[3] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008.

[4] W. Mao and J. Gratch, "A utility-based approach to intention recognition," in *AAMAS*, 2004.

[5] O. Schrempf and U. Hanebeck, "A generic model for estimating user intentions in human-robot cooperation," in *ICINCO*, 2005.

[6] D. C. Logan, "Known knowns, known unknowns, unknown unknowns and the propagation of scientific enquiry," *Journal of experimental botany*, vol. 60, no. 3, pp. 712–714, 2009.

[7] C. Domshlak and J. Hoffmann, "Probabilistic planning via heuristic forward search and weighted model counting," *JAIR*, 2007.

[8] A. Sharma, P. R. Medikeri, and Y. Zhang, "Domain concretization from examples: Addressing missing domain knowledge via robust planning," *arXiv:2011.09034*, 2020.

[9] H. H. Zhuo, T. Nguyen, and S. Kambhampati, "Model-lite case-based planning," in *AAAI*, 2013.

[10] ——, "Refining incomplete planning domain models through plan traces," in *IJCAI*, 2013.

[11] Q. Yang, K. Wu, and Y. Jiang, "Learning action models from plan examples using weighted max-sat," *Artificial Intelligence*, 2007.

[12] H. Zhuo, Q. Yang, D. Hu, and L. Li, "Learning complex action models with quantifiers and logical implications," *Artificial Intelligence*, 2010.

[13] H. H. Zhuo and Q. Yang, "Action-model acquisition for planning via transfer learning," *Artificial Intelligence*, 2014.

[14] H. H. Zhuo and S. Kambhampati, "Action-model acquisition from noisy plan traces," in *IJCAI*, 2013.

[15] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *JAIR*, 2007.

[16] R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz, "Tell me when and why to do it! run-time planner model updates via natural language instruction," in *HRI*, 2012.

[17] I. Stahl, "Predicate invention in ilp — an overview," in *Machine Learning: ECML*, 1993.

[18] S. Kambhampati, "Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models," in *AAAI*, 2007.

[19] C. Weber and D. Bryce, "Planning and acting in incomplete domains," in *ICAPS*, 2011.

[20] T. Nguyen, S. Sreedharan, and S. Kambhampati, "Robust planning with incomplete domain models," *Artificial Intelligence*, 2017.

[21] Y. Zhang, S. Sreedharan, and S. Kambhampati, "Capability models and their applications in planning." in *AAMAS*, 2015, pp. 1151–1159.

[22] A. Cimatti and M. Roveri, "Conformant planning via symbolic model checking," *JAIR*, vol. 13, pp. 305–338, 2000.

[23] D. Abel, D. E. Hershkowitz, and M. L. Littman, "Near optimal behavior via approximate state abstraction," in *ICML*, 2016.

[24] S. Srivastava, S. Russell, and A. Pinto, "Metaphysics of planning domain descriptions," in *AAAI*, 2016.

[25] B. Marthi, S. Russell, and J. Wolfe, "Angelic semantics for high-level actions," in *ICAPS*, 2007.

[26] J. J. Finger, "Exploiting constraints in design synthesis," Ph.D. dissertation, Stanford, CA, USA, 1987.

[27] J. McCarthy, "Epistemological problems of artificial intelligence," in *IJCAI*, 1977.

[28] M. L. Ginsberg and D. E. Smith, "Reasoning about action ii: The qualification problem," *Artificial Intelligence*, 1988.

[29] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *AIJ*, 1998.

[30] T. Jaakkola, S. Singh, and M. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," *Advances in Neural Information Processing Systems*, 1999.

[31] T. Chakraborti, G. Briggs, K. Talamadupula, Y. Zhang, M. Scheutz, D. Smith, and S. Kambhampati, "Planning for serendipity," in *IROS*. IEEE, 2015, pp. 5300–5306.

[32] Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo, and S. Kambhampati, "Plan explicability and predictability for robot task planning," in *ICRA*. IEEE, 2017, pp. 1313–1320.

[33] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila, "Toward human-aware robot task planning." in *AAAI spring symposium: to boldly go where no human-robot team has gone before*, 2006.

[34] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.

[35] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[36] M. Fox and D. Long, "Pddl2.1: An extension to pddl for expressing temporal planning domains," *JAIR*, 2003.

[37] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," *Information Processing Letters*, 1987.

[38] E. Fink and Q. Yang, "Formalizing plan justifications," 1997. [Online]. Available: https://kilthub.cmu.edu/articles/Formalizing_Plan_Justifications/6605831

[39] M. Helmert, "The fast downward planning system," *JAIR*, 2006.