

THESIS

USE OF MINIMAL ABSTRACTION TO ADDRESS GOAL UNDERSPECIFICATION

Submitted by

Philip D. Hopkins

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2026

Master's Committee:

Advisor: Sarath Sreedharan

Nathaniel Blanchard

Jeremy Daily

Copyright by Philip D. Hopkins 2026

All Rights Reserved

ABSTRACT

USE OF MINIMAL ABSTRACTION TO ADDRESS GOAL UNDERSPECIFICATION

The problem of identifying and accounting for under-specified or misspecified goals has been widely considered central to AI safety. While most work in this space focuses on querying users about potential negative side effects, almost none look at the question of addressing the cause of misspecification, namely, the user’s misunderstanding regarding the consequences of pursuing the specified goal. In this work, this shortcoming is addressed by identifying the minimum level of abstraction at which the user can reason about possibly unavoidable side-effects of the current specification. This enables the user to make informed choices about what side effects are acceptable. A planning compilation is also introduced that allows the calculation of both the minimal abstraction and the set of unavoidable side-effects effectively. The effectiveness of the proposed method is evaluated through computational experiments, performed on standard planning benchmarks, and a user study.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Professor Sarath Sreedharan. He took a chance on me as a researcher, and I am grateful for the opportunity that he gave to me. I am grateful for the guidance from him in all aspects of my journey to this point, and I am deeply grateful for him to taking the time to help me to understand things.

I would also like to thank Professor Nathaniel Blanchard and Professor Jeremy Daily for being part of my committee. I am grateful for your willingness to be on the committee, and willingness to help in the event that I needed it.

I would also like to thank those in the HAPI lab, including Erfan Mirhaji, Septia Rani, Dennis Kim, Brittany Cates, Trisha Ghali, Roya Daneshi, and Kazim Abrar Mahi. They talked to me about things, laughed with me, and challenged me. I appreciate all of their support. I would especially like to thank Dennis Kim for his help with the user study in this thesis, and especially the statistical analysis performed.

I would lastly like to thank all of the faculty in the Department of Computer Science. Each of them have been very welcoming and helpful to me, and I appreciate their guidance.

DEDICATION

For those who believed in me, even when I would not believe in myself.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
DEDICATION	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
Chapter 1 Introduction	1
Chapter 2 Background	4
2.1 Classical Planning	4
2.2 Model Abstraction and Abstraction Lattice	5
Chapter 3 Related Work	7
Chapter 4 Goal Recommendation Problem	9
4.1 Planning Compilation	14
Chapter 5 Evaluation	18
5.1 Computational Experiments	18
5.2 Discussion	19
5.3 User Study	23
5.3.1 User Study Experimental Setup	24
5.3.2 User Study Details	25
5.3.3 User Study Results	31
Chapter 6 Conclusions	53
Bibliography	54

LIST OF TABLES

5.1	Description and Results of Problems from IPC Domains using an Optimal Planner. Results of 5 trials in the form of (average $\pm$ standard deviation). F = Number of Fluents in Original Problem, A = Number of actions in the original grounded model, CF = Number of fluents in the compiled model, CA = Number of actions in the compiled model, $tN(s)$ = Time needed to solve the original problem in seconds. The largest Sokoban instance only used one copy.	20
5.2	Description and Results of Problems from IPC Domains using an Optimal Planner. Results of 5 trials in the form of (average $\pm$ standard deviation). CF = Number of fluents in the compiled model, CA = Number of actions in the compiled model, $numS$ = Number of Sets produced, $tC(s)$ = Time taken by the procedure that uses planning compilations in seconds, $tEX(s)$ = Time taken by the exhaustive search method. Rover and Trucks had an instance where some copies of the problem timed out, and those times are not included. The largest Sokoban instance only used one copy.	21
5.3	Description and Results of Problems from IPC Domains using a Satisficing Planner. Results of 5 trials in the form of (average $\pm$ standard deviation). CF = Number of fluents in the compiled model, CA = Number of actions in the compiled model, $numS$ = Number of Sets produced, tC = Time for Solving Compiled Problem, $RamC$ = Memory Usage on Compiled Problem. The largest Sokoban instance only used one copy.	22
5.4	Propositions used in the Laundry Robot scenario of the User Study	26
5.5	Explanations displayed in the Laundry Robot scenario of the User Study when no plan is possible. The rules are evaluated from top to bottom, with the explanation of the first rule found to be true is displayed.	27
5.6	Explanations displayed in the Laundry Robot scenario of the User Study when a plan is possible.	28
5.7	Propositions used in the Light Bulb Robot scenario of the User Study	37
5.8	Explanations displayed in the Light Bulb Robot scenario of the User Study when no plan is possible. The rules are evaluated from top to bottom, with the explanation of the first rule found to be true is displayed.	38
5.9	Explanations displayed in the Light Bulb Robot scenario of the User Study when a plan is possible.	39

LIST OF FIGURES

1.1	A diagrammatic overview of this method applied to a robotics problem. The user instructs the robot to move to point B. The method presented here is able to identify that there are two unavoidable consequences, namely the robot being dirty or the wheels being covered in grease. Each of these side effects can individually be described in two abstract models in the lattice, and to the user, a combination of these two models is presented as the final abstract model, where the user can reason about both potential side effects.	2
5.1	User study screenshot showing end of IRB consent form.	26
5.2	User study screenshot showing details presented to the user about the first scenario. . .	29
5.3	User study screenshot showing the first scenario, with the user presented with a list of fluents.	30
5.4	User study screenshot showing first scenario, showing the entire list of fluents that the user must give preference about.	31
5.5	User study screenshot showing first scenario, with the explanation provided to the user when they have chosen a goal that is not possible.	32
5.6	User study screenshot showing showing first scenario, with the explanation provided to the user when they have chosen a goal that is not possible. This explanation differs from the explanation when all are True.	33
5.7	User study screenshot showing showing first scenario, with the plan provided to the user that achieves their selected goals.	34
5.8	User study screenshot showing the NASA TLX questions asked of the user following each scenario.	35
5.9	User study screenshot showing the pair-wise NASA TLX questions asked of the user following each scenario.	36
5.10	User study screenshot showing details presented to the user about the second scenario. . .	40
5.11	User study screenshot showing the second scenario, with the user presented with two sets of fluents to choose between.	41
5.12	User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected set of goals.	42
5.13	User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected set of goals. This shows a different plan from the different selection.	43
5.14	User study screenshot showing final questions presented to the user where a user answers questions comparing the two methods.	44
5.15	User study screenshot showing demographic questions presented to the user.	45
5.16	User study screenshot showing the first scenario, with the user presented with two sets of fluents to choose between.	46
5.17	User study screenshot showing showing first scenario, with the plan provided to the user that achieves their selected set of goals.	47

5.18	User study screenshot showing showing first scenario, with the plan provided to the user that achieves their selected set of goals. This shows a different plan from the different selection.	48
5.19	User study screenshot showing second scenario, showing the entire list of fluents that the user must give preference about.	49
5.20	User study screenshot showing second scenario, with the explanation provided to the user when they have chosen a goal that is not possible.	50
5.21	User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected goals.	51
5.22	User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected goals. The goals selected lead to the desired outcome, but a longer plan.	52

Chapter 1

Introduction

Generating behaviors that align with the user’s true underlying intent, even when the specific goal specification may be under- or misspecified, is widely considered *the central problem* when it comes to AI safety [1]. As such, the problem has been getting increasing attention. Here, goal underspecification scenarios are examined where the user may omit some undesired consequences from the goal specification, which could arise if the agent were to attempt to achieve the specified goal directly. Currently, the majority of the work in this space tends to focus more on identifying potential negative side effects [2] and querying the users about them [3]. Unfortunately, it appears that none of the current works in the space tries to address the true underlying cause for goal underspecification, namely, the fact that the user is not able to correctly anticipate the consequences of the current specification. This may have been caused by the user being unaware of certain aspects of the task, or because the task was too complex for them to correctly reason about the consequences. To address this challenge, a method is developed here that identifies the set of minimal consequences or side effects that cannot be avoided for a given goal specification. This set is presented along with the most abstract representation of the true agent model, where they can correctly reason why these side effects are unavoidable. This method returns the autonomy back to the user, as they can now weigh the different alternatives and decide which one they find most acceptable.

To illustrate this method, consider the example shown in Figure 1.1. Here, there is a robot that is following instructions given by a user. The robot starts on the right side of the map and is instructed to move to point B on the left side. However, there exists no path the robot could take where the only new fact added to the current state is that the robot is in point B. This is because the only two paths to point B involve the robot passing through an area containing spilled grease or that is dusty. So, for the robot to be at point B, either its wheel will be covered in grease or the robot itself will be dusty. The robot might not be able to make the call on which

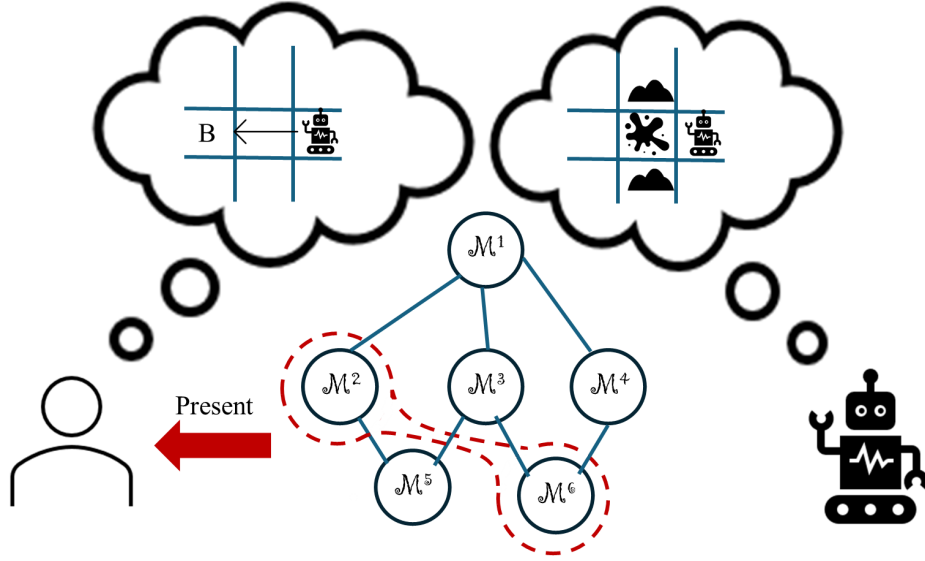


Figure 1.1: A diagrammatic overview of this method applied to a robotics problem. The user instructs the robot to move to point B. The method presented here is able to identify that there are two unavoidable consequences, namely the robot being dirty or the wheels being covered in grease. Each of these side effects can individually be described in two abstract models in the lattice, and to the user, a combination of these two models is presented as the final abstract model, where the user can reason about both potential side effects.

of these is a better outcome. As such, this method will present these two options to the user, along with a simplified task representation that treats it as a simple path planning problem, which now contains the possibility of the robot becoming dusty or being covered in grease under certain conditions. Note that this is a significant simplification of the robot’s true model, as the robot may be a complex system that leverages sophisticated sensors and actuators. However, presented with the information, the user can decide which of these outcomes, if any, is acceptable.

This method will leverage a lattice-based representation of the true task model to identify the consequence set and model abstraction. While such methods have been used in the context of explanation generation [4], this work is the first to investigate such mechanisms, and even model abstractions, in the context of resolving goal misspecifications. Additionally, a planning compilation is introduced that allows calculation of the set of unavoidable consequences and minimal model abstraction directly from the original problem. For the class of planning problems considered, such methods could provide considerable computational advantage, as in the general case,

the lattice could be exponential in the number of fluents in the problem. The proposed approach is evaluated through computational experiments and user studies. The computational experiments on the standard planning benchmark demonstrate the effectiveness of the compilation methods over methods that leverage an explicit search over possible abstractions. The user study also provides evidence that the method provides advantages over simpler querying methods.

In summary, the contributions of this thesis include:

1. A new solution approach to resolve goal underspecification, by allowing reasoning about the consequences of the current goal specification.
2. A novel planning compilation is introduced that helps identify the unavoidable side-effects for a given goal specification, along with the minimal abstraction where such side-effects can be reasoned about.
3. Both computational experiments over standard planning benchmarks and a user study are performed to evaluate the proposed method.

Chapter 2

Background

Next, some basic notations and concepts that will be used throughout the thesis will be established.

2.1 Classical Planning

This work will use classical planning [5] as the base planning formalism. Here, a planning problem or model is described by a tuple of the form $\mathcal{M} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where

- $\mathcal{F}$ is the set of propositional fluents in the problem that determines the state space
- $\mathcal{A}$ is the set of actions in the problem
- $\mathcal{I} \subseteq \mathcal{F}$ is the initial state (the fluents that are true at the beginning of the problem)
- $\mathcal{G}$ is a propositional logical formula defined over $\mathcal{F}$ that captures the goal specification for the current problem

Here, each state s possible under the given model, can be uniquely identified by the set of fluents that are true in the state, i.e, $s \subseteq \mathcal{F}$. Equivalently, each state can be represented by a conjunctive formula that contains a positive literal for each fluent in the set representation and a negative literal for each missing fluent. The notation $\mathcal{G}^{\mathcal{F}}$ is used to denote the list of fluents that are listed as positive literals in the goal specification $\mathcal{G}$.

Here, each action $a \in \mathcal{A}$ is further defined by a tuple of the form:

$a = \langle pre^{\mathcal{M}}(a), add^{\mathcal{M}}(a), del^{\mathcal{M}}(a), c^{\mathcal{M}}(a) \rangle$. Where,

- $pre^{\mathcal{M}}(a)$ a conjunctive formula expressed over $\mathcal{F}$ that must hold true for the action to be performed
- $add^{\mathcal{M}}(a) \subseteq \mathcal{F}$ are the propositions made true by the action

- $del^{\mathcal{M}}(a) \subseteq \mathcal{F}$ are the propositions made false by the action
- $c^{\mathcal{M}}(a)$ is the cost of the action

The result of an action a being performed in a given state $s \subseteq \mathcal{F}$ is captured by a transition function δ defined as

$$\delta(s, a, \mathcal{M}) = \begin{cases} s \cup add^{\mathcal{M}}(a) \setminus del^{\mathcal{M}}(a) & \text{if } pre^{\mathcal{M}}(a) \models s \\ undefined & \text{otherwise} \end{cases}$$

The transition function is overloaded to apply to action sequences. In the context of action sequences, the transition function returns the result of repeatedly executing the actions on the state returned by the previous action.

A solution to a planning problem is a plan, which is a (potentially empty) action sequence $\pi = \langle a_1, a_2, \dots, a_n \rangle$, whose execution in the initial state results in a state that satisfies the goal specification, i.e., $\delta(I, \pi, \mathcal{M}) \models \mathcal{G}$. The cost of a plan is equal to the sum of the individual actions, and a plan is said to be optimal if there exists no plan with a lower cost.

2.2 Model Abstraction and Abstraction Lattice

The paper will focus on model abstractions formed by projecting out a subset of propositions. Not only do these provide mathematically simple abstraction operations, but they also naturally capture scenarios where the user may be unaware of certain aspects of the task. The notations and conventions set by previous works that have used such abstractions in the context of explanation generation [4] are reused here. To begin, the projection operator $\mathcal{P}_{\Lambda}$ is defined, where $\Lambda \subseteq \mathcal{F}$ is the set of propositions being projected out. If $\mathcal{P}_{\Lambda}$ is applied on a state s it results in a new state $\mathcal{P}_{\Lambda}(s) = s \setminus \Lambda$. Similarly, when applied to an action, it removes the propositions in Λ from every model component, and when applied to goal specification, it results in a new conjunctive formula obtained by dropping all literals corresponding to propositions in Λ . This operator, when applied to a model, results in an abstraction since every transition possible in the original model is preserved

in the new model, in so far that for every $\delta(s, a, \mathcal{M}) = s'$, the new model $\mathcal{P}_\Lambda(\mathcal{M})$ has a valid transition $\delta(\mathcal{P}_\Lambda(s), a, \mathcal{P}_\Lambda(\mathcal{M})) = \mathcal{P}_\Lambda(s')$. More formally

Definition 1. For a planning model $\mathcal{M} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, a model $\mathcal{M}' = \langle \mathcal{F}', \mathcal{A}', \mathcal{I}', \mathcal{G}' \rangle$ considered an **abstraction of $\mathcal{M}$** , denoted as $\mathcal{M} \sqsubseteq \mathcal{M}'$, if there exist a set of propositions Λ , such that $\mathcal{F}' = \mathcal{F} - \Lambda$, $\mathcal{I}' = \mathcal{P}_\Lambda(\mathcal{I})$, $\mathcal{G}' = \mathcal{P}_\Lambda(\mathcal{G})$ and every action $a' \in \mathcal{A}'$ is obtained by applying the projection operator $\mathcal{P}_\Lambda$ on some $a \in \mathcal{A}$ with $a' = \langle \mathcal{P}_\Lambda(\text{pre}^\mathcal{M}(a)), \mathcal{P}_\Lambda(\text{add}^\mathcal{M}(a)), \mathcal{P}_\Lambda(\text{del}^\mathcal{M}(a)), c^\mathcal{M}(a) \rangle$.

With the basic projection and abstraction definitions in place, the abstraction lattice can now be discussed. An abstraction lattice allows for the organization of all the potential abstractions of a given concrete model, which is going to be the true task model used by the agent to reason about their plans.

Definition 2. For a model $\mathcal{M}^\#$, the **model lattice $\mathcal{L}$** is a tuple of the form $\mathcal{L} = \langle \mathbb{M}, \mathbb{E}, \mathbb{F}, \ell \rangle$, where $\mathbb{M}$ is the set of lattice nodes, $\mathbb{E}$ the lattice edges, $\mathbb{F}$ the superset of propositions considered for abstraction within this lattice and finally ℓ is a function mapping edges to labels, provided $\mathcal{M}^\# \in \mathbb{M}$ and all nodes $\mathcal{M}' \in \mathbb{M}$ satisfy the condition $\mathcal{M}^\# \sqsubseteq \mathcal{M}'$. Additionally, for each edge $e_i = (\mathcal{M}_i, \mathcal{M}_j)$ there exists a proposition $f \in \mathbb{F}$ such that $\mathcal{P}_f(\mathcal{M}_i) = \mathcal{M}_j$ and $\ell(\mathcal{M}_i, \mathcal{M}_j) = f$.

The assumptions that every lattice is proposition-conserving (every missing proposition has an edge in the lattice) and that every lattice is complete (every subset of propositions has its own model in the lattice) are also made.

An important aspect of these lattices, which will be used extensively here, is the fact that a plan π that is valid in a more concrete model in the lattice is guaranteed to be valid in a more abstract model in the lattice. In other words, if π is a plan for $\mathcal{M}$, then it must be valid for $\mathcal{M}'$, where $\mathcal{M} \sqsubseteq \mathcal{M}'$. This result follows from the definition of the abstraction and projection operator.

Chapter 3

Related Work

The idea of objective misspecification or underspecification has been widely recognized as a major hurdle to achieving value alignment [6], a central criterion for verifiably safe AI systems. While most of the earlier work in this space was focused on decision-theoretic planning settings, where objectives are specified via rewards (cf. [7]), more recent works have started focusing on goal misalignment (cf. [8]). These works have mostly focused on cases where the goal misalignment may arise from their misunderstanding of the underlying task. While these works acknowledge this misunderstanding as the cause for misalignment, the proposed solution strategies focus on querying the user directly to correct the specification, rather than resolving the user’s misunderstanding about the task. This latter approach, which this paper adopts, not only allows the user to provide more accurate specifications in the current interaction but may also help them avoid providing misspecified objectives in future interactions.

A closely related problem to the one studied in the paper is that of avoiding side-effects [9]. Similar to this work, a side-effect could be described as an effect that was not specified in the goal description. The goal of most of these works is to identify which of these side effects may be considered a negative one. Many works in this space rely on human feedback to identify the negative ones (cf. [10, 11]). Others rely on predefined metrics, such as reducing the autonomy of other agents [9] or the ability to complete future tasks [2]. In this work, side effects are not simply being marked as being negative, but rather presented to the user with various alternatives and letting them choose. In this aspect, the approach here is similar to oversubscription planning [12], insofar as different side-effects are assumed to have different utilities, and this underlying utility function is unknown. While oversubscription planning starts with a fully specified goal and looks for achievable subsets, the method here starts from a partially specified one and looks for what supersets can be achieved. However, tools developed for oversubscription planning settings to

help users identify their most preferred achievable subsets (cf. [13]) could also be deployed in the current setting to help improve the user experience.

This work is also closely related to explanations, particularly those related to resolving users' confusion about the task model [14]. In fact, the notion of a model abstraction lattice has been utilized by previous works to generate contrastive plan explanations [4] and to explain planning problem unsolvability [15]. However, the target model information provided here isn't necessarily explanations, but rather the prerequisite task knowledge they require to make sense of why the side-effect may occur. As such, this work is also closely related to works aimed at establishing a shared mental model or common ground [14]. Finally, there has also been work done on using environment design as a means of avoiding model misspecification [16]. Such methods could be used in conjunction with the method presented here to reduce the number of options presented to the user and the level of task details that must be presented.

The concept of understanding side-effects can also be viewed in the lens of hazard analysis [17]. Hazard analysis is concerned with assessing, and mitigating, risk. As one of the first steps in any hazard analysis, the actual hazards must be identified. In this respect, this work can be seen as uncovering side-effects that could be hazards; this could also be seen as identifying hazards that were previously not described. This work is also related to Failure Modes and Effects Analysis (FMEA), which examines failures and the effects that they may have [18]. The work described here is similar in that it can describe the side-effects, good or bad, that may be an unavoidable result due to some initial condition. These initial conditions could include an aspect of the system that either results in a failure or is itself a failure in a system, and the resulting side-effects could be seen as effects of such failures. In these ways, this work can also be seen in terms of identifying hazards, as well as effects of modeled failures.

Chapter 4

Goal Recommendation Problem

The problem starts with an agent who is assigned to achieve some goal $\mathcal{G}$ by a user. It is assumed the agent's model that includes this goal is represented by $\mathcal{M}^\#$. Now, it is assumed that the human model is some abstraction of this concrete model $\mathcal{M}^\#$, as such is contained in the corresponding model abstraction lattice $\mathcal{L}$. Now the problem starts when the agent realizes that it cannot come up with a plan that not only achieves the state factors listed in $\mathcal{G}$, but would also end up adding other facts. As in the case of the robot in the motivating example, it is unclear whether these various possible unintended effects may in fact be acceptable to the user. Now, one could potentially present the user with the set of possible minimal facts that can't be avoided when pursuing $\mathcal{G}$. In the example, these may be the facts `is_dirty` and `wheels_greasy`. Since the user may be operating at a higher level of abstraction, it might be unclear to them why these are unavoidable. As such, additional information might have to be given to the user, such that in their updated mental model, they can correctly reason about the side effects. In the example, this might mean telling them about the locations with the dirt and grease, and what happens when you perform the action `move` on such locations. More formally, the goal recommendation problem is defined as follows:

Definition 3. For an agent model $\mathcal{M}^\# = \langle \mathcal{F}^\#, \mathcal{A}^\#, \mathcal{I}^\#, \mathcal{G}_H \rangle$, where the goal specification $\mathcal{G}_H$ was provided by the user operating with an unknown model $\mathcal{M}^H$, where $\mathcal{M}^\# \sqsubseteq \mathcal{M}^H$, the **goal recommendation problem** or *GRP* is to find a tuple of the form $\mathcal{R} = \langle \mathbb{G}, \hat{\mathcal{M}}^H \rangle$, where $\mathbb{G} = \{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_k\}$ is a set of goals such that every $\mathcal{G}_i$ has $\mathcal{G}_H^\mathcal{F} \subseteq \mathcal{G}_i^\mathcal{F}$ and $\hat{\mathcal{M}}^H$ is a target human model. Here, $\mathcal{R}$ is considered a valid solution if for every goal $\mathcal{G}_i \in \mathbb{G}$ there exists a plan π_i for $\mathcal{M}^\#$, such that the following two conditions are met

$$C1 \quad \delta(\mathcal{I}^\#, \pi_i, \mathcal{M}^\#) \setminus \mathcal{I}^\# = \mathcal{G}_i^\mathcal{F}, \text{ and}$$

$$C2 \quad \delta(\hat{\mathcal{I}}^H, \pi_i, \hat{\mathcal{M}}^H) \setminus \mathcal{I}^\# = \mathcal{G}_i^\mathcal{F}$$

The two conditions here for the solution state that for each alternate goal specification presented to the user, there should exist a plan which, for both the concrete agent model and the abstract target human mental model, only adds the factors listed in the goal specification and nothing more.

There are a few other properties of the definition that are worth keeping in mind. Firstly, here propositions are only considered that were added to the state and not necessarily deleted. Next, the only factors examined are ones that were made true at the goal state and not intermediate facts. Arguably, in many safety-critical situations, both of these could be important factors. However, this simpler formulation was chosen, as this setting can be extended to easily capture both the above-stated conditions. New fluents only need be introduced and action definitions updated in such a way that one set of fluents is created whenever a fact true in the initial state is deleted, and the other is added whenever any fact is made true. With these new fluents being part of the goal state, the previous definition can easily capture these new considerations as well. Additionally, facts already in the initial state that were problematic are not considered; the user would have noticed. Finally, note that this problem only deals with finding a target model and doesn't necessarily focus on how that model information is communicated to the user. Here, the means of communication may depend on the problem setting, and existing work on explaining model information could also be leveraged for this purpose (cf. [4]).

It is worth noting that not all solutions for a goal recommendation problem are equally useful to the user. One could, in particular, think of two dimensions. Firstly, how many alternatives does the solution present, and at what level of abstraction is the target human model? As for the first dimension, a case is considered where the solution consists of all non-overlapping goal specifications containing a minimal number of additional fluents. Such solutions will be referred to as *minimally covering solutions*. More formally,

Definition 4. A solution to GRP is said to be **minimally covering** if there does not exist a goal specification $\mathcal{G}' \notin \mathbb{G}$ such that $\mathcal{G}'$ is not a superset of any $\mathcal{G}_i \in \mathbb{G}$ and for $\mathcal{M}^\#$ satisfies the conditions (C1) and (C2) specified in Definition 3, and there exists no two goal specification $\mathcal{G}_j \in \mathbb{G}$ and $\mathcal{G}_k \in \mathbb{G}$, such that, $\mathcal{G}_j \subseteq \mathcal{G}_k$.

Now, coming to the next criterion, the objective will be to find a solution that contains the most abstract human target model possible. This helps minimize the amount of information to be communicated to the user and the cognitive load they need to suffer to reason about the side effects. In fact, the goal would be to find a minimally covering solution, which contains the most abstract target model possible. Such solutions will be referred to as minimally covering and maximally abstract solutions.

Definition 5. For a given GRP, a solution $\mathcal{R} = \langle \mathbb{G}, \hat{\mathcal{M}}^H \rangle$ is said to be **minimally covering and maximally abstract**, if there exist no other minimally covering solution $\mathcal{R} = \langle \mathbb{G}', \hat{\mathcal{M}}' \rangle$, such that $\hat{\mathcal{M}}^H \sqsubset \hat{\mathcal{M}}'$.

With the basic definitions in place, calculating the actual solutions can be discussed. The first component to such a solution would be to have a procedure that allows testing, for a given goal specification $\mathcal{G}$, and model pair $\mathcal{M}$ and $\mathcal{M}'$, where $\mathcal{M} \sqsubseteq \mathcal{M}'$ and $\mathcal{G} \subseteq \mathcal{F}'$, whether there exists a new goal specification that satisfies the conditions specified in Definition 3. Here, this test is converted into checking the solvability of a new planning problem formed by updating $\mathcal{M}_C$. Here, the problem $\mathcal{M}_C$ will be identical to $\mathcal{M}$, except it will use a new goal specification $\mathcal{G}_C$, given as

$$\mathcal{G}_C = \mathcal{G} \bigwedge_{f \in (\mathcal{F} \setminus \mathcal{F}') \setminus I} \neg f$$

This goal specification ensures that the goal state that results from the plan cannot add any new facts that are either not already in the initial state or part of the fluents in $\mathcal{M}'$. This leads to the first theoretical property of the framework.

Proposition 1. For a model pair $\mathcal{M}$ and $\mathcal{M}'$, such that $\mathcal{M} \sqsubseteq \mathcal{M}'$, and for a goal specification $\mathcal{G}$, where $\mathcal{G} \subseteq \mathcal{F}'$, if there exists a valid plan π for the new model $\mathcal{M}_C$, then π must satisfy the conditions (C1) and (C2) for the models $\mathcal{M}$ and $\mathcal{M}'$ and the fluent set $\delta(I, \pi, \mathcal{M}) \setminus I$.

The proof for this proposition is straightforward. It has been established that any plan valid in $\mathcal{M}$ must be valid in $\mathcal{M}'$. This goal specification ensures that the resultant state cannot contain any

fluent that is not part of the fluent set of the abstract model, or already part of the initial state. This means that $\delta(I, \pi, \mathcal{M}) \setminus I \subseteq \mathcal{F}'$. Since the projection operator will only remove the fluents not part of $\mathcal{F}'$, it can be seen that $\delta(I, \pi, \mathcal{M}) \setminus I = \delta(I', \pi, \mathcal{M}') \setminus I'$. This means that the plan results in the same fluent set in both models, hence proving the conditions, and identifying the possible goal specification to be the conjunctive formula corresponding to $\delta(I, \pi, \mathcal{M}) \setminus I \subseteq \mathcal{F}'$.

Now, with this test in place, one could perform a simple procedure to identify a maximally abstract minimally covering solution. Here, an exhaustive breadth-first search is performed over the nodes in the lattice. Starting with the most abstract node, a test is run on each possible node. Children are added to the fringe only if a plan is not found on the modified compiled model $\mathcal{M}_c^\#$ corresponding to the concrete model. Every time a node is found where the test passes, the corresponding goal and the model are saved to be returned as the final solution. Here, the goal set will correspond to all the goals found, and the target model corresponds to the union of all the models found. Algorithm 1 provides the pseudocode for the procedure.

Theorem 1. *Algorithm 1 is a sound and complete procedure to identify a maximally abstract, minimally covering solution for a given GRP problem, provided the lattice is complete and proposition conserving.*

Proof Sketch. In terms of soundness, the statements made are that the algorithm only returns a minimally covering, maximally abstract solution. The returned information, being a valid solution, comes from Proposition 1. Now, for it not to be a minimal covering solution, the search would have skipped over a solution that is not a superset of one of the goals in $\mathbb{G}$. This isn't possible since it's an exhaustive search, and it only skips a node if its parent supported a solution. As such, any new solution supported by it must be a superset. A similar argument can be made for maximally abstract models. The final union of the model should only contain fluents that will be added to the final state. If it didn't, the search would have found the solution at a more abstract model. Since, per the properties of the abstraction, the plan should still be valid in the abstract model and would have resulted in the same state, since all fluents are part of that model.

Algorithm 1 Exhaustive Search Procedure for Generating $\mathcal{R}$

```
1: procedure EXHAUSTIVE-SEARCH
2:   Input: Lattice  $\mathcal{L}$  with nodes  $\mathbb{M}$ , human goal  $\mathcal{G}^H$ 
3:   Output: Minimally Covering, Maximally Abstract solution for a GRP problem
4:   Procedure:
5:    $\mathbb{G} \leftarrow$  Empty Set
6:    $\hat{\mathbb{M}} \leftarrow$  Empty Set
7:    $Q \leftarrow$  Node Queue
8:   Add to  $Q$  the model  $\hat{\mathcal{M}}_0$  the most abstract model obtained by projecting out  $\Lambda = \mathbb{F}$ 
9:   while  $Q$  is not empty do
10:     $\hat{\mathcal{M}} \leftarrow \text{Pop}(Q)$ 
11:     $\mathcal{M}_c^\# \leftarrow \mathcal{M}^\#$  modified with  $\mathcal{G}^c = \mathcal{G}^H \bigwedge_{f \in (\mathcal{F}^\# \setminus \hat{\mathcal{F}}) \setminus I^\#} \neg f$ 
12:    if  $\text{GetPlan}(\mathcal{M}_c^\#)$  finds a solution  $\pi$  then
13:       $\mathcal{G}_i \leftarrow$  conjunction over fluents in  $\delta(I^\#, \pi, \mathcal{M}^\#)$ 
14:       $\mathbb{G} = \mathbb{G} \cup \mathcal{G}_i$ 
15:       $\hat{\mathbb{M}} = \hat{\mathbb{M}} \cup \hat{\mathcal{M}}$ 
16:    else
17:      Add the child nodes of  $\hat{\mathcal{M}}$  to  $Q$ 
18:    end if
19:  end while
20:   $\hat{\mathcal{M}} \leftarrow$  an abstract model containing only the fluents contained in the models in  $\hat{\mathbb{M}}$ 
21:  return  $\langle \mathbb{G}, \hat{\mathcal{M}} \rangle$ 
22: end procedure
```

For completeness, it must be shown that the search will find a solution if one exists. This follows from the fact that it is testing every possible model for a solution. $\square$

Note that the above theorem only holds when the lattice contains an abstract model for every possible relevant abstraction (i.e., it is complete) and there are edges for each potential proposition that could be introduced to the node (i.e., proposition conserving). Such a lattice might be exponential in the number of fluents in $\mathbb{F}$ and in many cases, projecting out all fluents except the ones in the original goal (i.e., $\mathbb{F} = \mathcal{F} \setminus \mathcal{G}_H^{\mathcal{F}}$) may need to be considered. As such, performing an exhaustive search over this space may not be a feasible approach.

4.1 Planning Compilation

Instead, the focus will be on the use of a planning-based compilation to directly identify the goal specification and abstraction set. Given the focus on classical planning problems, there are a number of efficient optimal planners available, which can be leveraged. As is able to be seen in the empirical results, the advantage of using an existing optimal planner more than makes up for the overhead of planning directly in the complex concrete node $\mathcal{M}^\#$.

The compilation consists of adding a pair of actions for each fluent not part of the initial state that either check if the fluent is true or false in the goal state, following the execution of a plan. An updated goal specification requires at least one action per fluent to be executed. A fluent being true in the goal state has a higher cost than a fluent being false. In this way, the planner will then find a plan that adds the least number of new fluents to the final state. To ensure that the check is only performed at the final state, an execution flag *can_exec* is also used, which will be part of the preconditions of all the original actions and needs to be false for the checks to be performed.

More formally, for the agent model $\mathcal{M}^\# = \langle \mathcal{F}^\#, \mathcal{A}^\#, \mathcal{I}^\#, \mathcal{G}_H \rangle$ the new compiled model is given as $\mathcal{M}' = \langle \mathcal{F}', \mathcal{A}', \mathcal{I}', \mathcal{G}' \rangle$, where

- $\mathcal{F}' = \mathcal{F}^\# \cup \mathcal{F}^{checks} \cup \{can_exec\}$

- $\mathcal{F}^{checks} = \{checked_f : f \in (\mathcal{F} \setminus (I^\# \cup \mathcal{G}_H^\mathcal{F}))\}$, contains a fluent to track whether the state of the corresponding fluent f not in initial state has been checked in the goal state.
- $\mathcal{A}' = \hat{\mathcal{A}}^\# \cup \mathcal{A}^{check} \cup \{flip_exec\}$
- $\mathcal{I}' = \mathcal{I} \cup \{can_exec\}$
- $\mathcal{G}' = \mathcal{G}_H \bigwedge_{f \in \mathcal{F}^{checks}} f$
- $\hat{\mathcal{A}}^\# = \{\hat{a} : a \in \mathcal{A}\}$, where the precondition of the action has been modified to contain *can_exec* fluent
 - $pre'(\hat{a}) = pre^\#(a) \wedge \{can_exec\}$. Here, $pre^\#(a)$ is the precondition of the action in the original model $\mathcal{M}^\#$.
- $\mathcal{A}^{check} = \{a_f^{true} : f \in \mathcal{F}\} \cup \{a_f^{false} : f \in \mathcal{F}\}$ where:
 - $pre'(a_f^{true}) = f \wedge \neg can_exec$
 - $add(a_f^{true}) = \{checked_f\}$
 - $c(a_f^{true}) > |S^\#| * C_{max}$, where C_{max} is the cost of the most costly action in $\mathcal{M}^\#$ and $|S^\#|$ is the number unique states possible under $\mathcal{F}^\#$.
 - $pre(a_f^{false}) = \neg f \wedge \neg can_exec$
 - $add(a_f^{false}) = \{checked_f\}$
 - $c(a_f^{false}) = 1$
- *flip_exec* has a single precondition *can_exec*, and a single delete effect that removes that fluent.

It is worth noting that, per the definition, the cost of doing a single a_f^{true} is greater than the cost of the most costly plan possible in $\mathcal{M}^\#$. As such, any optimal planner will try to avoid using a single one of the true check actions when possible. Now this leads to the first proposition connecting this compilation to the earlier procedure.

Proposition 2. *For an optimal plan π' for the compiled model $\mathcal{M}'$, the fluent list $\delta(\mathcal{I}', \pi', \mathcal{M}') \setminus \mathcal{I}'$ corresponds to a goal specification that could be identified by Algorithm 1.*

The result follows from the fact that the optimal plan would add the smallest number of possible fluents to the goal state. This would correspond to a minimal goal condition identified by the algorithm, as no subset of it would correspond to a valid solution, else the plan wouldn't have been optimal.

Now, the above result only shows that a possible entry from the output of Algorithm 1 can be found. A way to find all possible entries would be desirable. The way this will be done is by rerunning the compilation while disallowing previously identified descriptions. To do this, a new fluent *checked_disallow* is introduced, which will now be part of the goal specification. Here, a new disallow check action a_{dis} is used, which for a set of already identified goal specifications $\hat{\mathbb{G}}$, is given as:

- $pre(a^{dis}) = \bigwedge_{\mathcal{G} \in \hat{\mathbb{G}}} \neg \mathcal{G} \wedge \neg can_exec$
- $add(a^{dis}) = \{checked_disallow\}$

Proposition 3. *For a set of already identified goal specifications $\hat{\mathbb{G}}$, the above restriction will never prevent the compilation from identifying a goal specification $\mathcal{G}'$, where $\hat{\mathbb{G}} \cup \{\mathcal{G}'\}$ could be part of minimal covering solution.*

This follows from the fact that the disallow condition basically disallows the new goal state from either containing the exact same set of propositions as a previously found one or a superset of it. This is exactly the same condition placed on goal specifications for it to be part of a minimally covering solution.

Now the new procedure would be to repeatedly call this compilation until no more plans can be found. The list of goals found this way becomes the $\mathbb{G}$ component of the solution, while the target model would be the most abstract model that only contains fluents that appear in $\mathbb{G}$. This procedure of iteratively calling the compilation while disallowing previously found solutions until unsolvability, and then identifying the target model from the union of those goals, is called the

compilation approach. This leads to the final result, which shows that the compilation approach is equivalent to the earlier exhaustive search.

Theorem 2. *The compilation approach is a sound and complete approach for finding a minimally covering, maximally abstract solution to GRP, provided a sound and complete optimal planner is being used.*

Proof Sketch. Here, Proposition 2 shows that the compilation can identify a goal specification that could be part of a minimally covering maximally abstract solution. Proposition 3 shows that it will never disallow a potential candidate that is part of a minimally covering solution that contains the previous solution. Now all that must be ensured is that it finds the next goals that contain the minimal number of fluents. This is again guaranteed by the fact that an optimal planner is being used. Now, the next part to be proved is that the model identified by finding the most abstract model that only contains the fluents in the identified goal specifications, corresponds to a maximally abstract one. First off, the model needs to contain all the fluents that are part of the goal specification. Otherwise, it will either not satisfy condition C2 from Definition 3, or the goal specification contains redundant fluents, in which case, it can't be part of a minimally covering solution. This means no models can be considered that are more abstract than that. Now the question would be, is it ever needed to consider models that are less abstract than that? Again, the answer is no. As per the definition of abstraction, all plan validity is conserved via abstraction. Additionally, the abstraction operator will preserve the status of all fluents that are not projected out. As such, both conditions C1 and C2 should hold in the model for all goals $\mathcal{G}_i$ that are part of the set identified by this method. This proves the soundness of the method.

As long as a complete planner is used, which will always return a plan if one exists, every potential candidate not part of the previously identified solution can be exhaustively tested for. This means that this approach would also be a complete procedure for identifying minimally covering, maximally abstract solutions to GRP. □

Chapter 5

Evaluation

To evaluate the method, a set of computational experiments are performed, and a user study was conducted. The goal of the computational experiments was to compare the effectiveness of the two proposed methods, and with the user study, to test the effectiveness of presenting a covering goal set and a target model as a way to help users choose their desired goals.

5.1 Computational Experiments

Computational experiments were performed on different International Planning Competition (IPC) [19] domains with increasing numbers of fluents, to determine the time the exhaustive search and the planning compilation-based method would take to identify a minimally covering, maximally abstract solution. These experiments were performed on a machine using a 12-core 12th Gen Intel i7-12700K processor running at 3.6 GHz, using 64 GB of RAM, with an Alma Linux distribution for the operating system. The results of these experiments, along with details on the domains and instance sizes, can be seen in Tables 5.1 and 5.2. Optimal plans were found using Fast Downward 24.06 using A* search [20, 21] with LM-cut heuristic [22]. Implementation of the algorithm and compilation as described here were done in Python, using Python 3.10. The implementation made extensive use of the Unified Planning [23] package. All problems were fully grounded. A comparison of this method against a condition where the compilation was solved using a satisficing planner was also performed. These results of this experiment are found in Table 5.3. While such a planner can no longer guarantee minimality of goal set or the model being maximally abstract, the experiment checked if the use of satisficing planners gave a significant speed up. While a speedup was observed, and timeouts were eliminated, the lack of optimality also caused slowdowns for some domains.

For each IPC domain, five different instances of the problem were selected with increasing actions and fluents. For each instance, five different copies of the problem were created, with

each having a goal state obtained by taking a random subset of the propositions of the problem. A solvability test was performed on each random instance to make sure that there were valid plans for the instances. All listed solution sizes and time taken were identified by averaging across these five random instances. Unfortunately, for the largest problem instance from Sokoban, more than one random solvable goal specification could not be found, so for that problem, only the information regarding that single instance is listed.

5.2 Discussion

In Table 5.1, the complexity of each problem instance is characterized in terms of the number of grounded fluents (F), of grounded actions A and the time taken to solve the original problem in seconds $tN(s)$. For the compiled problem, the added complexity is presented in terms of the number of total grounded fluents in the compiled problem (CF) and the number of actions (CA). The primary metric of interest is to compare the time taken by the compilation method ($tC(s)$) and the exhaustive search ($tEX(s)$). These results can be found in Table 5.2. The size of the identified solution ($numS$) is also listed, specifically in terms of the number of goal specifications found, when one of the approaches is able to complete its run.

The results largely follow the expected outcomes. It can be seen that the exhaustive search method is only successful on the very smallest of problems, in the very simplest of domains, like Elevator and Gripper. Also of note is the fact that many of the IPC domains tested (Driverlog, Elevator, and Gripper) do not produce more than one set of propositions which are true in the final state. This may be an aspect of these domains, as others (Sokoban and Trucks) have more unique goal-states produced.

Overall, the time used by the compilation was more than a single Fast Downward plan being generated, as is expected, but the times themselves are not excessively long. Only the largest instances of Rover, Sokoban, Trucks, and Blocksworld provided issues with timeouts, but some moderately sized problems in domains that are generally considered harder planning domains (like Sokoban (cf. [24])) still performed well with the compilation.

Table 5.1: Description and Results of Problems from IPC Domains using an Optimal Planner. Results of 5 trials in the form of (average $\pm$ standard deviation). F = Number of Fluents in Original Problem, A = Number of actions in the original grounded model, CF = Number of fluents in the compiled model, CA = Number of actions in the compiled model, $tN(s)$ = Time needed to solve the original problem in seconds. The largest Sokoban instance only used one copy.

Domain	F	A	CF	CA	$tN(s)$
Blocksworld	29	40	48	75	0.432 ± 0.104
	41	60	73	121	0.676 ± 0.0745
	71	112	129	225	4.686 ± 0.0210
	109	180	202	363	4.908 ± 0.2750
	131	220	224	443	5.490 ± 1.1525
Driverlog	32	88	56	133	0.441 ± 0.0428
	39	108	64	155	0.552 ± 0.0952
	44	120	76	181	0.510 ± 0.0276
	49	144	83	209	0.653 ± 0.1095
	57	168	97	245	0.717 ± 0.0673
Elevator	4	4	8	9	0.249 ± 0.0686
	8	16	15	27	0.239 ± 0.0181
	12	36	22	53	0.278 ± 0.0472
	16	64	29	87	0.321 ± 0.0241
	20	100	36	129	0.404 ± 0.0382
Gripper	20	34	31	53	0.307 ± 0.0309
	28	50	56	77	0.371 ± 0.0147
	36	66	55	101	0.448 ± 0.0309
	44	82	67	125	1.013 ± 0.0271
	52	98	79	149	4.450 ± 0.3219
Rover	35	63	59	108	0.387 ± 0.0263
	28	53	49	92	0.349 ± 0.0106
	44	76	76	137	0.495 ± 0.1351
	50	86	88	159	0.473 ± 0.0262
	59	144	101	225	1.249 ± 1.2813
Sokoban	86	114	149	237	4.276 ± 0.0144
	72	102	125	205	4.241 ± 0.0753
	62	78	108	167	4.193 ± 0.0486
	134	202	239	409	4.874 ± 0.5711
	368	888	694	1537	9.100**
Trucks	91	261	173	422	0.996 ± 0.0409
	117	336	223	545	1.222 ± 0.0471
	191	789	369	1142	2.587 ± 0.0457
	226	936	437	1355	3.237 ± 0.1328
	327	1794	637	2411	6.187 ± 0.4191

Table 5.2: Description and Results of Problems from IPC Domains using an Optimal Planner. Results of 5 trials in the form of (average $\pm$ standard deviation). CF = Number of fluents in the compiled model, CA = Number of actions in the compiled model, $numS$ = Number of Sets produced, $tC(s)$ = Time taken by the procedure that uses planning compilations in seconds, $tEX(s)$ = Time taken by the exhaustive search method. Rover and Trucks had an instance where some copies of the problem timed out, and those times are not included. The largest Sokoban instance only used one copy.

Domain	CF	CA	$numS$	$tC(s)$	$tEX(s)$
Blocksworld	48	75	1 ± 0	0.599 ± 0.021	30m Timeout
	73	121	2.8 ± 1.48	2.361 ± 0.8478	30m Timeout
	129	225	2.2 ± 1.64	36.113 ± 37.4484	30m Timeout
	202	363	0	30m Timeout	30m Timeout
	224	443	0	30m Timeout	30m Timeout
Driverlog	56	133	1 ± 0	0.657 ± 0.0497	30m Timeout
	64	155	1 ± 0	1.079 ± 0.4665	30m Timeout
	76	181	1 ± 0	0.797 ± 0.1569	30m Timeout
	83	209	1 ± 0	1.196 ± 0.3889	30m Timeout
	97	245	1 ± 0	4.235 ± 5.4000	30m Timeout
Elevator	8	9	1 ± 0	0.348 ± 0.0805	1.307 ± 0.4083
	15	27	1 ± 0	0.361 ± 0.0281	9.154 ± 4.8937
	22	53	1 ± 0	0.422 ± 0.0406	116.729 ± 6.5209
	29	87	1 ± 0	0.483 ± 0.0084	1102.488 ± 123.2032
	36	129	1 ± 0	0.618 ± 0.0680	30m Timeout
Gripper	31	53	1 ± 0	0.487 ± 0.0375	186.757 ± 78.0433
	56	77	1 ± 0	0.538 ± 0.0305	30m Timeout
	55	101	1 ± 0	0.797 ± 0.0744	30m Timeout
	67	125	1 ± 0	3.972 ± 0.3537	30m Timeout
	79	149	1 ± 0	27.101 ± 2.9174	30m Timeout
Rover	59	108	1 ± 0	11.965 ± 3.0789	30m Timeout
	49	92	1 ± 0	2.238 ± 0.6940	30m Timeout
	76	137	1.4 ± 0.55	$0.638 \pm 0.0297^*$	30m Timeout
	88	159	1.6 ± 0.55	30m Timeout	30m Timeout
	101	225	2.2 ± 1.79	30m Timeout	30m Timeout
Sokoban	149	237	5.4 ± 3.36	27.208 ± 14.4601	30m Timeout
	125	205	9.8 ± 10.03	27.023 ± 13.6787	30m Timeout
	108	167	9.8 ± 10.80	26.358 ± 22.9010	30m Timeout
	239	409	7.8 ± 9.07	217.772 ± 337.8253	30m Timeout
	694	1537	0	30m Timeout	30m Timeout
Trucks	173	422	4.2 ± 5.22	5.078 ± 4.3877	30m Timeout
	223	545	8.6 ± 9.91	17.016 ± 11.3202	30m Timeout
	369	1142	5.4 ± 2.70	256.890 ± 228.6576	30m Timeout
	437	1355	2.0 ± 2.00	$1299.9 \pm 695.5^*$	30m Timeout
	637	2411	1.8 ± 4.02	30m Timeout	30m Timeout

Table 5.3: Description and Results of Problems from IPC Domains using a Satisficing Planner. Results of 5 trials in the form of (average $\pm$ standard deviation). CF = Number of fluents in the compiled model, CA = Number of actions in the compiled model, $numS$ = Number of Sets produced, tC = Time for Solving Compiled Problem, $RamC$ = Memory Usage on Compiled Problem. The largest Sokoban instance only used one copy.

Domain	CF	CA	$numS$	$tC(s)$	$RamC(MB)$
Blocksworld	48	75	1 ± 0	0.598 ± 0.0357	121.728 ± 1.2968
	73	121	4.0 ± 4.00	2.885 ± 2.2035	121.408 ± 0.7789
	129	225	3.4 ± 2.30	18.205 ± 11.3110	269.372 ± 2.9467
	202	363	29.0 ± 23.67	380.749 ± 162.9106	2384.900 ± 175.4066
	224	443	98.8 ± 137.39	4.673 ± 0.0289	236.199 ± 2.9590
Driverlog	56	133	1.4 ± 0.89	0.677 ± 0.1924	123.232 ± 0.2086
	64	155	1 ± 0	0.721 ± 0.0681	122.976 ± 242.6520
	76	181	3.4 ± 2.30	1.334 ± 0.6671	124.960 ± 0.2529
	83	209	1 ± 0	0.772 ± 0.0313	124.704 ± 537.8476
	97	245	7.8 ± 13.54	3.085 ± 4.4406	126.208 ± 1.2819
Elevator	8	9	1.4 ± 0.55	0.432 ± 0.1161	117.984 ± 0.3118
	15	27	1.4 ± 0.89	0.418 ± 0.1386	119.840 ± 0.4381
	22	53	2.0 ± 2.24	0.599 ± 0.3988	121.184 ± 0.3683
	29	87	3.4 ± 3.36	0.931 ± 0.6725	120.672 ± 0.4141
	36	129	9.0 ± 1.00	2.268 ± 0.2882	122.656 ± 0.5611
Gripper	31	53	1.4 ± 0.55	$0.524 \pm .1171$	121.440 ± 0.3752
	56	77	1.2 ± 0.45	$0.550 \pm .1003$	122.976 ± 0.2906
	55	101	1.2 ± 0.45	$0.651 \pm .1054$	123.200 ± 0.3919
	67	125	1.8 ± 0.45	$0.854 \pm .1232$	123.136 ± 0.4322
	79	149	1.8 ± 0.45	$0.947 \pm .0981$	125.024 ± 0.1824
Rover	59	108	3.0 ± 1.87	17.466 ± 25.6858	123.232 ± 0.1338
	49	92	1.6 ± 0.55	1.409 ± 0.2994	122.912 ± 0.2862
	76	137	5.0 ± 3.46	613.926 ± 573.9194	7967.141 ± 7259.5800
	88	159	4.2 ± 2.77	0.616 ± 0.2743	2822.420 ± 6035.9077
	101	225	55.6 ± 106.58	0.627 ± 0.0840	163.640 ± 82.8287
Sokoban	149	237	6.4 ± 3.21	29.850 ± 12.7868	222.808 ± 0.6542
	125	205	33.8 ± 49.58	140.269 ± 199.3421	225.517 ± 0.7866
	108	167	26.4 ± 28.40	110.117 ± 114.2140	229.338 ± 0.7189
	239	409	84.0 ± 101.36	368.241 ± 442.9987	218.185 ± 1.7621
	694	1537	223	8.177288771	229552
Trucks	173	422	7.4 ± 9.32	4.489 ± 4.2508	130.688 ± 1.6681
	223	545	18.6 ± 24.66	13.608 ± 15.8459	135.008 ± 1.9213
	369	1142	117.0 ± 243.22	18.024 ± 9.8021	211.015 ± 61.6193
	437	1355	135.2 ± 185.57	94.911 ± 114.4333	1538.269 ± 1636.5726
	637	2411	141.2 ± 164.69	16.023 ± 22.5185	915.991 ± 866.5724

In terms of memory usage, it was observed that the exhaustive search was more memory-intensive. The average memory usage for the compilation method across all instances was 218.3 MB of RAM used. The minimum was 117.8 MB for the smallest instance of Elevator, while the maximum was 3729.7 MB for the second smallest instance of Sokoban. For exhaustive search, the average was 337.9 MB of RAM used. The minimum used was 122.6 MB for the same elevator instance, and the maximum was 28628.8 MB for the third smallest instance of Sokoban.

In general, these results indicate that the methods presented here could be used over a wide range of domains and problem sizes. However, there is still room to develop more effective methods for larger problem instances.

5.3 User Study

The user study was designed to test the following hypotheses:

- H1 - Users will make fewer mistakes, in terms of including undesired side-effects, when presented with a minimally covering goal specification list along with maximally abstract model information (i.e., the solution to the corresponding goal recommendation problem), rather than having to describe preferences on all possible true fluents.
- H2 - Users will use less time to decide their desired goal specification, when provided with a minimally covering maximally abstract solution to the corresponding goal recommendation problem, rather than having to describe preferences on all possible true fluents.
- H3 - Users will prefer specifying their desired outcome via a minimally covering maximally abstract solution rather than having to describe preferences on all possible true fluents (side-effects).

The first two hypotheses deal with objective measures related to the user achieving their goal, while the third corresponds to their subjective preference of the method. The user study did not provide users the ability to pose questions or go beyond being presented the fluents in order to

limit the scope of the study and examine the efficacy of the system without further explainability incorporated.

5.3.1 User Study Experimental Setup

Participants and Recruitment

A total of 39 participants were recruited for this experiment (21 identified as male and 18 as female), the majority of whom were within the age range of 25 to 44. Eligibility required participants to be 18 or older with English proficiency. Participants were sourced from the Prolific platform and compensated through the same. IRB approval was also obtained for the study.

Design and Randomization

The study used a within-subjects assignment with two conditions. In the first condition, the user selects a desired goal specification, given a minimally covering and maximally abstract solution for the corresponding goal recommendation problem. In the second, they are provided the list of all fluents and they can decide which ones they want to be present in the goal states and which they do not. Model information is provided to the user in terms of how a plan corresponding to their specification would evolve. Each participant was exposed to both conditions via two scenarios: one scenario involved a laundromat, where the robot should make sure to separate the different types of clothes, and the other scenario involved a changing light bulbs, where the robot should avoid breaking other bulbs. For both conditions and scenarios, participants were asked to complete their goal selection within four minutes. The survey would force them to move to the next page once a timer reached four minutes. The order in which participants viewed the scenarios was randomized to support counterbalancing. This study was single-blind in that participants were unaware that multiple versions of their condition existed. The study was designed to last 17 minutes and had an average completion time of 13 minutes. The complete text of the user study can be found in Appendix A.

Metrics

The primary metrics used were the correctness of the specifications chosen by the participants, the time taken, and their reported preference for the different specification mechanisms. Information was also collected on subjective cognitive load measures through a NASA TLX questionnaire [25]. These questions were based on a Likert scale between 0-20, inclusive. A Likert scale between 0-10 was also used to measure how easy the users found each interface.

Data Quality and Exclusions

Participants who failed the attention checks were not compensated, and their data were excluded. Repeat participation was also not allowed, and any data from repeat participation was excluded. Missing data was not an issue as the questionnaire required responses. Users that did not correctly respond to the tasks had times considered invalid, and were thus not used in comparisons testing H2. Although unique Prolific IDs were relied upon, IP addresses were collected and could be cross-referenced if necessary.

5.3.2 User Study Details

Here, details about the user study are described. The user first begins by completing a consent form, with the end of the form shown in Figure 5.1.

Once users complete the consent form, they are presented with information about the first scenario, as shown in Figure 5.2. Following this page, they see either a list of fluents they must make decisions on, or sets of fluents to choose between. Figure 5.3 shows the first scenario, the timer provided to the users, and some of the fluents they chose between. The entire list of fluents is shown in Figure 5.4. Figure 5.5 shows an update that is provided to a user when they have made a selection, but before they have moved on. Users were allowed to change their choices, and Figure 5.6 shows a different choice made, with a different explanation provided as to why the selection does not lead to a valid plan. Figure 5.7 shows the plan provided to a user when they have selected a valid plan; however, this plan is possible in the model, but may not be the correct plan to achieve everything desired. For example, the first scenario of the laundry robot could result

complete the entire study.

Who to contact

For questions regarding the rights of research subjects, any complaints or comments regarding the manner in which the study is being conducted, contact the CSU Institutional Review Board at: CSU_IRB@colostate.edu.

Clicking on the next button below implies your consent for data on your responses to be collected.

If you wish to participate in this research, please provide your Prolific ID and click the next button. Otherwise, please exit the survey.



Figure 5.1: User study screenshot showing end of IRB consent form.

in clothing being ruined because it was not separated. In Figure 5.7, the plan correctly separates the clothing. Table 5.4 shows the list of propositions for this scenario, with Table 5.5 showing the explanations that were displayed when a plan was not possible and Table 5.6 showing the plans and explanations that were displayed when a plan was possible. Following the first scenario, users are presented with the NASA TLX questions, shown in Figures 5.8 and 5.9.

Table 5.4: Propositions used in the Laundry Robot scenario of the User Study

Proposition	Description
P1	All of the clothing was washed.
P2	The light clothing was separated from the dark clothing.
P3	The clothing was washed in one load.
P4	The clothing was washed in two loads.
P5	The light clothing was washed.
P6	The dark clothing was washed.

Table 5.5: Explanations displayed in the Laundry Robot scenario of the User Study when no plan is possible. The rules are evaluated from top to bottom, with the explanation of the first rule found to be true is displayed.

Propositions Selected	Explanation Displayed
$\neg P1 \vee \neg P5 \vee \neg P6$	No sequence of actions to achieve the goal is possible from your selection. Reason: All clothing being washed is part of the goal, which is required. In addition, both the light and dark clothing must be washed for "All of the clothing is washed" to be true.
$P2 \wedge P3$	No sequence of actions to achieve the goal is possible from your selection. Reason: The light clothing being separated from the dark clothing requires two loads to wash. It cannot be selected at the same time as "The clothing was washed in one load."
$\neg P2 \wedge \neg P3$	No sequence of actions to achieve the goal is possible from your selection. Reason: The light clothing being separated from the dark clothing requires two loads to wash. It cannot be Not Achieved (False) at the same time as "The clothing was washed in one load" is also skipped.
$\neg P2 \wedge P4$	No sequence of actions to achieve the goal is possible from your selection. Reason: The light clothing being separated from the dark clothing is required for two loads to wash. It must be selected at the same time as "The clothing was washed in two loads."
$P2 \wedge \neg P4$	No sequence of actions to achieve the goal is possible from your selection. Reason: The light clothing being separated from the dark clothing is required for two loads to wash. It must be selected at the same time as "The clothing was washed in two loads."
$P3 \wedge P4$	No sequence of actions to achieve the goal is possible from your selection. Reason: The selection of "The clothing was washed in one load" cannot be simultaneously Achieved (True) or simultaneously Not Achieved (False) with "The clothing was washed in two loads." Exactly one of these two must be Achieved (True).
$\neg P3 \wedge \neg P4$	No sequence of actions to achieve the goal is possible from your selection. Reason: The selection of "The clothing was washed in one load" cannot be simultaneously Achieved (True) or simultaneously Not Achieved (False) with "The clothing was washed in two loads." Exactly one of these two must be Achieved (True).

Table 5.6: Explanations displayed in the Laundry Robot scenario of the User Study when a plan is possible.

Propositions Selected	Explanation Displayed	
$P1 \wedge \neg P2 \wedge P3 \wedge P5 \wedge P6$	Sequence of actions to achieve the goal from your selections:	
	Action Step	Reasons for Taking This Action
	1. Wash all clothes	This makes "All of the clothing was washed" Achieved (True), and also makes "The clothing was washed in one load" Achieved (True). Because "All of the clothing was washed" is Achieved (True), it must also be Achieved (True) that "The light clothing was washed" and "The dark clothing was washed." Both of these are also made Achieved (True) with this step.
$P1 \wedge P2 \wedge P4 \wedge P5 \wedge P6$	Sequence of actions to achieve the goal from your selections:	
	Action Step	Reasons for Taking This Action
	1. Separate light and dark clothing	This makes "The light clothing was separated from the dark clothing" Achieved (True). Because "All of the clothing was washed" is Achieved (True), it must also be Achieved (True) that "The light clothing was washed" and "The dark clothing was washed."
	2. Wash light clothing	This makes "The light clothing was washed" Achieved (True).
	3. Wash dark clothing	This makes "All of the clothing was washed" Achieved (True), and also makes "The clothing was washed in two loads" Achieved (True). This also makes "The dark clothing was washed" Achieved (True).

For this scenario, consider a laundromat using a robot to wash the laundry for you. The goal is to wash all clothing. You are trying to wash both light and dark clothing, and washing the light and dark clothing together will cause the light colors to become discolored - this ruins the clothing, making it unusable.

A timer of 4 minutes will start when you begin the question. If you do not finish within 4 minutes, the survey will move forward automatically.

Here, your goal is to help the robot to complete the task without ruining the clothes. You will be presented one of two ways to do it. You will either choose between sets of outcomes, or indicate for each statement whether it is an outcome you want ("Achieved" or "True") or not ("Not Achieved" or "False"). For the list of statements, some possible combinations of choices may be unachievable. For both the sets of statements and the list of statements, explanations will appear below the choices as you make them and can be read before you proceed if you would like to change your selections.

Figure 5.2: User study screenshot showing details presented to the user about the first scenario.

Upon completion of the first scenario, users are presented with information about the second scenario, as shown in Figure 5.10. They are then provided with the second scenario, which contains the other method than the one they received for the first scenario. In this case, the first scenario used a list of fluents that required preference for each fluent, so the second scenario uses sets of fluents that the user chooses between. The scenario and the sets of fluents are shown in Figure 5.11. Upon selection of either choice, they are shown a plan for their choice, with corresponding explanations, as shown in Figures 5.12 and 5.13 for the first and second sets, respectively. Table 5.7 shows the list of propositions for this scenario, with Table 5.8 showing the explanations that were displayed when a plan was not possible and Table 5.9 showing the plans and explanations that were displayed when a plan was possible. After this, another series of NASA TLX questions

0347

Given the following statements for the scenario as described, please mark whether this statement should be Achieved (True) or Not Achieved (False) at the end of the scenario, when the robot has finished washing the laundry. Please provide an answer for each statement.

As you make selections, the actions the robot will take (with reasons for taking each action) will be shown to you below the choices, or the reasons why no steps can be generated. You may change your selections as many times as you wish, and the feedback will update based on your choices. Only your answers when you press the Next button, or when the timer expires, will be recorded.

	Achieved (True)	Not Achieved (False)
All of the clothing was washed.	☐	☐
The light clothing was separated from the dark clothing.	☐	☐
The clothing was washed in one load.	☐	☐
The clothing was washed in two loads.	☐	☐

Figure 5.3: User study screenshot showing the first scenario, with the user presented with a list of fluents.

are asked of the user. Then, they are asked questions to compare the methods, as shown in Figure 5.14, and demographic questions, as shown in Figure 5.15.

Users were randomly assigned between the two groups, either receiving the list of fluents for the first scenario or the sets of fluents for the first scenario. Figures 5.16, 5.17, and 5.18 show the first scenario with the sets of fluents, and Figures 5.19, 5.20, 5.21, and 5.22 show the second scenario with the list of fluents.

robot has finished washing the laundry. Please provide an answer for each statement.

As you make selections, the actions the robot will take (with reasons for taking each action) will be shown to you below the choices, or the reasons why no steps can be generated. You may change your selections as many times as you wish, and the feedback will update based on your choices. Only your answers when you press the Next button, or when the timer expires, will be recorded.

	Achieved (True)	Not Achieved (False)
All of the clothing was washed.	☐	☐
The light clothing was separated from the dark clothing.	☐	☐
The clothing was washed in one load.	☐	☐
The clothing was washed in two loads.	☐	☐
The light clothing was washed.	☐	☐
The dark clothing was washed.	☐	☐

→

Figure 5.4: User study screenshot showing first scenario, showing the entire list of fluents that the user must give preference about.

5.3.3 User Study Results

Regarding the first hypothesis, it was found that 10 participants failed when provided with the list of fluents, while only a single participant failed when provided with the recommendation problem solution. As such, the result was a failure rate of 28% for the former and failure rate of 2.5% for the latter. This provides strong evidence for the hypothesis H1 holding. It is worth noting that, in addition to users who provided the wrong specification (in that they provided an undesired

when you press the Next button, or when the timer expires, will be recorded.

	Achieved (True)	Not Achieved (False)
All of the clothing was washed.	☒	☐
The light clothing was separated from the dark clothing.	☒	☐
The clothing was washed in one load.	☒	☐
The clothing was washed in two loads.	☒	☐
The light clothing was washed.	☒	☐
The dark clothing was washed.	☒	☐

No sequence of actions to achieve the goal is possible from your selection.

Reason: The light clothing being separated from the dark clothing requires two loads to wash. It cannot be selected at the same time as "The clothing was washed in one load."



Figure 5.5: User study screenshot showing first scenario, with the explanation provided to the user when they have chosen a goal that is not possible.

outcome), timing out of the task was also counted as a failure. However, the timeout only counted for 27% of all failures.

Regarding hypothesis H2, the time to complete any task when choosing for each fluent on a list of fluents was 106.9 ± 55.55 seconds (given as average $\pm$ standard deviation) compared to the time when provided the goal recommendation (77.1 ± 41.82 seconds). An independent-samples t-test showed this to be a statistically significant difference, with a p-value of 0.02. This provides strong evidence in support of H2.

when you press the Next button, or when the timer expires, will be recorded.

	Achieved (True)	Not Achieved (False)
All of the clothing was washed.	☒	☐
The light clothing was separated from the dark clothing.	☒	☐
The clothing was washed in one load.	☒	☐
The clothing was washed in two loads.	☒	☐
The light clothing was washed.	☐	☒
The dark clothing was washed.	☒	☐

No sequence of actions to achieve the goal is possible from your selection.

Reason: All clothing being washed is part of the goal, which is required. In addition, both the light and dark clothing must be washed for "All of the clothing is washed" to be true.

→

Figure 5.6: User study screenshot showing showing first scenario, with the explanation provided to the user when they have chosen a goal that is not possible. This explanation differs from the explanation when all are True.

When considering the subjective measures, somewhat mixed results are seen. To start with, the reported temporal and mental demand ratings across the two conditions showed no statistically significant differences. However, both measures were marginally higher for lists of fluents, indicating that participants tended to experience greater temporal pressure and mental workload when engaging with list conditions. Similarly, only 17 people preferred using goal sets over using the full fluent list, while 12 people reported preferring the fluent list. Additionally, users rated lists

	Achieved (True)	Not Achieved (False)
All of the clothing was washed.	☒	☐
The light clothing was separated from the dark clothing.	☒	☐
The clothing was washed in one load.	☐	☒
The clothing was washed in two loads.	☒	☐
The light clothing was washed.	☒	☐
The dark clothing was washed.	☒	☐

Sequence of actions to achieve the goal from your selections:

Action Step	Reasons for Taking this Action
1. Separate light and dark clothing	This makes "The light clothing was separated from the dark clothing" Achieved (True). Because "All of the clothing was washed" is Achieved (True), it must also be Achieved (True) that "The light clothing was washed" and "The dark clothing was washed."
2. Wash light clothing	This makes "The light clothing was washed" Achieved (True).
3. Wash dark clothing	This makes "All of the clothing was washed" Achieved (True), and also makes "The clothing was washed in two loads" Achieved (True). This also makes "The dark clothing was washed" Achieved (True).

Figure 5.7: User study screenshot showing showing first scenario, with the plan provided to the user that achieves their selected goals.

Please use the sliders below to answer each of the provided questions about the previous scenario.

How much mental and perceptual activity, was required (eg., thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?

Low
0

High
20

Mental Demand



How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

Low
0

High
20

Physical Demand



How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

Low
0

High
20

Temporal Demand

Figure 5.8: User study screenshot showing the NASA TLX questions asked of the user following each scenario.

For the following pair-wise comparisons of the six scales (Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, Frustration) choose the one that contributed more to the workload of this survey.

Effort or Performance?

Effort ☐	Performance ☐
---------------------------------	--------------------------------------

Temporal Demand or Frustration?

Temporal Demand ☐	Frustration ☐
--	--------------------------------------

Temporal Demand or Effort?

Temporal Demand ☐	Effort ☐
--	---------------------------------

Physical Demand or Frustration?

Physical Demand ☐	Frustration ☐
--	--------------------------------------

Performance or Frustration?

Figure 5.9: User study screenshot showing the pair-wise NASA TLX questions asked of the user following each scenario.

Table 5.7: Propositions used in the Light Bulb Robot scenario of the User Study

Proposition	Description
P1	Two new bulbs were in the compartment when the robot ascended the first time.
P2	One new bulb was in the compartment when the robot ascended the first time.
P3	The first light bulb was changed.
P4	The second light bulb was changed.
P5	The first old bulb was dropped.
P6	One new bulb was in the compartment when the robot ascended the second time.
P7	The first old light bulb was returned.

as slightly easier to work with than sets of goals by a small margin (average of 7.5 vs 7.286). As such, there is not any strong evidence to support the final hypothesis. These results may indicate different subjective interpretations of "ease-of-use" compared to the actual effort needed. It may also be the result of using simple domains containing only a handful of fluents, and the results may be different if the user study included much more complex domains.

Table 5.8: Explanations displayed in the Light Bulb Robot scenario of the User Study when no plan is possible. The rules are evaluated from top to bottom, with the explanation of the first rule found to be true is displayed.

Propositions Selected	Explanation Displayed
$P1 \wedge P2$	No sequence of actions to achieve the goal is possible from your selection. Reason: The statement "Two new bulbs were in the compartment when the robot ascended the first time" cannot be Achieved (True) at the same time as the statement "One new bulb was in the compartment when the robot ascended the first time" is Achieved (True).
$P1 \wedge P6$	No sequence of actions to achieve the goal is possible from your selection. Reason: The statement "Two new bulbs were in the compartment when the robot ascended the first time" cannot be Achieved (True) at the same time as the statement "One new bulb was in the compartment when the robot ascended the second time" is Achieved (True).
$\neg P1 \wedge \neg P2$	No sequence of actions to achieve the goal is possible from your selection. Reason: The statement "Two new bulbs were in the compartment when the robot ascended the first time" cannot be Not Achieved (False) at the same time as the statement "One new bulb was in the compartment when the robot ascended the first time" is Not Achieved (False).
$\neg P1 \wedge \neg P6$	No sequence of actions to achieve the goal is possible from your selection. Reason: The statement "Two new bulbs were in the compartment when the robot ascended the first time" cannot be Not Achieved (False) at the same time as the statement "One new bulb was in the compartment when the robot ascended the second time" is Not Achieved (False).
$P5 \wedge P7$	No sequence of actions to achieve the goal is possible from your selection. Reason: "The first old light bulb was returned" and "The first old bulb was dropped" cannot simultaneously both be Achieved (True).
$\neg P5 \wedge \neg P7$	No sequence of actions to achieve the goal is possible from your selection. Reason: "The first old light bulb was returned" and "The first old bulb was dropped" cannot simultaneously both be Not Achieved (False).
$P2 \wedge P5$	No sequence of actions to achieve the goal is possible from your selection. Reason: "One new bulb was in the compartment when the robot ascended the first time" and "The first old bulb was dropped" cannot both be simultaneously Achieved (True).
$\neg P2 \wedge \neg P5$	No sequence of actions to achieve the goal is possible from your selection. Reason: "One new bulb was in the compartment when the robot ascended the first time" and "The first old bulb was dropped" cannot both be simultaneously Not Achieved (False).
$\neg P3 \vee \neg P4$	No sequence of actions to achieve the goal is possible from your selection. Reason: Both light bulbs must be changed, as this is the goal.

Table 5.9: Explanations displayed in the Light Bulb Robot scenario of the User Study when a plan is possible.

Propositions Selected	Explanation Displayed	
$P1 \wedge \neg P2 \wedge P3 \wedge P4 \wedge P5 \wedge \neg P6 \wedge \neg P7$	Sequence of actions to achieve the goal from your selections:	
	Action Step	Reasons for Taking This Action
	1. Bring two new light bulbs up.	This makes "Two new bulbs were in the compartment when the robot ascended the first time" Achieved (True). This leaves zero empty spaces in the compartment.
	2. Change first light bulb with zero empty spaces in the compartment.	This makes "The first light bulb was changed" Achieved (True). The bulb is dropped because two bulbs are in the compartment, not allowing the robot to place the old bulb in the compartment before removing the new one. This makes "The first old bulb was dropped" Achieved (True).
$\neg P1 \wedge P2 \wedge P3 \wedge P4 \wedge \neg P5 \wedge P6 \wedge P7$	3. Change second light bulb with one empty space in the compartment.	This makes "The second light bulb was changed" Achieved (True). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.
	Sequence of actions to achieve the goal from your selections:	
	Action Step	Reasons for Taking This Action
	1. Bring one new light bulb up.	This makes "One new bulb was in the compartment when the robot ascended the first time" Achieved (True). This leaves one empty space in the compartment
	2. Change first light bulb with one empty space in the compartment.	This makes "The first light bulb was changed" Achieved (True). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.
	3. Return first old light bulb.	This makes "The first old light bulb was returned" Achieved (True). This leaves two empty spaces in the compartment.
	4. Bring one new light bulb up.	This makes "One new bulb was in the compartment when the robot ascended the second time" Achieved (True). This leaves one empty space in the compartment.
	5. Change second light bulb with one empty space in the compartment.	This makes "The second light bulb was changed" Achieved (True). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.

For this scenario, consider a robot that is changing light bulbs in out-of-reach places. It has one arm, and can store two light bulbs in a compartment at any time. The robot can only move when the arm is not holding a bulb. The goal is to change two bulbs that are close to each other. If a light bulb is dropped, there is a good chance that the bulb will shatter – this will not only ruin the bulb, but cause shards of glass to be scattered everywhere. These shards are both a health hazard, and require someone to clean them up.

A timer of 4 minutes will start when you begin the question. If you do not finish within 4 minutes, the survey will move forward automatically.

Here, your goal is to help the robot to complete the task without shattering any bulbs. You will be presented one of two ways to do it. You will either choose between sets of outcomes, or indicate for each statement whether it is an outcome you want (“Achieved” or “True”) or not (“Not Achieved” or “False”). For the list of statements, some possible combinations of choices may be unachievable. For both the sets of statements and the list of statements, explanations will appear below the choices as you make them and can be read before you proceed if you would like to change your selections.



Figure 5.10: User study screenshot showing details presented to the user about the second scenario.

0349

Given the following sets of statements for the scenario as described, choose the set of statements that, if they are outcomes of the actions of the robot ("Achieved" or "True") after the robot has finished every step in its plan, would best accomplish your goals.

As you make selections, the actions to achieve the goal (with reasons for taking each action) will be shown to you below the choices, or the reasons why no steps to achieve the goal can be generated. You may change your selections as many times as you wish, and the feedback will update based on your choices. Only your answers when you press the Next button, or when the timer expires, will be recorded.

Set 1:

Two new bulbs were in the compartment when the robot ascended the first time.
The first light bulb was changed.
The second light bulb was changed.
The first old bulb was dropped.

Set 2:

One new bulb was in the compartment when the robot ascended the first time.
The first light bulb was changed.
The second light bulb was changed.
One new bulb was in the compartment when the robot ascended the second time.
The first old light bulb was returned.

☐ Set 1

☐ Set 2

Figure 5.11: User study screenshot showing the second scenario, with the user presented with two sets of fluents to choose between.

The first old light bulb was returned.

☒ Set 1

☐ Set 2

Sequence of actions to achieve the goal from your selection:

Action Step	Reasons for Taking this Action
1. Bring two new light bulbs up.	This makes "Two new bulbs were in the compartment when the robot ascended the first time" Achieved (True). This leaves zero empty spaces in the compartment
2. Change first light bulb with zero empty spaces in the compartment.	This makes "The first light bulb was changed" Achieved (True). The bulb is dropped because two bulbs are in the compartment, not allowing the robot to place the old bulb in the compartment before removing the new one. This makes "The first old bulb was dropped" Achieved (True).
3. Change second light bulb with one empty space in the compartment.	This makes "The second light bulb was changed" Achieved (True). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.



Figure 5.12: User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected set of goals.

Set 1
Set 2

Steps to achieve the goal from your selection:

Action Step	Reasons for Taking this Action
1. Bring one new light bulb up.	This makes "One new bulb was in the compartment when the robot ascended the first time" Achieved (True). This leaves one empty space in the compartment
2. Change first light bulb with one empty space in the compartment.	This makes "The first light bulb was changed" Achieved (True). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.
3. Return first old light bulb.	This makes "The first old light bulb was returned" Achieved (True). This leaves two empty spaces in the compartment
4. Bring one new light bulb up.	This makes "One new bulb was in the compartment when the robot ascended the second time" Achieved (True). This leaves one empty space in the compartment
5. Change second light bulb with one empty space in the compartment.	This makes "The second light bulb was changed" Achieved (True). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.

Figure 5.13: User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected set of goals. This shows a different plan from the different selection.

Please rate (from 0-10, with 0 being the lowest) your ease of use of the system when shown sets of statements that would be achieved (true) in a goal state and choosing between the sets.

0 1 2 3 4 5 6 7 8 9 10

Ease with Sets of Statements

☐

Please rate (from 0-10, with 0 being the lowest) your ease of use of the system when shown all statements that would be achieved (true) in a goal state and providing a choice of preference for each statement.

0 1 2 3 4 5 6 7 8 9 10

Ease with List of All Statements

☐

Between the two methods (being shown sets of statements and choosing between them vs. list of statements and giving choice of preference for each statement), which do you prefer?

☐ Sets of Statements

☐ All statements with preference for each

Why did you prefer one of the methods over the other?

Figure 5.14: User study screenshot showing final questions presented to the user where a user answers questions comparing the two methods.

What is your gender?

☐ Male

☐ Female

☐ Non-binary / third gender

☐ Prefer not to say

What range does your age fall in?

☐ Under 18

☐ 18 - 24

☐ 25 - 34

☐ 35 - 44

☐ 45 - 54

☐ 55 - 64

☐ 65 - 74

☐ 75 - 84

☐ 85 or older



Figure 5.15: User study screenshot showing demographic questions presented to the user.



Given the following sets of statements for the scenario as described, choose the set of statements that, if they are outcomes of the actions of the robot ("Achieved" or "True") after the robot has finished every step in its plan, would best accomplish your goals.

As you make selections, the actions to achieve the goal (with reasons for taking each action) will be shown to you below the choices, or the reasons why no steps to achieve the goal can be generated. You may change your selections as many times as you wish, and the feedback will update based on your choices. Only your answers when you press the Next button, or when the timer expires, will be recorded.

Set 1:

All of the clothing was washed.
The light clothing was separated from the dark clothing.
The light clothing was washed.
The dark clothing was washed.
The clothing was washed in two loads.

Set 2:

All of the clothing was washed.
The light clothing was washed.
The dark clothing was washed.
The clothing was washed in one load.

☐ Set 1

☐ Set 2

Figure 5.16: User study screenshot showing the first scenario, with the user presented with two sets of fluents to choose between.

Set 2:

All of the clothing was washed.
The light clothing was washed.
The dark clothing was washed.
The clothing was washed in one load.

☒ Set 1

☐ Set 2

Sequence of actions to achieve the goal from your selection:

Action Step	Reasons for Taking this Action
1. Separate light and dark clothing	This makes "The light clothing was separated from the dark clothing" Achieved (True). Because "All of the clothing was washed" is Achieved (True), it must also be Achieved (True) that "The light clothing was washed" and "The dark clothing was washed."
2. Wash light clothing	This makes "The light clothing was washed" Achieved (True).
3. Wash dark clothing	This makes "All of the clothing was washed" Achieved (True), and also makes "The clothing was washed in two loads" Achieved (True). This also makes "The dark clothing was washed" Achieved (True).



Figure 5.17: User study screenshot showing showing first scenario, with the plan provided to the user that achieves their selected set of goals.

All of the clothing was washed.
The light clothing was separated from the dark clothing.
The light clothing was washed.
The dark clothing was washed.
The clothing was washed in two loads.

Set 2:

All of the clothing was washed.
The light clothing was washed.
The dark clothing was washed.
The clothing was washed in one load.

☐ Set 1

☒ Set 2

Steps to achieve the goal from your selection:

Action Step	Reasons for Taking this Action
1. Wash all clothes	This makes "All of the clothing was washed" Achieved (True), and also makes "The clothing was washed in one load" Achieved (True). Because "All of the clothing was washed" is Achieved (True), it must also be Achieved (True) that "The light clothing was washed" and "The dark clothing was washed." Both of these are also made Achieved (True) with this step.



Figure 5.18: User study screenshot showing showing first scenario, with the plan provided to the user that achieves their selected set of goals. This shows a different plan from the different selection.

0350

Given the following statements for the scenario as described, please mark whether this statement should be Achieved (True) or Not Achieved (False) at the end of the scenario, when the robot has finished changing the light bulbs. Please provide an answer for each statement.

As you make selections, the actions the robot will take (with reasons for taking each action) will be shown to you below the choices, or the reasons why no steps can be generated. You may change your selections as many times as you wish, and the feedback will update based on your choices. Only your answers when you press the Next button, or when the timer expires, will be recorded.

	Achieved (True)	Not Achieved (False)
Two new bulbs were in the compartment when the robot ascended the first time.	☐	☐
One new bulb was in the compartment when the robot ascended the first time.	☐	☐
The first light bulb was changed.	☐	☐
The second light bulb was changed.	☐	☐
The first old bulb was dropped.	☐	☐
One new bulb was in the compartment when the robot ascended the second time.	☐	☐
The first old light bulb was returned.	☐	☐

Figure 5.19: User study screenshot showing second scenario, showing the entire list of fluents that the user must give preference about.

change your selections as many times as you wish, and the feedback will update based on your choices. Only your answers when you press the Next button, or when the timer expires, will be recorded.

	Achieved (True)	Not Achieved (False)
Two new bulbs were in the compartment when the robot ascended the first time.	☒	☐
One new bulb was in the compartment when the robot ascended the first time.	☒	☐
The first light bulb was changed.	☒	☐
The second light bulb was changed.	☒	☐
The first old bulb was dropped.	☒	☐
One new bulb was in the compartment when the robot ascended the second time.	☒	☐
The first old light bulb was returned.	☒	☐

No sequence of actions to achieve the goal is possible from your selection.

Reason: The statement "Two new bulbs were in the compartment when the robot ascended the first time" cannot be Achieved (True) at the same time as the statement "One new bulb was in the compartment when the robot ascended the first time" is Achieved (True).



Figure 5.20: User study screenshot showing second scenario, with the explanation provided to the user when they have chosen a goal that is not possible.

	Achieved (True)	Not Achieved (False)
Two new bulbs were in the compartment when the robot ascended the first time.	☒	☐
One new bulb was in the compartment when the robot ascended the first time.	☐	☒
The first light bulb was changed.	☒	☐
The second light bulb was changed.	☒	☐
The first old bulb was dropped.	☒	☐
One new bulb was in the compartment when the robot ascended the second time.	☐	☒
The first old light bulb was returned.	☐	☒

Sequence of actions to achieve the goal from your selections:

Action Step	Reasons for Taking this Action
1. Bring two new light bulbs up.	This makes "Two new bulbs were in the compartment when the robot ascended the first time" Achieved (True). This leaves zero empty spaces in the compartment
2. Change first light bulb with zero empty spaces in the compartment.	This makes "The first light bulb was changed" Achieved (True). The bulb is dropped because two bulbs are in the compartment, not allowing the robot to place the old bulb in the compartment before removing the new one. This makes "The first old bulb was dropped" Achieved (True).
3. Change second light bulb with one empty space in the compartment.	This makes "The second light bulb was changed" Achieved (True). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.

Figure 5.21: User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected goals.

change your selections as many times as you wish, and the feedback will update based on your choices. Only your answers when you press the Next button, or when the timer expires, will be recorded.

	Achieved (True)	Not Achieved (False)
Two new bulbs were in the compartment when the robot ascended the first time.	☐	☒
One new bulb was in the compartment when the robot ascended the first time.	☒	☐
The first light bulb was changed.	☒	☐
The second light bulb was changed.	☒	☐
The first old bulb was dropped.	☐	☒
One new bulb was in the compartment when the robot ascended the second time.	☒	☐
The first old light bulb was returned.	☒	☐

Sequence of actions to achieve the goal from your selections:

Action Step	Reasons for Taking this Action
1. Bring one new light bulb up.	This makes "One new bulb was in the compartment when the robot ascended the first time" Achieved (true). This leaves one empty space in the compartment.
2. Change first light bulb with one empty space in the compartment.	This makes "The first light bulb was changed" Achieved (true). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.
3. Return first old light bulb.	This makes "The first old light bulb was returned" Achieved (true). This leaves two empty spaces in the compartment.
4. Bring one new light bulb up.	This makes "One new bulb was in the compartment when the robot ascended the second time" Achieved (true). This leaves one empty space in the compartment.
5. Change second light bulb with one empty space in the compartment.	This makes "The second light bulb was changed" Achieved (true). The bulb is not dropped because only one bulb is in the compartment, allowing the robot to place the old bulb in the compartment before removing the new one.

Figure 5.22: User study screenshot showing showing second scenario, with the plan provided to the user that achieves their selected goals. The goals selected lead to the desired outcome, but a longer plan.

Chapter 6

Conclusions

This thesis presents the first attempt at trying to resolve the problem of goal underspecification by helping the user better understand the task and, by extension, the consequences of their goal specification. In particular, the use of model abstractions as a way to inform the users about only the part of the task that is relevant to reason about minimal unavoidable side-effects for the given goal specification was a particular focus. Two specific ways of doing it are provided, namely a blind exhaustive search and a planning compilation. Both user studies and computational experiments were performed to verify the proposed method. The initial results seem to point towards the usefulness of the method.

Going forward, extending these methods to be applicable beyond classical planning is a strong possible direction. Here, increasing both the complexity of the task (including those that contain numeric and temporal constraints) and looking at richer objective specification mechanisms (like rewards) would be of interest. Such settings might require a novel abstraction mechanism. Additionally, if the planning problems are quite complex, directly planning in the concrete model may not be feasible, and novel heuristics may need to be developed to search over the abstraction lattice.

Bibliography

- [1] Stuart Russell. Artificial intelligence and the problem of control. In *Perspectives on digital humanism*, pages 19–24. Springer Nature, 2022.
- [2] Victoria Krakovna, Laurent Orseau, Richard Ngo, Miljan Martic, and Shane Legg. Avoiding side effects by considering future tasks. *Advances in Neural Information Processing Systems*, 33:19064–19074, 2020.
- [3] Shun Zhang, Edmund H Durfee, and Satinder Singh. Minimax-regret querying on side effects for safe optimality in factored markov decision processes. In *IJCAI*, pages 4867–4873, 2018.
- [4] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In *27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 4829–4836. International Joint Conferences on Artificial Intelligence, 2018.
- [5] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.
- [6] Dylan Hadfield-Menell, Stuart Russell, Pieter Abbeel, and Anca D. Dragan. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3909–3917, Barcelona, Spain, 2016. Curran Associates, Inc.
- [7] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J. Russell, and Anca D. Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6765–6774, Long Beach, CA, USA, 2017. Curran Associates, Inc.

- [8] Malek Mechergui and Sarath Sreedharan. Goal alignment: re-analyzing value alignment problems using human-aware ai. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10110–10118, 2024.
- [9] Toryn Q Klassen, Sheila A McIlraith, Christian Muise, and Jarvis Xu. Planning to avoid side effects. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9830–9839, 2022.
- [10] Sandhya Saisubramanian, Ece Kamar, and Shlomo Zilberstein. A multi-objective approach to mitigate negative side effects. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 354–361, 2021.
- [11] Sandhya Saisubramanian, Shlomo Zilberstein, and Ece Kamar. Avoiding negative side effects due to incomplete knowledge of ai systems. *AI magazine*, 42(4):62–71, 2021.
- [12] David E Smith. Choosing objectives in over-subscription planning. In *ICAPS*, volume 4, page 393, 2004.
- [13] Rebecca Eifler, Martim Brandao, Amanda Coles, Jeremy Frank, and Jörg Hoffmann. Evaluating plan-property dependencies: A web-based platform and user study. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pages 687–691, 2022.
- [14] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan explanations as model reconciliation: moving beyond explanation as soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 156–163, 2017.
- [15] Sarath Sreedharan, Siddharth Srivastava, David Smith, and Subbarao Kambhampati. Why can’t you do that hal? explaining unsolvability of planning tasks. In *International Joint Conference on Artificial Intelligence*, 2019.

- [16] Kelsey Sikes, Sarah Keren, and Sarath Sreedharan. Reducing human-robot goal state divergence with environment design. In *ICAPS 2024 Workshop on Human-Aware Explainable Planning*, 2024.
- [17] J Dennis Lawrence, Nuclear Regulatory Commission, et al. Software safety hazard analysis. Technical report, Nuclear Regulatory Commission, Washington, DC (United States). Div. of Reactor Controls and Human Factors, 1996.
- [18] Nathaniel Ozarin and Michael Siracusa. A process for failure modes and effects analysis of computer software. In *Annual Reliability and Maintainability Symposium, 2003.*, pages 365–370. IEEE, 2003.
- [19] Ayal Taitler, Ron Alford, Joan Espasa, Gregor Behnke, Daniel Fišer, Michael Gimelfarb, Florian Pommerening, Scott Sanner, Enrico Scala, Dominik Schreiber, Javier Segovia-Aguas, and Jendrik Seipp. The 2023 international planning competition. *AI Magazine*, 45(2):280–296, 2024.
- [20] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.
- [21] Malte Helmert. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence*, 173(5-6):503–535, 2009.
- [22] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: what’s the difference anyway? In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 19, pages 162–169, 2009.
- [23] Andrea Micheli, Arthur Bit-Monnot, Gabriele Röger, Enrico Scala, Alessandro Valentini, Luca Framba, Alberto Rovetta, Alessandro Trapasso, Luigi Bonassi, Alfonso Emilio Gerevini, Luca Iocchi, Felix Ingrand, Uwe Köckemann, Fabio Patrizi, Alessandro Saetti, Ivan Serina, and Sebastian Stock. Unified planning: Modeling, manipulating and solving ai planning problems in python. *SoftwareX*, 29:102012, 2025.

- [24] Robert A Hearn and Erik D Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005.
- [25] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.