Do You See What I See?
An Egocentric View of our Pansophical Planning Problems

Xiaotian Liu, Alison Paredes, Christian Muise
Queen's University, Kingston, ON, Canada
{liu.x, 20asp3, christian.muise}@queensu.ca

Abstract

Classical planning takes a pansophical view of the world: everything is fully known, observed, and static. While there are extensions to partial observability, this leapfrogs an important intermediate step of embodied agent design: egocentricity. In this work, we propose a semi-automated mechanism that allows planning domain designers to convert classical planning problems into an egocentric alternative. The generated planning problems are classical as well, and we introduce an open-loop replanning mechanism that progressively explores the egocentric space until the original goal is solved (or deemed unsolvable). Our work serves as a crucial first step towards embodied agents that can be equipped with an appropriately specified egocentric version of known environment dynamics.

1 Introduction

Most classical planning problems are defined with a pansophical reference frame where the environment is fully observable. However, in many planning tasks, an agent only has a limited view of the environment. Existing approaches on partially observable planning problems, such as conformant or contingent planning, use belief states to represent uncertainty in the environment (Hoffmann and Brafman 2005; 2006). However, representing belief states often requires access to information such as what objects are present in the environment in advance. Many planning applications do not provide access to such information. These types of egocentric agents often need to incorporate exploration in the strategy (James, Rosman, and Konidaris 2019). Egocentricity can also be necessary for embedded agents to transfer skills across environments (Charniak 2020). Often, solving a planning problem egocentrically requires manual conversion by a domain expert. The conversion process can be time-consuming, sometimes requiring the definition of a new set of syntax. A standardized semi-automatic approach can lift the burden of manual conversion, decreasing the hurdle of egocentric planning research and open up the door to domain-independent planning techniques to this setting.

We propose a novel semi-automatic approach that can convert classic planning problems defined in PDDL into egocentric alternatives. Our method takes a set of user-defined object types with their initial states as input, and outputs the egocentric version of the original problem. We assume that the world is made up of objects. Thus predicates define all objects' states and relationships. By defining which objects can be observed by an egocentric agent, we can extract the observable state space from the corresponding predicates. We use an exploration algorithm that searches through the original problem's state space until a plan is found or deemed impossible. The iterative exploration is facilitated by an open-loop replanning mechanism, which progressively updates the observable state space. Thus, our overall framework consists of two interacting processes: one determines which set of fluents are observable, and the other explores the unobserved state spaces.

The egocentric planning sub-problems generated by our approach can be solved using off-the-shelf planners. Our algorithm also keeps syntax and vocabularies consistent with the original planning problem, making results easily interpretable. We test the validity of our approach by using it on five classical planning problems and corresponding egocentric interpretations of them. Our experimental results demonstrate that our approach is practical through both quantitative and qualitative evaluations. To the best of our knowledge, there has not been any attempt to semi-automatically convert classic pansophical planning problems into egocentric alternatives. More complex or domain-specific problems may require further modification of our proposed approach. Nevertheless, we believe this work is an important step forward that can encourage more automated planning research in egocentric settings.

In the following sections, we provide preliminary definitions and a detailed outline of our framework, followed by descriptions of our experimental setup and a discussion of the results. In addition, we provide a step by step example of our approach using a simple grid based planning problem. Finally, we will conclude this paper with a summary of our work and possible future directions.

2 Preliminaries

Modified STRIPS Notation

We extend the commonly used STRIPS notation to include the notion of objects. In STRIPS, a planning problem is defined with a tuple $\langle F, I, A, G \rangle$. F is a set of fluents, $I \subseteq F$ is the initial state of the world, and $G \subseteq F$ is the goal state. A is the set of actions available. For each action

$a \subseteq A, PRE(a) \subseteq F$ is the precondition of that action, $ADD(a) \subseteq F$ is the add effect, and $DEL(a) \subseteq F$ is the delete effect. Our method operates over planning problems specified in PDDL. Therefore, we consider fluents to be represented by predicates with typed objects. We define a set O to be the set of objects in each planning problem. We include O in the STRIPS problem, and our complete planning problem is defined as a tuple $P = \langle O, F, I, A, G \rangle$.

We will demonstrate our approach using the standard language for specifying automated planning problems, PDDL (Haslum et al. 2019). The details of the PDDL language are beyond the scope of this paper, and we refer the interested reader to (Haslum et al. 2019) for a complete discussion.

Grid Navigation Example

We use a 2D navigation problem specified in PDDL as an example to illustrate our approaches throughout this paper. The problem is the simplified version of the Search-and-Rescue problem (Teichteil-Königsbuch and Fabiani 2007). In this planning problem, an agent is spawned on a 2D grid, and its goal is to search and pick up a person and then navigate to the hospital. The planning problem includes 4 object types: a robot object, location objects, a person object, and a hospital object. The predicates include object location predicates, connection predicates connecting locations, and a holding predicate indicating whether an agent is holding a person. The action set includes moving to another location and picking up the person. A visual illustration of the problem with the corresponding PDDL definitions are shown in Figures 1 and 2.

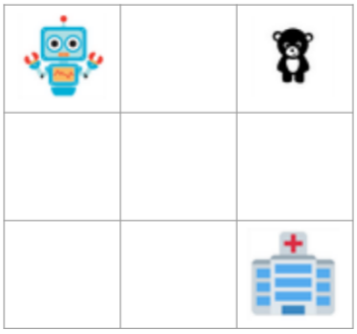

Figure 1: Graphical illustration for Search-and-Rescue

```
DOMAIN:
(define (domain searchandrescue)
  ......
  (:predicates
    (conn ?v0 - location ?v1 - location ?v2 - direction)
    (robot-at ?v0 - robot ?v1 - location)
    (person-at ?v0 - person ?v1 - location)
    (hospital-at ?v0 - hospital ?v1 - location)
    (carrying ?v0 - robot ?v1 - person)
    (handsfree ?v0 - robot)
    (move ?v0 - direction)
    (pickup ?v0 - person)
    (dropoff))

    (:action move-robot
        :parameters (?robot - robot ?from - location ?to - location ?dir - direction)
        :precondition (and (move ?dir)
            (conn ?from ?to ?dir)
            (robot-at ?robot ?from))
        :effect (and
            (not (robot-at ?robot ?from))
            (robot-at ?robot ?to))  )

    (:action pickup-person ......)

    (:action dropoff-person ......) ) )

PROBLEM:
(define (problem searchandrescue)
    (:domain searchandrescue)
    ......
    (:init
        (conn f0-0f f0-1f right)
    ......
        (conn f2-2f f1-2f up)
        (dropoff )
        (handsfree robot0)
        (move up)
        (move down)
        (move left)
        (move right)
        (pickup person0)
        (robot-at robot0 f0-0f)
        (person-at person0 f0-2f)
        (hospital-at hospital0 f2-2f)
    (:goal
        (person-at person0 f2-2f)))
```

Figure 2: PDDL for Search-and-Rescue

3 Approach

For a planning problem to be egocentric, only parts of the state space are observable. We can construct an egocentric planning problem $P^{'} = \langle O^{'}, F^{'}, I^{'}, A^{'}, G^{'} \rangle$ from the original pansophical planning problem by finding the corresponding subsets for each element in P. $\langle O^{'}, F^{'}, I^{'} \rangle$ are the egocentric version of the problem's objects, fluents and initial states. Their value depends on an agent's observable state space. $A^{'}$ is the set of actions available to the egocentric agent. It contains actions in the original pansophical set A with additional actions required for exploration. $G^{'}$ is the goal of the egocentric agent, which is to gather information until the original goal G can be achieved. The objectives of our approach are to determine the current egocentric state and to facilitate exploration. We separate these tasks into two separate algorithms. The first algorithm extracts the observable states from an pansophical planning state space. We call this algorithm the Egocentric Subset Extractor, or ESE. The second algorithm facilitates exploration and information gathering in an partially observable environment. We call this component the Iterative Exploration via Replanning, or IER. The subsequent sections will motivate and describe both algorithms in detail, followed by a demonstrated example of our approach on the grid world domain.

Egocentric Subset Extractor

The ESE algorithm determines which fluents in the pansophical planning problem are observable. We adopt an object-centric view of the world and assume objects' states/relationships are represented by fluents in the form of object typed predicates. Thus, if we can determine which set of objects are observable by an agent, we can find the corresponding set of fluents that are also observable. The visibility of certain objects depends on the agent's egocentric state, and we refer to these as *anchor objects*. The ESE algorithm determines which subset of fluents in the initial set in a planning problem $P = \langle O, F, I, A, G \rangle$ should be included in the egocentric problem.

Our approach requires the user to determine the type and

the initial set of observable objects. We define this set S as a tuple $\langle T, C, R \rangle$. T contains the anchoring object types, and C is a set of anchor objects currently observable. R is defined as a set of predicates that connect one anchor object to another. We make the following four assumptions for our approach:

1. We can uniquely determine a planning problem's egocentric aspects by way of the observable anchor objects.
2. If an object is observable, then its states and relationships with other objects, in the form of predicates, are also observable.
3. Any predicates that do not contain any anchor objects are always observable.
4. There exist a set of predicates that define connections or relations among anchor objects.

The first assumption is a direct result of adopting an object-centric view of the world. The second assumption allows us to determine which fluents in the form of predicates are observable. Intuitively, predicates represent the state and relationship among objects. Thus any predicates that contain observable objects should also be observable. For the third assumption, if a predicate contains no anchor objects, its observability is independent of an agent's egocentric state. We assume these predicates should always be observable by the agent. Finally, we assume anchor objects are connected via relational predicates. Agents should be able to observe other anchor objects that are immediately related to the current set of anchor objects. These related anchor objects are used to determine where the agent need to explore, which we will discuss in a later section. Our ESE algorithm first iterates through all the relational predicates of type R to extract observable relational predicates and their anchor objects. It then extracts all the non-relational predicates that are either always observable or containing observable anchor objects. The extracted predicates define the egocentric state of the agent. The full ESE algorithm is shown in Algorithm 1.

In the grid example, the user needs to define a set of initial observable anchor objects, anchor types and relational predicates, $S = \langle T, C, R \rangle$. Anchor objects in this problem are the location objects. The relational predicates are `(conn )`. Initially, only `f0-0f` is observable by the agent. The ESE algorithm first extracts all the relational predicates containing `f0-0f`. The algorithm then finds all location objects related to the `f0-0f` which are `f0-1f` and `f1-0f`. Next, the ESE algorithm extracts non-relational predicates in initial state I that contains `f0-0f`, `f1-0f`, `f0-1f`. `(robot-at robot0 f0-0f)` is extracted as a result. Finally, all fluents contain no location objects are considered observable and are also extracted. The egocentric initial state $I^{'}$ is shown in Figure 3.

Algorithm 1: Egocentric Subset Extractor

Input: $P = \langle O, F, I, A, G \rangle$
Anchor Object Set $S = \langle T, C, R \rangle$
Output: Egocentric projection $P^{'}$

Initialize $O^{'} = \emptyset$ and $I^{'} = \emptyset$;
for $p \in I$ ***where*** *type(p)* $\in R$ **do**
  if *p has object* $o \in C$ **then**
    for $o \in p$ **do**
      if *type(o)* $\in T$ **then**
        $O^{'} = O^{'} \cup \{o\}$;
for $p \in I$ ***where*** $type(p) \notin R$ **do**
  if *p has* $o \in O^{'}$ ***or*** *p is constant* **then**
    $I^{'} = I^{'} \cup \{p\}$;
    for $o \in p$ **do**
      $O^{'} = O^{'} \cup \{o\}$;
$F^{'} = F$;
$A^{'} = A$;
$G^{'} = G$;
return $P^{'} = \langle O^{'}, F^{'}, I^{'}, A^{'}, G^{'} \rangle$;

Iterative Exploration via Replanning

An egocentric agent can only observe a subset of the whole state space. Thus, a way to explore an unknown environment is often required to formulate a plan. An exploration strategy must identify which part of the state space needs to be explored while keeping track of state information observed previously. Our Iterative Exploration via Replanning (IER) algorithm is such an approach that progressively explores unknown or partially known environments. We designed IER to generate and modify existing PDDL files directly. As a result, we can use off-the-shelf planners directly.

In addition to $P^{'} = \langle O^{'}, F^{'}, I^{'}, A^{'}, G^{'} \rangle$ generated by the ESE algorithm, IER requires the user to identify an additional set of *exploration actions*, E, as input. We define exploration actions as actions that an agent needs to change its current observed state space.

By taking actions in E, an agent will transit to a state where new objects and fluents are observable.

An exploration action a must have at least one anchoring object in its parameters. Both $PRE(a)$ and $ADD(a) \cup DEL(a)$ must also have predicates containing these anchoring objects. To distinguish between observed and unobserved objects, we create `(unknown ?obj)` predicates to identify unobserved objects. For an exploration action a, `(unknown o)` predicates are added to $PRE(a)$ for each anchor object, `o`, declared in the parameters set. After the action is taken, the unknown object is deemed to be revealed and will be removed using $DEL(a)$.

When a plan with the original goal cannot be found, our algorithm sets exploration as a goal. We achieve this by introducing an `(exploration)` predicate to replace the original goal state. Since exploration is achieved through exploration actions in E, we need to add the `(exploration)` goal predicate to $ADD(a)$ for each $a \in E$. After an exploration step, a new planning problem $P^{'}$ is generated using ESE. These steps are repeated iteratively through replanning until a plan for the original goal can be

```
(:init
    (conn f0-0f f0-1f right)
    (conn f0-0f f1-0f down)
    (conn f0-1f f0-0f left)
    (conn f1-0f f0-0f up)
    (dropoff )
    (handsfree robot0)
    (move up)
    (move down)
    (move left)
    (move right)
    (pickup person0)
    (robot-at robot0 f0-0f)
    )
```

Figure 3: Initial condition extracted by ESE

found. IER is shown in detail in Algorithm 2.

In the grid example, the action required for the agent to move to another egocentric state is the `move-robot` action. IER first iterates through all the fluents in the initial condition set $I^{'}$ and identifies unknown locations by adding `(unknown f1-0f)` and `(unknown f0-1f)` fluents. To make an exploration action, IER creates a new action `explore` by adding `(unknown location)` in `move-robot`'s precondition, `(not (unknown location))` and `(explored)` predicates in its effect set. The `exploration` action is illustrated in Figure 4. Since no plan can be found for $G^{'}$, we replace the goal state with $G^{'} = \{\texttt{(explored)}\}$. This forces the agent to move to another location to gather more information about the broader environment. In the next iteration, the ESE algorithm will extract a new set of observable objects followed by IER until a plan with the original goal can be found or the problem is deemed unsolvable.

```
(:action exploration
    :parameters (?robot - robot ?from - location ?to
                    - location ?dir - direction)
    :precondition (and (move ?dir)
        (conn ?from ?to ?dir)
        (robot-at ?robot ?from)
        (unkown ?to))
    :effect (and
        (not (robot-at ?robot ?from))
        (robot-at ?robot ?to))
        (not (unkown ?to))
        (explored))
```

Figure 4: An exploration action example

4 Evaluation

We tested our approach empirically on five classical planning domains written in PDDL. These domains are Blocks World, Minecraft, Search-and-Rescue, Sokoban, and Elevators. In addition, we used classical planning problems de-

Algorithm 2: Iterative Exploration via Replanning

Input: Problem $P = \langle O, F, I, A, G\rangle$
Anchor object set $S = \langle T, C, R\rangle$
Exploration action set E
Output: Plan M for the pansophical environment

Initialize $O_e, F_e, I_e, A_e, G_e = \emptyset$;
plan $M = [\,]$;
while *no plan can be found for* P **do**
  $O_e = O$;
  $A_e = A$;
  for $o \in O$ **do**
    if $type(o) \in T$ *and* $o \notin C$ **then**
      $I_e = I_e \cup \{$ `(unknown o)` $\}$;
      for $a \in E$ **do**
        $a^{'} = a.copy()$;
        $PRE(a^{'}) = PRE(a^{'}) \cup$ `(unknown o)`
        $DEL(a^{'}) = DEL(a^{'}) \cup$ `(unknown o)`
        $ADD(a^{'}) = ADD(a^{'}) \cup$ `(explored)`
        $A_e = A_e \cup a'$
  $G_e = \{$`(explored)`$\}$;
  $F_e = F \cup I_e \cup G_e$;
  $\pi = \text{SOLVE}(\langle O_e, F_e, I_e, A_e, G_e\rangle)$;
  $M.extend(\pi)$;
  for $a \in \pi$ **do**
    if $a \in E$ **then**
      add anchor objects in a to C;
  $I_e = \text{PROGRESS}(I_e, \pi)$;
  $G_e = G$;
  $P = \text{ESE}(\langle O_e, F_e, I_e, A_e, G_e\rangle, \langle T, C, R\rangle)$;
$M.extend(\text{SOLVE}(P))$;
return M;

fined in the PDDLGym libarary(Silver and Chitnis 2020), which contains domain and problem files written in PDDL, along with visualization APIs. For each planning problem, the user only needs to add the `(unknown ?obj)` predicate, the `(explored )` predicate, and the exploration actions. No modifications are needed for the problem file. It takes only several minutes for manual conversion if the user is already familiar with the domain and PDDL modelling in general.

For each planning domain, we tested five different problem setups. The results are summarized in Table 1. We evaluate the performance based on the percentage of successful conversions from an pansophical planning problem to the egocentric alternative. In addition, we compared the average number of steps required for the egocentric agent to solve a planning problem. We used Tarski(Ramírez and Francès 2021) to parse each planning domain, and the actual planning is done on the Planning.Domains online solver (Muise 2016): `http://solver.planning.domains`.

The results show that our method can successfully convert most classical planning problems to an egocentric version. For Search-and-Rescue, Sokoban and Blocks World, our algorithm can successfully convert 100% of the testing problems. In the case of Sokoban, we were able to covert

	Success Rate	Egocentric Plan Len	Pansophical Plan Len
Search-&-Resc.	100%	26	10
Blocks World	100%	16	11
Elevator	100%	29	22
Sokoban	75%	64	41
Minecraft	0%	NA	27

Table 1: Results of tested planning domains

four out of five problems. Unlike the other four problems, Sokoban contains irreversible actions that can result in dead-ends. The failure occurred when the agent can no longer achieve its original goal due to such actions. The only domain where our method is not applicable is Minecraft. Although there is the notion of a grid location in Minecraft, the agent can reach all locations in a single step. Our method relies on connection predicates to facilitate gradual exploration. If an agent can reach a state without passing through any other state, our method will treat both states as visible in the egocentric formulation.

5 Discussion

The experiments show that our algorithm can successfully convert the Blocks World and the Elevator domains to egocentric equivalents. This demonstrates that our approach is not limited to just agent navigation problems. Both Search-and-Rescue and Sokoban are 2D navigation problems where the notions of agent sand egocentricity are apparent. In both problems, agents are defined explicitly by the domain designer as `agent` or `robot` objects. However, an explicit definition may not always be necessary. We can define agents as the set of actions that can change the observable state. For example, in Blocks World this action set is `{stack, unstack}`. Intuitively, the agent is whomever that have the ability to move the blocks.

The location object is an obvious choice to define egocentricity in 2D navigation problems. However, defining it for Blocks World is more challenging. The set of observable states changes depending on the perspective of the observer. Our solution adopts a (literal) top-down perspective, limiting the observable state to only blocks that are on the top of stacks. We define the block object as the anchor object. Each block is connected via `(on ?block1 ?block2)` connection predicates. We then set `unstack` action as the exploration action. A block is considered explored when the `unstack` is conducted. Our IER algorithm then extracts a new block to explore using the `(unknown ?block)` predicate. The Blocks World formulation shows as long as all required inputs can be satisfied, our method can convert a planning domain without an explicit definition of an agent.

When the goal of a planning problem can no longer be reached, the problem is considered unsolvable. Our method treats exploration as a way to gather information about the state-space independent of the original goal. Thus, such exploration could lead the agent to reach a dead-end where the original goal is no longer reachable. For problems like Search-and-Rescue and Blocks World, all actions are reversible, and the initial goal will always be solvable when enough information is gathered. However, in Sokoban, the agent can reach a dead-end via irreversible actions. For example, when a block is pushed to a corner, the agent will not be able to push the block back. In our implementation, our agent does not consider the feasibility of the original goal when exploring. One can avoid such situations via dead-end checking algorithms, but only in situations where there is sufficient information available to the agent for them to reliably avoid these dead-ends.

One of the future directions we would like to take this work is applying dead-end detection techniques such as (Lipovetzky, Muise, and Geffner 2016) to avoid actions that cause dead-ends. We want to integrate the agent's original goal with exploration for more goal oriented exploration strategies. In this work, we only tested our method on classical planning domains. These domains are not representative of real life planning settings. We want to eventually apply our techniques in embodied egocentric agent design to solve planning problems such as in (Shridhar et al. 2020).

6 Related Work

Defining the environment in an egocentric manner is important to many agent based applications. Charniak demonstrates the utility of the egocentric view in reinforcement learning in a Grid World setting (2020). They show that the egocentric view improves the learning algorithm's ability to apply the learned policy to new problems never before seen during training. Zhang et al. proposed an egocentric vision-based assistive co-robot system (2013). This work allows humans to actively engage in control loops via egocentric camera and gesture input. Bertasius, Chan, and Shi presented a generative adversarial network model that use first-person images to generate a realistic basketball sequence via egocentric motion planning (2018). All these defined planning and egocentricity in their respective domain-specific settings. In contrast, our method focuses on egocentricity in the classic domain-independent planning setting.

There exists work on planning under partial observability, such as conformant, contingent, and epistemic planning. However, we distinguish egocentric planning from these partial observable planning settings with regards to the use of the belief space. Conformant can be interpreted as classical planning in belief space, and contingent planning adds uncertainty to the observable state space and can be formulated as and-or search problems in the belief space (Bonet and Geffner 2000; Hoffmann and Brafman 2005; 2006). Epistemic planning represents belief states as epistemic states which can be reasoned using epistemic logic (Bolander 2017). Planners constructed for all of these partially observable planning settings often require explicitly defining the possible initial belief space. This requires the users of these planners to have knowledge of all possible objects that will be present in the planning problem, *in advance*. However, in many planning problems, an acting agent might not have access to all unique objects in advance, and new objects and relationships need to be discovered during

exploration. For example, an egocentric agent that is navigating in a 2D grid might not know all possible "location" objects to formulate a proper belief state. The approach we take here is to iteratively convert information the agent has observed as a fully observable planning problem to determine whether the original goal can be reached or more exploration is needed. However, the fully observable planner we used can be replaced by conformant, contingent, or epistemic planners when dealing with uncertainties with agents, actions, or the environment. That is to say, those areas of planning under partial observability are *complementary* to our work, and may be incorporated in future work.

Some previous works have studied planning in open worlds, a similar class of problems to the egocentric problems we define in this paper. Talamadupula et al. used a hindsight optimization method to solve planning problems in partially observable worlds (2010). The proposed method uses a prior distribution to generate and aggregate samples of close-world problems that can be solved using an off-the-shelf planner. Kiesel et al. proposed a novel partial-satisfaction goal construct that allows predefined objects to be discovered via replanning (2012). James, Rosman, and Konidaris presented a framework that derives egocentric views of planning problems for the purpose of learning portable representations, sufficient for planning, of classes of tasks (2019). These representations are learned from traces of actions and transitions. Our work, however, derives these views from the planning problem's pansophical description. The advantage is our approach does not require training data. Jiang et al. introduces a method to identify and reason over objects in an environment via a database (2019). The approach converts open world problems to close world settings via hypothetical instances of unknown objects. All these works adopted assumptions about open-world problems that we expose via our approach to deriving egocentric problems. However, these problems require complete reformulation of close world planning domains via their respective specifications. To the best of our knowledge, we are the first to propose a method to semi-automatically convert classical planning problems into an egocentric alternative with relative ease.

7 Summary

In this work, we have proposed an open-loop replanning method for converting classical pansophical planning problems into egocentric alternatives. Our method is semi-automated and requires very little change to the original problem. Our approach consists of an Egocentric Subset Extractor to extract the observable state space of an agent. It also contains the Iterative Exploration via Replanning algorithm that facilitates exploration in an unknown environment. We tested our method on five classical planning problems and converted most of these problems to their corresponding egocentric versions. The results also demonstrate that our approach is not limited to grid navigation problems such as Search-and-Rescue and Sokoban. It can convert problems, such as Blocks World and Elevator, where the notion of the agent is not explicitly defined. Our work serves as a crucial first step towards embodied agents that can be equipped with an appropriately specified egocentric version of known environment dynamics.

References

Bertasius, G.; Chan, A.; and Shi, J. 2018. Egocentric basketball motion planning from a single first-person image. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 5889–5898. IEEE Computer Society.

Bolander, T. 2017. A gentle introduction to epistemic planning: The DEL approach. In Ghosh, S., and Ramanujam, R., eds., *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017*, volume 243 of *EPTCS*, 1–22.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In Chien, S. A.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge, CO, USA, April 14-17, 2000*, 52–61. AAAI.

Charniak, E. 2020. Extrapolation in gridworld markov-decision processes. *CoRR*.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proc. ICAPS*, volume 2005.

Hoffmann, J., and Brafman, R. I. 2006. Conformant planning via heuristic forward search: A new approach. *Artif. Intell.* 170(6-7):507–541.

James, S.; Rosman, B.; and Konidaris, G. D. 2019. Learning portable representations for high-level planning. *CoRR*.

Jiang, Y.; Walker, N.; Hart, J. W.; and Stone, P. 2019. Open-world reasoning for service robots. In Benton, J.; Lipovetzky, N.; Onaindia, E.; Smith, D. E.; and Srivastava, S., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019*, 725–733. AAAI Press.

Kiesel, S.; Burns, E.; Ruml, W.; Benton, J.; and Kreinmendahl, F. 2012. Open World Planning via Hindsight Optimization University of New Hampshire. Technical report, University of New Hampshire.

Lipovetzky, N.; Muise, C.; and Geffner, H. 2016. Traps, invariants, and dead-ends. In *The 26th International Conference on Automated Planning and Scheduling*.

Muise, C. 2016. Planning.Domains. In *The 26th International Conference on Automated Planning and Scheduling - Demonstrations*.

Ramírez, M., and Francès, G. 2021. Tarski. Accessed on 2021-05-17.

Shridhar, M.; Thomason, J.; Gordon, D.; Bisk, Y.; Han, W.; Mottaghi, R.; Zettlemoyer, L.; and Fox, D. 2020. ALFRED:

A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 10737–10746. IEEE.

Silver, T., and Chitnis, R. 2020. Pddlgym: Gym environments from PDDL problems. *CoRR* abs/2002.06432.

Talamadupula, K.; Benton, J.; Kambhampati, S.; Schermerhorn, P. W.; and Scheutz, M. 2010. Planning for human-robot teaming in open worlds. *ACM Trans. Intell. Syst. Technol.* 1(2):14:1–14:24.

Teichteil-Königsbuch, F., and Fabiani, P. 2007. A multi-thread decisional architecture for real-time planning under uncertainty. In *3rd ICAPS'07 Workshop on Planning and Plan Execution for Real-World Systems*.

Zhang, J.; Zhuang, L.; Wang, Y.; Zhou, Y.; Meng, Y.; and Hua, G. 2013. An egocentric vision based assistive co-robot. In *IEEE 13th International Conference on Rehabilitation Robotics, ICORR 2013, Seattle, WA, USA, June 24-26, 2013*, 1–7. IEEE.