# Partial-Order Ambiguous Observations of Fluents and Actions for Goal Recognition as Planning

*Jennifer Nelson*
*University of Utah*

UUCS-20-004

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

26 April 2020

**Abstract**

This work readies goal recognition for real-world scenarios by adapting a foundational compilation by Ramírez and Geffner to work with partial-order, ambiguous observations of both facts and actions. We first redefine what observations can be and what it means to satisfy them. We provide a compilation from goal recognition problem to classical planning problem, then prove it accommodates these more complex observation types. Our compilation can be adapted towards other planning-based plan/goal recognition techniques, as Ramírez and Geffner's compilation was.

We prove that our method is at least as accurate as an "ignore complexity" strategy that uses Ramírez and Geffner's compilation. Experimental results confirm that, while slower, our method never has more (and often has fewer) false positives. (Both methods have no false negatives.) We discuss these findings in the context of goal recognition problem difficulty, and present an avenue for future work.

# PARTIAL-ORDER AMBIGUOUS OBSERVATIONS OF FLUENTS AND ACTIONS FOR GOAL RECOGNITION AS PLANNING

by
Jennifer Nelson

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

The School of Computing

Approved:

| | |
|---|---|
| Rogelio E. Cardona Rivera<br>Thesis Faculty Supervisor | Ross Whitaker<br>Director, School of Computing |
| Erin Parker<br>Honors Faculty Advisor | Sylvia D. Torti<br>Dean, Honors College |

March 2020

# Partial-Order Ambiguous Observations of Fluents and Actions for Goal Recognition as Planning


*Jennifer Nelson*
*University of Utah*



UUCS-20-004



School of Computing
University of Utah
Salt Lake City, UT 84112 USA



26 April 2020


**Abstract**


This work readies goal recognition for real-world scenarios by adapting a foundational compilation by Ramírez and Geffner to work with partial-order, ambiguous observations of both facts and actions. We first redefine what observations can be and what it means to satisfy them. We provide a compilation from goal recognition problem to classical planning problem, then prove it accommodates these more complex observation types. Our compilation can be adapted towards other planning-based plan/goal recognition techniques, as Ramírez and Geffner's compilation was.

We prove that our method is at least as accurate as an "ignore complexity" strategy that uses Ramírez and Geffner's compilation. Experimental results confirm that, while slower, our method never has more (and often has fewer) false positives. (Both methods have no false negatives.) We discuss these findings in the context of goal recognition problem difficulty, and present an avenue for future work.

ABSTRACT

This work readies goal recognition for real-world scenarios by adapting a foundational compilation by Ramírez and Geffner to work with partial-order, ambiguous observations of both facts and actions. We first redefine what observations can be and what it means to satisfy them. We provide a compilation from goal recognition problem to classical planning problem, then prove it accommodates these more complex observation types. Our compilation can be adapted towards other planning-based plan/goal recognition techniques, as Ramírez and Geffner's compilation was. We prove that our method is at least as accurate as an "ignore complexity" strategy that uses Ramírez and Geffner's compilation. Experimental results[1] confirm that, while slower, our method never has more (and often has fewer) false positives. (Both methods have no false negatives.) We discuss these findings in the context of goal recognition problem difficulty, and present an avenue for future work.


[1]Compilation, evaluation, and analysis code can be found at `https://github.com/qed-lab/Complex-Observation-Compiler`

# TABLE OF CONTENTS

# 1 Introduction

Plan/goal recognition is the problem of identifying the plans and goals of an agent, given some observations of their behavior(s) [10]. Plan/goal recognition has applications wherever a system needs to anticipate an agent's actions or desires. This variety includes robot-human coordination [11], human-computer collaboration [4], assisted cognition [5], network monitoring [8], interactive narratives [2], and language recognition [1, 12].

Ramírez and Geffner [7] realized that goal recognition problems are similar to classical planning problems, and created a formulation to compile recognition problems into planning problems ready for off-the-shelf planning algorithms. Previously, plan recognition relied on specialized algorithms and handcrafted libraries. Rather than rely on a library of possible plan-goal pairs, Ramírez and Geffner's formulation relies on a set of possible goals and a *domain theory* describing possible actions. It assumes that any plan which reaches a possible goal optimally, while also "explaining" all observations in order is part of the solution to a recognition problem.

In addition to defining an optimal solution set, Ramírez and Geffner [7] also relaxed its own optimality assumption to allow approximate solutions computed with faster algorithms. This also allowed solutions to "skip" some observations if necessary. Ramírez and Geffner [6] also relaxed the optimality assumption, such that goals whose optimal plans differed significantly from the observations were considered less likely. Sohrabi et al. [9] further relaxed the optimality assumption, admitting that observation sequences may be non-optimal, noisy, or missing segments. It assumed observations of single fluents, rather than actions.

The methods above all assume total-ordered fully specified observations, though real-world applications may be more complex. One might find artifacts of past actions, but not know the order in which the artifacts appeared. One might see the actor pick something up,

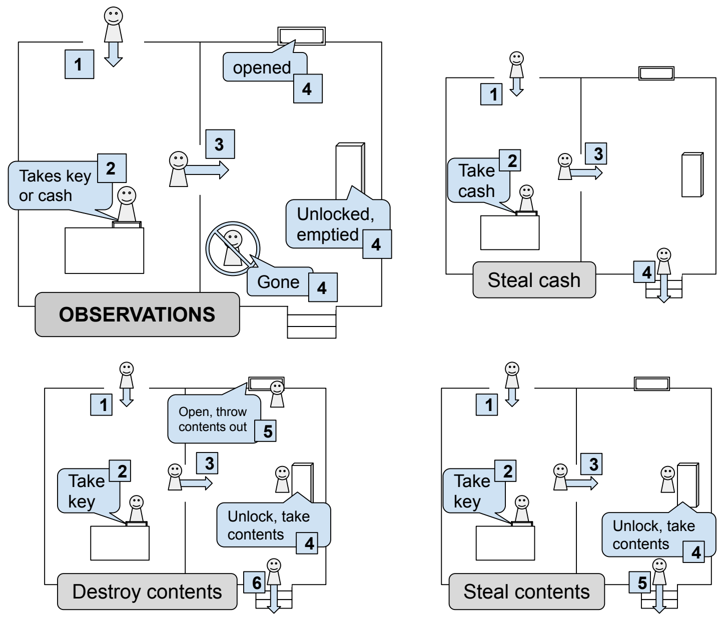


Figure 1: DetectiveBot's observations, and unconstrained plans for the culprit's possible motivations

but not know what was picked up (an ambiguous observation). One might later observe that a key is missing from that spot (a fluent observation). This is important information if the agent's goal is behind a locked door, but current methods cannot use it. Our work provides methods to utilize this information. In this work we modify the original [7] compilation, but our approach can be adapted for any of the methods mentioned above. We focus only on the "optimal" set of answers for complex observations, leaving relaxations to future work.

# 2 Motivating Example

We illustrate our method with the following scenario: DetectiveBot is trying to solve a breaking-and-entering case at a museum. Cameras record the culprit breaking into the museum office, rifling through the manager's top drawer, pocketing something, then sprinting into an unfilmed backroom. A witness states they saw the culprit running down a stairwell later that night. DetectiveBot inspects the backroom: it contains a single window (opened), a chest (unlocked and emptied), and the stairwell towards the exit. DetectiveBot wants to figure out the culprit's motives. Were they stealing cash from the managers drawer? Were they stealing the contents of the chest? Or were they destroying the contents of the chest?

DetectiveBot knows the culprit took either cash or a key to the chest from the drawer, then entered the back room. DetectiveBot does not know what order things happened in the backroom, but it knows that the window was opened, the chest was unlocked and emptied, and the culprit left via the stairwell. First DetectiveBot computes plans for what the culprit would've done for each of their three possible motives, unconstrained by DetectiveBot's observations (Figure 2). Then it computes plans for each of the three possible motives, such that each plan also "sees" (satisfies) the observations. DetectiveBot compares the unconstrained plans to their constrained counterparts, and discovers that only one pair has identical costs: destroying the contents.

Fortunately, DetectiveBot just upgraded its plan recognition software, or else it would've been forced to model the situation as a sequence of fully-specified ordered steps. It couldn't have modeled the culprit rifling through the drawers, because it didn't know what the culprit was taking. It wouldn't know how to model anything about the backroom, because it didn't see any actions, just the results. Even if it could model the backroom facts as actions, it still wouldn't know their relative order. It would've had to either forget them or fix their order, possibly accidentally fixing the culprit *leaving* before opening the chest or window.

# 3 The Goal Recognition Problem with Complex Observation Constraints

**Planning Background** This paper relies on the formulation of **classical planning**. Classical planning is a model of problem solving, wherein agent actions are fully observable and deterministic. Classical problems are typically represented in the STRIPS formalism [3]. A STRIPS planning problem is a tuple $P = \langle F, I, A, G, f_{\text{cost}} \rangle$ where $F$ is the set of fluents, $I \subseteq F$ is the initial state, $G \subseteq F$ is the set of goal conditions, and $A$ is a set of deterministic actions. Each action is a triple $a = \langle \text{PRE}(a), \text{ADD}(a), \text{DEL}(a) \rangle$, that represents the precondition, add, and delete lists respectively, all subsets of $F$. A state is a conjunction of fluents. An action $a$ is applicable in a state $s$ if $\text{PRE}(a) \subseteq s$; applying said applicable action in the state results in a new state $s' = (s \setminus \text{DEL}(a)) \cup \text{ADD}(a)$ and incurs a non-negative cost determined by the function $f_{\text{cost}}$:$A \rightarrow R^{0+}$.

The solution to a planning problem $P$ is a plan $\pi = [a_1, ..., a_m]$, a sequence of actions $a_i \in A$ that transforms the problem's initial state $I$ to a state $s_m$ that satisfies the goal; *i.e.,* $G \subseteq s_m$. The cost $\text{c}(\pi)$ of a plan $\pi$ is $\sum_{a_i \in \pi} f_{\text{cost}}(a_i)$. A plan segment is a section of a plan, denoted $\pi_j^k = [a_j, ..., a_k]$ $(a_i \in A, 1 \leq j \leq k \leq m)$.

The execution trace $trace(\pi, I) = [I, a_1, s_1, ..., a_m, s_m]$ of plan $\pi$ from initial state $I$ is defined as the alternating sequence of states and actions, starting with $I$, such that $s_i$ results from applying $a_i$ to state $s_{i-1}$.

**Handling Complex Observations** Our formulation is based on the formulation by Ramírez and Geffner, but we exchange the sequence of total-order action observations $O$ with the more expressive "observation group" $\Theta$. For brevity, we use the notation $P[G]$ to mean an incomplete planning problem $P$ completed with the addition of the goal $G$.

**Definition 1.** *A **goal recognition problem over a domain theory** is the tuple* $T = \langle P, \mathcal{G}, \Theta \rangle$*, where* $P = \langle F, I, A, f_{\text{cost}} \rangle$ *is an incomplete planning problem,* $\mathcal{G}$ *is the set of possible goals, and* $\Theta$ *is an observation group as defined below. The solution to* $T$ *is the set* $\mathcal{G}^* = \{G \in \mathcal{G} : \exists \pi$ *satisfying* $\Theta$ *and optimally solving* $P[G]\}$

$\Theta$ relaxes the assumption that all observations are totally ordered and grounded actions. Instead, we allow an observation to be either an observed action or a set of observed fluents. Further, we allow partial orderings in the observations as well as ambiguous observations via sets of possible observations.

Fundamental to this formulation are **observation groups**, which impose constraints on the observations they contain. We describe two types of observations, three types of groups, and what it means for a plan to **satisfy** each. To enable partial-ordering, we define satisfaction of a group/observation *by* a plan *through* a plan segment.

**Definition 2.** *An **action observation*** $o$ *of action* $a \in A$ *is satisfied by the plan* $\pi$ *through segment* $\pi_j^k$ *iff* $a = a_i$ *for some* $a_i \in \pi_j^k$ *(*$j \leq i \leq k$*).*

An action observation is merely an action, and is satisfied by plan segments that contain that action.

**Definition 3.** *A **fluent observation*** $o$ *of fluents* $(F_o \subseteq F)$ *is satisfied by the plan* $\pi$ *with initial state* $I$ *through segment* $\pi_j^k$ *iff* $F_o \subseteq s_i$ *for some* $s_i$ $(j \leq i \leq k)$ *in* $trace(\pi, I)$*.*

A fluent observation is a set of fluents, and is satisfied by plan segments that mark out a time period where those fluents are true for some state. The actions in the plan segment do not need to contribute to the observed fluents for this notion of satisfaction. The plan segment merely marks a time period in which the fluents were observed. It may be that $F_o$ was true since the initial state, but was not observable until much later. Our intent is to rule out goal-plan pairs where the plan never co-occurs with the fluents in $F_o$ being true.

Now we define **ordered groups** that impose ordering constraints on members. An ordered group can only be satisfied by a plan segment if that segment can be split into chunks that satisfy each member *in order*. (These chunks are the reason we define satisfaction in terms of plan segments.)

**Definition 4.** *An **ordered observation group** $\Theta_< = [\theta_1, ..., \theta_n]$ is a totally ordered sequence of observation groups and/or simple observations. A plan $\pi$ satisfies $\Theta_<$ through the plan segment $\pi_j^k$ iff there exists a monotonically increasing function of the form $f : [1, n+1] \rightarrow [j, k+1]$, which maps members of $\Theta_<$ to segments of $\pi_j^k$ such that $\pi_{f(i)}^{f(i+1)-1}$ satisfies $\theta_i$.*

The function $f$ above is used to ensure ordering. It maps consecutive group members to consecutive plan segments. $f(i)$ marks the beginning of $\theta_i$'s plan segment. We allow no gaps in plan segments, so $\theta_i$'s segment ends right before $\theta_{i+1}$'s segment begins, and $\theta_n$'s segment ends where the whole plan segment ends.

Next we define **unordered groups** that are only satisfied when all members are. When embedded in an ordered group, these form the *partial* part of *partial order*.

**Definition 5.** *An **unordered group** $\Theta_\wedge = \{\theta_1, ..., \theta_n\}$ is a set of observation groups and/or simple observations that have no ordering constraints with respect to each other. A plan $\pi$ satisfies $\Theta_\wedge$ through the segment $\pi_j^k$ iff $\pi_j^k$ satisfies all members.*

Lastly, we define **option groups**. Unlike the other groups, this group may contain only simple observations, not nested groups, and is intended to describe a set of mutually exclusive *possible* observations. This is how we support ambiguous observations: by transforming each into an option group of all its possible interpretations. This group is satisfied if at least one of its members is satisfied.

**Definition 6.** *An **option group** $\Theta_\oplus = |o_a, ..., o_b|$ is a set of single observations. A plan $\pi$ satisfies $\Theta_\oplus$ through the segment $\pi_j^k$ satisfies at least one member.*

# 4 Compilation to Planning Problem

We compile a plan recognition problem into a set of planning problems $\{P'[G'] \mid G'$ derived from $G \in \mathcal{G}\}$ such that a solution to $P'[G']$ "explains" the observations nested in $\Theta$, while respecting $\Theta$'s ordering constraints and not double-explaining different observations in the same option group. If an optimal solution to $P'[G']$ has the same cost as an optimal solution to $P[G]$, $G$ and the plan solving $P'[G']$ are considered a solution to the plan recognition problem.

This compilation adds an "explanation" action for every observation. To ensure a solution to $P'[G']$ respects $\Theta$'s constraints, we use ordering fluents to ensure an explanation may only happen *after* all prior observations have been explained, and that only one explanation is allowed per observation, or per option group. Let $nest(\theta)$ denote the set of all observations nested within $\theta$ or its subgroups. (For example, an observation can be placed in an option group that is inside an unordered group, which is embedded in an ordered group.)

**Definition 7.** *For the plan recognition problem $T = \langle P = \langle F, I, A, f_{\text{cost}} \rangle, \mathcal{G}, \Theta \rangle$ the planning problem for each $G \in \mathcal{G}$ is defined as $P'[G'] = \langle F', I', A', G' \rangle, f'_{\text{cost}}$ such that:*

- $F' = F \cup F_e$*, where* $F_e = \{p_{o_i} | \forall o_i \in nest(\Theta)\}$
- $I' = I$
- $A' = A \cup A_e$*, where* $A_e = \{e_{o_i} | \forall o_i \in nest(\Theta)\}$*, and*
- $G' = G \cup F_e$*.*
- $f'_{\text{cost}}$ *is similar to* $f_{\text{cost}}$*, with the addition of costs for* $A_e$*. (See Definitions 8 and 9.)*

We further define $A_e$ and $F_e$, and later prove that a solution to $P'[G']$ satisfies $\Theta$.

**Definition 8.** *The explanation action $e_{o_i}$ for the fluent observation $o_i$ corresponding to fluents $F_{o_i} \subseteq F$ is a dummy action that marks that $F_{o_i}$ is observed, defined as:*

- $\text{PRE}(e_{o_i}) = F_{o_i} \cup \{\neg p_{o_i}\} \cup \{p_{o_{pre}} | o_{pre} \in B\}$ *where $B$ is the set of all observations nested in any group immediately preceding a group that $o_i$ is nested within.*
- $\text{ADD}(e_{o_i}) = \{p_{o_i}\}$
- $\text{DEL}(e_{o_i}) = \emptyset$
- $f_{\text{cost}}(e_{o_i}) = 0$
- $p_{o_i} = p_{o_j}$ *for all $o_j$ in the same option group as $o_i$*

This definition is based on those of Sohrabi et al. [9], except multiple fluents can be included in the same observation. A metric planner is needed to work with this zero-cost action, or the cost of these actions can be subtracted post-planning.

**Definition 9.** *The explanation action $e_{o_i}$ for the action observation $o_i$ corresponding to action $a \in A$ is an action identical to $a$ but with additional ordering fluents:*

- $\text{PRE}(e_{o_i}) = \text{PRE}(a) \cup \{\neg p_{o_i}\} \cup \{p_{o_{pre}} | o_{pre} \in B\}$ *where $B$ is the set of all observations nested in any group immediately preceding a group that $o_i$ is nested within.*
- $\text{ADD}(e_{o_i}) = \text{ADD}(a) \cup \{p_{o_i}\}$
- $\text{DEL}(e_{o_i}) = \text{DEL}(a)$
- $f_{\text{cost}}(e_{o_i}) = f_{\text{cost}}(a)$
- $p_{o_i} = p_{o_j}$ *for all $o_j$ in the same option group as $o_i$*

Note that explanation actions have the precondition $\neg p_{o_i}$, but add $p_{o_i}$ as an effect. As no action removes $p_{o_i}$, this means an explanation action cannot be used twice. Additionally, explaining an observation in an option group prevents all other explanations from that option group from being used. In the next section, we prove that our compilation indicates members of $\mathcal{G}^*$: $G$ is in $\mathcal{G}^*$ when the optimal plan for $P'[G']$ costs the same as an optimal plan for $P[G]$. To find all members of $\mathcal{G}^*$, we optimally solve $P'[G']$ and $P[G]$ for all $G$ in $\mathcal{G}$, and compare costs.

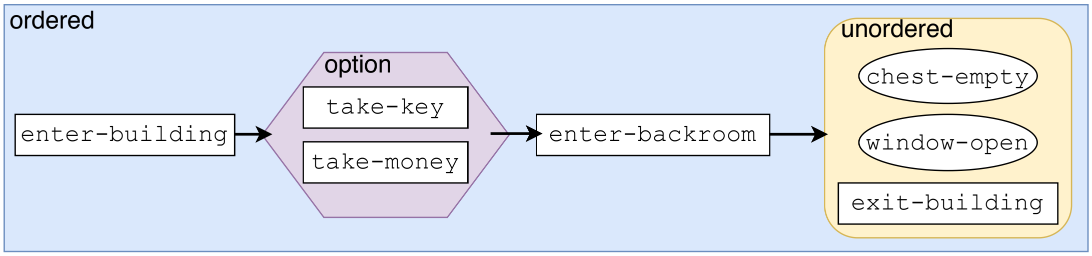


Figure 2: DetectiveBot's observation groups as diagram.

### 4.1 Example

We illustrate this compilation using the motivating example. DetectiveBot encodes its observations using the following observation groups:

```
[enter-building, |take-key,take-money|, enter-backroom,
 {(window-opened),(chest-empty), exit-building}]
```

(Square brackets indicate an ordered group, bars indicate an option group, curly brackets indicate an unordered group, and parentheses indicate a fluent observation)

From this, DetectiveBot compiles its model of the world, the observations, and one of its posited goals: `outside` $\wedge$ `contents-destroyed`. It first fully grounds the domain using any objects in the world (in the example domain little grounding is needed, though our software grounds `(NOT (OUTSIDE))` to `(NOT-OUTSIDE)`). It adds six ordering fluents, two for the entering actions, one for the option group (`MUTEX_1`), and three for the unordered group. The problem's goal is to see all of these ordering fluents, and to see the posited goal. DetectiveBot keeps normal actions available, and adds actions corresponding to each of its observations. For example, the observation of `take-key` compiles to the following:

```
(:action EXPLAIN_OBS_TAKE-KEY
  :parameters ()
  :precondition (and
    ( not ( MUTEX_1 ) ) ;; Don't repeat anything in this option group
    ( KEY-IN-DRAWER )
    ( IN-OFFICE )
    ( NOT-OUTSIDE )
    ( OBSERVATION_0 )   ;; Must have seen prior observation
  )
  :effect (and
    (increase (total-cost) 1)
    ( HOLDING-KEY )
    (not ( KEY-IN-DRAWER ))
    ( MUTEX_1 )         ;; Mark that we've observed this option group
```

```
  )
)
```

If an unordered group was prior to the option group, instead of the observation `enter-building`, `EXPLAIN_OBS_TAKE_KEY` would have ordering fluents from every observation in the prior unordered group, and anything nested in that unordered group.

The full original domain, problem, and compiled version can be found in Appendix A.

# 5 Proofs

In this section we present two main proofs. The first is a proof that our compilation indicates if a goal is in the solution to a goal recognition problem; *i.e.,* if the goal has an observation-satisfying plan that optimally reaches the goal. The second is a proof that our compilation will never yield a goal set larger than the goal set created by ignoring complex observations and using Ramírez and Geffner's 2009 compilation. This proves that, with respect to accuracy (*i.e.,* the size of the optimal goal set), our compilation is no worse than the compilation by Ramírez and Geffner. The subsequent section presents and empirical evaluation demonstrating that in several cases we are in fact better.

## 5.1 Goal Recognition Problem is Solved

We prove that our compilation produces a planning problem that solves the goal recognition problem in two steps. We first prove that any plan $\pi$ for $P'[G']$ has a corresponding plan $\psi(\pi)$ of equivalent cost that solves $P[G]$. We then prove that $\psi(\pi)$ satisfies $\Theta$ and (by the first proof) solves $P[G]$ with the same cost as $\pi$. If this cost is the same as an optimal plan for just $P[G]$, then $G$ is in the solution set $\mathcal{G}^*$ for $T$.

**Theorem 1.** *A plan $\pi$ for $P'[G']$ has a corresponding plan $\psi(\pi)$, solving $P[G]$, such that $c(\pi) = c(\psi(\pi))$.*

*Proof.* For $\pi$, let $\psi(\pi)$ be the same sequence of actions, but with fluent observation explanations removed, and action observation explanations replaced with their corresponding action in $A$. Because fluent explanations have no cost, and action explanations cost the same as their corresponding action, $c(\pi) = c(\psi(\pi))$. Fluent explanations only effect ordering fluents, and action explanations are identical to their corresponding actions, save for ordering fluents. Since $G$ does not include any ordering fluents, $\psi(\pi)$ still achieves $G$. □

**Theorem 2.** *If plan segment $\pi_j^k = [a_j, ..., a_k](a_i \in A')$ achieves all $p_{o_i}$ for $o_i \in nest(\Theta)$, then $\psi(\pi_j^k)$ satisfies $\Theta$.*

*Proof.* We prove this theorem through a series of Lemmas showing that such a plan segment will satisfy every observation and observation group.

**Lemma 2.1** (Simple Observations). *If $\pi_j^k$ achieves $p_{o_i}$, then $\psi(\pi_j^k)$ satisfies $o_i$.*

*Proof.* The only way for $\pi_j^k$ to achieve $p_{o_i}$ is through explanation action $e_{o_i}$. If $o_i$ is an observation of action $a$, then $e_{o_i}$ is translated to $a$ in $\psi(\pi_j^k)$, satisfying $o_i$. If $o_i$ is an observation of fluents $F_{o_i}$, then $e_{o_i}$ has $F_{o_i}$ as precondition, so $F_{o_i}$ must exist in the execution trace for $\pi_j^k$, and thus in the trace for $\psi(\pi_j^k)$. In either case, $\psi(\pi_j^k)$ satisfies $o_i$. □

**Lemma 2.2** (Option Group). *If $\pi_j^k$ achieves any $p_{o_i}$ for $o_i$ in the option group $\Theta_\oplus$, then $\psi(\pi_j^k)$ satisfies $\Theta_\oplus$.*

*Proof.* If $\pi_j^k$ achieves a particular $p_{o_i}$ for $o_i \in \Theta_\oplus$, then by Lemma 2.1, $\psi(\pi_j^k)$ satisfies $o_i$. By satisfying a member of $\Theta_\oplus$, $\psi(\pi_j^k)$ satisfies $\Theta_\oplus$. □

**Lemma 2.3** (Unordered Group). *If $\pi_j^k$ achieves all $p_{o_i}$ for $o_i \in nest(\Theta_\wedge)$, then $\psi(\pi_j^k)$ satisfies $\Theta_\wedge$.*

*Proof.* $\psi(\pi_j^k)$ satisfies every simple observation and option group contained directly in $\Theta_\wedge$, per Lemmas 2.1 and 2.2. $\psi(\pi_j^k)$ also satisfies any contained option groups, per Lemma 2.2. If $\Theta_\wedge$ contains unordered groups, this is equivalent to containing the unordered group's members directly. Any contained ordered groups are also satisfied by $\psi(\pi_j^k)$, via Lemma 2.4. □

**Lemma 2.4** (Ordered Group). *If $\pi_j^k$ achieves all $p_{o_i}$ for $o_i$ nested in the ordered group $\Theta_< = [\theta_1, ..., \theta_n]$, $\psi(\pi_j^k)$ satisfies $\Theta_<$.*

*Proof.* Let $f : [1, n+1] \rightarrow [j, k+1]$ be a function where $f(n+1) = k+1$ and $f(i)$ is the index of the *first* explanation action for any $o \in nest(\theta_i)$. Segment $\pi_{f(i)}^{f(i+1)-1}$ then achieves all $p_o$ for $o \in nest(\theta_i)$, since the explanation action at $f(i+1)$ has $\{p_o \mid o \in \theta_i\}$ as a precondition. Via the other Lemmas, $\psi(\pi_{f(i)}^{f(i+1)-1})$ satisfies $\theta_i$.

Let $f_\psi$ be of the form $f_\psi : [1, n+1] \rightarrow [1, |\psi(\pi_j^k)| + 1]$ where $f_\psi(n+1) = |\psi(\pi_j^k)| + 1$ and $f_\psi(i)$ maps to where the action at $f(i)$ *would* be if the $\psi(\cdot)$ transformation did not remove/transform it. This way, $f_\psi$ creates plan segments corresponding to the plan segments $f$ creates, such that

$$\psi(\pi_{f(i)}^{f(i+1)-1}) = (\psi(\pi_j^k))_{f_\psi(i)}^{f_\psi(i+1)-1}$$

Since the left-hand side of this equation satisfies $\theta_i$, so too does the right-hand side. This makes $f_\psi$ a monotonically increasing function which separates $\psi(\pi_j^k)$ into sections which satisfy each member of $\Theta_<$. With it, $\psi(\pi_j^k)$ satisfies $\Theta_<$. □

Lemmas 2.3 and 2.4 recurse into themselves if an unordered group contains an ordered group (or vice versa), but are satisfied by the base case where a group contains only simple observations and/or option groups.
With Lemmas 2.1 - 2.4, we prove a plan segment achieving all $p_{o_i}$ has a corresponding plan that satisfies $\Theta$. □

An optimal solution to $P'[G']$ necessarily achieves all $p_{o_i}$, and so by Theorem 2, has a corresponding plan that satisfies $\Theta$. With Theorem 1, we prove that this corresponding plan also solves $P[G]$. If the cost of this plan is the same as the cost for an optimal plan to just $P[G]$ (not constrained by $\Theta$), then **a plan exists that satisfies $\Theta$ *and* optimally solves $P[G]$.** By definition 1, $G$ is in $T$'s solution set $\mathcal{G}^*$.

## 5.2 No Worse than Ignoring Complexity

We prove that our compilation will never yield a goal set larger than the goal set created by ignoring complex observations. We begin by defining an "ignore complexity" strategy for simplifying observation groups to a form Ramírez and Geffner's 2009 compilation can handle. This strategy removes fluent observations and option groups, reduces unordered groups to a single member, then simplifies one- or no-member groups. We choose this strategy over strategies that try different orderings/option group members because they would take exponentially longer to solve, requiring as many tries as there are combinations of unordered group orders and option group choices. We sketch a proof that using observations will always be at least as accurate as ignoring them. Accuracy is measured by number of goals indicated: fewer false positives is more accurate. It's worth noting that both compilations have perfect recall for goal recognition problems (if $G_{true}$ is in $\mathcal{G}$) but may have imperfect recall for plan recognition problems, which are concerned with indicating both the correct $G$ *and* the true plan $\pi$ used to achieve $G$.

**Theorem 3.** *Given $T_{cpx} = \langle P, \mathcal{G}, \Theta \rangle$ and $T_{ign} = \langle P, \mathcal{G}, \Theta_{ign} \rangle$, where $\Theta_{ign}$ removes some number of observations from $\Theta$ without altering order constraints, $|\mathcal{G}^*_{cpx}| \leq |\mathcal{G}^*_{ign}|$, where $\mathcal{G}^*_{cpx}$ is the solution set to $T_{cpx}$ and $\mathcal{G}^*_{ign}$ is the solution set to $T_{ign}$.*

*Proof Sketch.* Assume $|\mathcal{G}^*_{cpx}| > |\mathcal{G}^*_{ign}|$. Then there exists some $G \in \mathcal{G}$ such that $G \in \mathcal{G}^*_{cpx}$ but $G \notin \mathcal{G}^*_{ign}$. This means an explanation action for some observation in $\Theta_{ign}$ created a larger cost for the optimal plan for $P'[G']$ compiled for $T_{ign}$, making $c^*(P'[G']) > c^*(P[G])$ and eliminating $G$ from $\mathcal{G}^*_{ign}$. Because the observations in $\Theta_{ign}$ are a subset of those in $\Theta_{cpx}$, that explanation action will also incur a cost for $P'[G']$ compiled for $T_{cpx}$, eliminating $G$ from $\mathcal{G}^*_{cpx}$. This contradicts the premise, so $|\mathcal{G}^*_{cpx}| \leq |\mathcal{G}^*_{ign}|$. □

# 6 Experimental Evaluation

We evaluate the proposed formulation against the "ignore complexity" strategy for accommodating complex observations using the compilation in Ramírez and Geffner [7]. We use the same domains and plan recognition problems, but generate new observations according to two parameters. The metric we're concerned with is the number of incorrect goals in the optimal goal set $\mathcal{G}^*$. By this metric we often perform better, and never perform worse. In some domains the "ignore complexity" strategy often found no incorrect goals, leaving our formulation no room for improvement. We report how often this occurred, and focus on cases where we could improve. In general, our method is slower.

## 6.1 Method

**Hypotheses** We hypothesize that the size of our goal set will often be smaller, and never larger, than the size of the goal set computed using simplified observations. We also measure the time it takes to compute the optimal goal set.

**Apparatus** We developed our software by expanding the original plan recognition as planning code developed by Ramírez and Geffner [7]. Our software ran atop Centos 7.2 Linux with the 3.10 kernel, deployed on hardware equipped with a 3.60GHz Intel Core i7-4790 Processor, 32GB DDR3 1600MHz overclocked RAM, and 240GB Intel 540 Solid State Drive. Optimal plans were generated using A* search with admissible h-max heuristic. When computing plans in $P'[G']$ we pruned paths whose estimated cost-to-goal reached the optimal cost of $P[G]$ (pre-computed and not counted towards measures of time). This sped up how soon incorrect goals were excluded from $\mathcal{G}^*$. For each $P'[G']$, we allotted ten times the time taken to compute $P[G]$, with a minimum of 20 seconds. (In the logistics domain, this was sometimes still not enough time.)

**Procedure** For every plan recognition problem in Ramírez and Geffner [7], consisting of a domain $P$ and possible goals $\mathcal{G}$ of which $G_{true}$ is correct, we found an optimal plan for $P[G_{true}]$, and generated complex observations from both the plan (**A**) and plan's trace (**A+F**). For each plan/trace, we generated 3 observation sets per setting, varying setting by amount of observations of unknown order (**U%**) and amount of 'ambiguated' observations missing a parameter (**D%**). To generate observations, we randomly removed half the optimal plan/trace, and removed 90% of fluents from states in each trace. With the remaining plan/trace we chunked **U%** of observations into unordered groups of size $\sim$3. Lastly, we 'ambiguated' **D%** of action observations by removing one parameter and replacing the observation with an option group of all matching observations. (E.g. (eat H)$\rightarrow \Theta_{\oplus} = |$(eat A),...,(eat Z)$|$) **D%** was only applied to action observations with at least one parameter.

We varied **U** and **D** over five settings:

- No Obscuration: **U=0% D=0%**

- Vary Deboundedness: **U=0% D=25%**
- Vary Unorderedness: **U=25% D=0%**
- Vary Unorderedness: **U=50% D=0%**
- Vary Both: **U=50% D=25%**

For each problem $\langle P, \mathcal{G}, \Theta \rangle$, and each $G \in \mathcal{G}$, we compiled two planning domains: $P'[G']$ (this work's compilation) and $P'_{ign}[G'_{ign}]$ (Using the "ignore complexity" strategy and Ramírez and Geffner's 2009 compilation). We compare the optimal costs of both to the optimal cost for $P[G]$ (precomputed). If $c^*(P'[G']) = c^*(P[G])$, we place $G$ in $\mathcal{G}^*_{cpx}$. If $c^*(P'_{ign}[G'_{ign}]) = c^*(P[G])$, we place $G$ in $\mathcal{G}^*_{ign}$.

## 6.2 Analysis

We conducted tests over the four domains in [7]: Block-Words, Ipc-Grid, Grid-Navigation, and Logistics. The size of $|\mathcal{G}|$ for each domain averaged 20.32, 8.56, 4.22, and 10.00, respectively. For the latter three domains, most problems were perfectly solved with the "ignore complexity" strategy, leaving no room for improvement. We removed instances where our simplification strategy resulted in an empty observation set. (This occurred **38** times, mostly in the [**A+F U:50%D:25%**] setting.) Table 1 reports the number of perfectly solved problems (Opt) and problems with room for improvement (Imp), per setting, per domain. It also reports the average number of observations ($|\Theta|$) per method and the average size of the solution set ($|\mathcal{G}^*|$) when improvable, and the average time to compute (whether or not improvable). Error rates indicate a 95% confidence interval. For $|\Theta|$, we defined option groups as being size 1.

We conducted an independent t-test, not assuming same variance, comparing the sizes of solution sets when improvable. We found a statistically significant difference between $|\mathcal{G}^*_{ign}|$($\mu$=3.91, $\sigma$=2.99) and $|\mathcal{G}^*_{cpx}|$($\mu$=2.51, $\sigma$=1.73) ($t$($df$=2372.97)=15.620, $p$ <0.01, $\mu_{ign}$–$\mu_{cpx}$=1.40, $d$=0.57) We also found statistically significant differences ($p$ <0.01) for each domain, with Block-Words having the largest difference ($t$($df$=1703.41)=14.15, $\mu_{ign}$–$\mu_{cpx}$=1.64, $d$=0.61).

Figure 3 shows the results from Block-Words in more detail, comparing the size of $\mathcal{G}^*_{cpx}$ and $\mathcal{G}^*_{ign}$ for each setting. Notches represent a 95% confidence interval around the median value, and dashed lines represent mean. It only considers instances where $|\mathcal{G}^*_{ign}| > 1$, leaving room for improvement.

## 6.3 Discussion

Figure 3 shows that complex observations can be a crucial factor in eliminating false hypotheses. Particularly for scenarios with multiple types of complexity, such as the [**A+F U:50% D:25%**] setting, ignoring complexity can cost three or four false positives. In no

Table 1: Empirical Evaluation Results Per Domain and Setting

| | | Setting | | Number Samples | | Observation Set Size ($\|\Theta\|$) | | | | Solution Set Size ($\|\mathcal{G}^*\|$) | | Time (seconds) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Ignore Complexity | | Our Method | | Ignore Cplex | Complx | Ignore Cplx | Complex |
| | | U% | D% | Opt | Imp | Opt | Imp | Opt | Imp | Improvable Samples | | All Samples (Opt+Imp) | |
| **A** : Action Observations Only | Block-Words | 0% | 0% | 97 | 86 | 4.45 ± 0.24 | 4.08 ± 0.27 | 4.45 ± 0.24 | 4.08 ± 0.27 | 3.03 ± 0.33 | 3.03 ± 0.33 | **59.98 ± 3.25** | **77.11 ± 3.82** |
| | | 0% | 25% | 57 | 126 | 3.18 ± 0.19 | 2.75 ± 0.16 | 4.67 ± 0.29 | 4.10 ± 0.22 | **4.27 ± 0.48** | **3.17 ± 0.36** | **46.57 ± 3.08** | **93.38 ± 5.00** |
| | | 25% | 0% | 97 | 86 | 3.98 ± 0.15 | 3.78 ± 0.21 | 4.42 ± 0.23 | 4.12 ± 0.29 | 3.27 ± 0.33 | 3.02 ± 0.32 | **54.83 ± 2.97** | **83.55 ± 4.60** |
| | | 50% | 0% | 65 | 118 | 2.97 ± 0.14 | 2.81 ± 0.12 | 4.45 ± 0.28 | 4.19 ± 0.23 | **3.81 ± 0.40** | **2.86 ± 0.33** | **47.92 ± 2.98** | **99.81 ± 5.98** |
| | | 50% | 25% | 46 | 137 | 2.61 ± 0.23 | 2.14 ± 0.13 | 4.59 ± 0.33 | 4.18 ± 0.21 | **5.01 ± 0.60** | **3.42 ± 0.45** | **41.12 ± 3.04** | **115.78 ± 6.75** |
| | Ipc-Grid | 0% | 0% | 76 | 14 | 6.92 ± 0.53 | 7.21 ± 1.24 | 6.92 ± 0.53 | 7.21 ± 1.24 | 2.00 ± 0.00 | 2.00 ± 0.00 | **4.20 ± 0.84** | **8.53 ± 1.75** |
| | | 0% | 25% | 76 | 14 | 4.82 ± 0.41 | 5.14 ± 0.87 | 6.89 ± 0.53 | 7.36 ± 1.19 | 2.29 ± 0.42 | 1.93 ± 0.42 | **3.62 ± 0.74** | **8.01 ± 1.62** |
| | | 25% | 0% | 74 | 16 | 5.78 ± 0.41 | 6.25 ± 0.79 | 6.84 ± 0.54 | 7.56 ± 1.03 | 2.12 ± 0.27 | 2.06 ± 0.31 | **3.63 ± 0.72** | **8.41 ± 1.74** |
| | | 50% | 0% | 73 | 17 | 4.38 ± 0.35 | 4.65 ± 0.85 | 6.89 ± 0.54 | 7.29 ± 1.13 | 2.41 ± 0.66 | 1.94 ± 0.34 | **3.22 ± 0.65** | **10.88 ± 2.58** |
| | | 50% | 25% | 65 | 25 | 3.42 ± 0.34 | 3.16 ± 0.60 | 6.91 ± 0.59 | 7.12 ± 0.85 | **2.40 ± 0.55** | **1.64 ± 0.29** | **2.79 ± 0.54** | **11.70 ± 2.78** |
| | Navigation | 0% | 0% | 58 | 5 | 9.31 ± 1.42 | 5.40 ± 0.68 | 9.31 ± 1.42 | 5.40 ± 0.68 | 3.60 ± 2.72 | 3.60 ± 2.72 | 0.20 ± 0.04 | 0.21 ± 0.02 |
| | | 0% | 25% | 52 | 11 | 6.17 ± 1.13 | 6.55 ± 2.56 | 8.90 ± 1.50 | 9.45 ± 3.36 | 2.73 ± 0.85 | 2.09 ± 0.63 | 0.21 ± 0.07 | 0.19 ± 0.03 |
| | | 25% | 0% | 56 | 7 | 7.36 ± 1.11 | 7.57 ± 5.36 | 8.96 ± 1.37 | 9.29 ± 6.51 | 2.71 ± 1.38 | 2.57 ± 1.50 | 0.19 ± 0.05 | 0.17 ± 0.02 |
| | | 50% | 0% | 56 | 7 | 5.80 ± 0.91 | 6.29 ± 4.33 | 8.91 ± 1.37 | 9.71 ± 6.31 | 2.43 ± 0.73 | 2.00 ± 0.53 | 0.19 ± 0.03 | 0.19 ± 0.02 |
| | | 50% | 25% | 52 | 11 | 4.58 ± 0.83 | 4.64 ± 2.00 | 8.94 ± 1.44 | 9.27 ± 4.08 | 3.00 ± 1.08 | 1.91 ± 0.97 | 0.19 ± 0.04 | 0.20 ± 0.03 |
| | Logistics | 0% | 0% | 54 | 6 | 9.83 ± 0.10 | 10.00 ± 0.00 | 9.83 ± 0.10 | 10.00 ± 0.00 | 2.00 ± 0.00 | 2.00 ± 0.00 | 892.68 ± 22.73 | 903.62 ± 22.72 |
| | | 0% | 25% | 52 | 8 | 6.83 ± 0.11 | 7.00 ± 0.00 | 9.83 ± 0.11 | 10.00 ± 0.00 | 2.12 ± 0.30 | 2.00 ± 0.45 | 902.31 ± 22.47 | 900.95 ± 22.82 |
| | | 25% | 0% | 45 | 15 | 7.84 ± 0.11 | 7.87 ± 0.19 | 9.84 ± 0.11 | 9.87 ± 0.19 | **2.13 ± 0.19** | **1.73 ± 0.33** | 884.55 ± 24.70 | 903.10 ± 24.98 |
| | | 50% | 0% | 49 | 11 | 6.86 ± 0.10 | 6.82 ± 0.27 | 9.86 ± 0.10 | 9.82 ± 0.27 | **2.18 ± 0.27** | **1.55 ± 0.35** | 893.55 ± 24.43 | 917.98 ± 22.67 |
| | | 50% | 25% | 47 | 13 | 5.28 ± 0.24 | 5.08 ± 0.39 | 9.83 ± 0.11 | 9.92 ± 0.17 | **2.46 ± 0.58** | **1.31 ± 0.29** | **888.69 ± 27.90** | **923.47 ± 20.64** |
| **A+F** : Action and Fluent Observations | Block-Words | 0% | 0% | 102 | 81 | 4.65 ± 0.25 | 4.51 ± 0.41 | 8.69 ± 0.43 | 8.40 ± 0.61 | **3.41 ± 0.53** | **2.57 ± 0.34** | **61.95 ± 3.71** | **112.73 ± 4.74** |
| | | 0% | 25% | 44 | 132 | 3.16 ± 0.41 | 1.98 ± 0.17 | 9.00 ± 0.71 | 8.45 ± 0.42 | **6.36 ± 0.80** | **2.36 ± 0.30** | **38.17 ± 3.33** | **144.43 ± 6.44** |
| | | 25% | 0% | 99 | 84 | 4.44 ± 0.22 | 4.06 ± 0.30 | 8.73 ± 0.44 | 8.36 ± 0.59 | **3.50 ± 0.46** | **2.77 ± 0.34** | **59.14 ± 3.44** | **140.76 ± 6.74** |
| | | 50% | 0% | 89 | 94 | 4.10 ± 0.24 | 3.27 ± 0.26 | 9.15 ± 0.45 | 8.00 ± 0.54 | **3.70 ± 0.47** | **2.49 ± 0.29** | **53.20 ± 3.09** | **165.64 ± 7.57** |
| | | 50% | 25% | 41 | 124 | 2.51 ± 0.29 | 2.02 ± 0.17 | 8.63 ± 0.59 | 8.81 ± 0.46 | **6.30 ± 0.75** | **2.35 ± 0.33** | **36.13 ± 3.03** | **190.01 ± 8.16** |
| | Ipc-Grid | 0% | 0% | 79 | 11 | 6.76 ± 0.57 | 7.18 ± 1.34 | 13.25 ± 1.01 | 14.73 ± 2.44 | 2.00 ± 0.00 | 2.00 ± 0.00 | **4.16 ± 0.83** | **1.23 ± 0.24** |
| | | 0% | 25% | 62 | 27 | 3.71 ± 0.44 | 2.67 ± 0.56 | 13.53 ± 1.19 | 13.33 ± 1.53 | **3.26 ± 0.97** | **1.37 ± 0.19** | **2.76 ± 0.58** | **1.60 ± 0.42** |
| | | 25% | 0% | 73 | 17 | 6.03 ± 0.52 | 6.82 ± 1.18 | 12.86 ± 1.01 | 15.88 ± 2.15 | **2.00 ± 0.00** | **1.76 ± 0.22** | **3.98 ± 0.80** | **1.81 ± 0.40** |
| | | 50% | 0% | 71 | 19 | 5.52 ± 0.50 | 5.84 ± 1.21 | 13.13 ± 1.06 | 14.58 ± 2.00 | **2.26 ± 0.39** | **1.79 ± 0.20** | **3.76 ± 0.75** | **2.34 ± 0.65** |
| | | 50% | 25% | 51 | 31 | 3.18 ± 0.38 | 3.06 ± 0.54 | 13.61 ± 1.27 | 14.52 ± 1.30 | **3.13 ± 0.82** | **1.58 ± 0.18** | 3.06 ± 0.64 | 2.59 ± 0.61 |
| | Navigation | 0% | 0% | 54 | 9 | 9.17 ± 1.49 | 8.33 ± 4.25 | 17.41 ± 2.79 | 17.56 ± 9.35 | 2.56 ± 1.02 | 2.44 ± 1.09 | **0.18 ± 0.02** | **0.23 ± 0.03** |
| | | 0% | 25% | 48 | 14 | 4.56 ± 0.95 | 4.07 ± 1.61 | 16.73 ± 2.91 | 20.14 ± 6.89 | **3.07 ± 0.86** | **1.79 ± 0.79** | 0.18 ± 0.02 | 0.20 ± 0.04 |
| | | 25% | 0% | 56 | 7 | 8.02 ± 1.31 | 9.29 ± 5.92 | 17.27 ± 2.71 | 18.71 ± 12.51 | 3.14 ± 1.35 | 2.57 ± 1.68 | **0.17 ± 0.01** | **0.20 ± 0.03** |
| | | 50% | 0% | 57 | 6 | 7.60 ± 1.14 | 8.17 ± 6.28 | 17.16 ± 2.67 | 20.00 ± 15.12 | 2.50 ± 0.88 | 1.67 ± 0.54 | **0.17 ± 0.01** | **0.20 ± 0.03** |
| | | 50% | 25% | 39 | 21 | 3.95 ± 0.81 | 4.52 ± 1.12 | 16.21 ± 3.14 | 20.62 ± 5.36 | **2.86 ± 0.52** | **1.19 ± 0.18** | **0.16 ± 0.01** | **0.19 ± 0.01** |
| | Logistics | 0% | 0% | 55 | 5 | 10.13 ± 0.43 | 8.60 ± 0.68 | 19.25 ± 0.20 | 19.80 ± 0.56 | 2.00 ± 0.00 | 1.80 ± 0.56 | 895.50 ± 22.44 | 913.19 ± 21.46 |
| | | 0% | 25% | 35 | 25 | 5.71 ± 0.54 | 4.68 ± 0.70 | 19.23 ± 0.24 | 19.40 ± 0.32 | **2.60 ± 0.46** | **1.32 ± 0.26** | **862.35 ± 27.86** | **901.43 ± 26.05** |
| | | 25% | 0% | 52 | 8 | 9.10 ± 0.41 | 10.25 ± 1.72 | 19.25 ± 0.20 | 19.62 ± 0.62 | 2.00 ± 0.00 | 1.75 ± 0.39 | 891.18 ± 22.76 | 910.72 ± 22.88 |
| | | 50% | 0% | 47 | 13 | 7.94 ± 0.37 | 8.00 ± 0.55 | 19.26 ± 0.22 | 19.46 ± 0.31 | **2.08 ± 0.17** | **1.54 ± 0.31** | **890.45 ± 23.33** | **933.17 ± 23.85** |
| | | 50% | 25% | 37 | 23 | 4.89 ± 0.37 | 4.26 ± 0.68 | 19.30 ± 0.23 | 19.30 ± 0.33 | **2.48 ± 0.45** | **1.22 ± 0.29** | **866.39 ± 32.33** | **930.81 ± 22.57** |

**U%** is percent of observations placed in an unordered set. **D%** is percent of 'ambiguated' observations. We distinguish between samples perfectly solved by the ignore strategy (Opt) and samples with room for improvement (Imp). $|\Theta|$ is the observation set size for the specified method and sample group. $|\mathcal{G}^*|$ is the size of the solution set over the improvable (Imp) samples. Bold indicates a t-test significant difference ($\alpha <$.05). Underlines indicate the smaller of two means. Values with error rates are means with a 95% confidence interval.

case were we less accurate, empirically confirming Theorem 3.

This considered, our method is consistently slower across domains, regardless of improvement. We hypothesize that this is due to a larger search space. Using more observations means including more actions in the planning domain, which usually takes longer to compute. This time is highly domain-dependent. For instance, Logistics takes hundreds of seconds while Ipc-Grid takes under a second.

For all domains except Block-Words, the number of instances where we could improve (*i.e.*, $|G^*_{ign}| \neq 1$) was too small to make significant conclusions. This brings up the concept

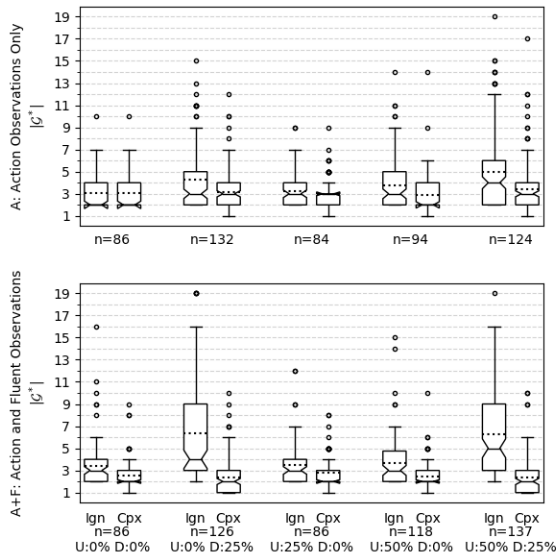


Figure 3: Comparison of solution set sizes $|G^*_{ign}|$ and $|G^*_{cpx}|$, from samples where improvement was possible. A solution set size of 1 is optimal.

of goal recognition difficulty. What makes Block-Words more difficult than the other domains? The other domains have, on average, larger observation sets to work with, derived from longer plans. Is it the number of observations available? Are the possible goals in its $\mathcal{G}$ more similar? If so, what makes them similar? Goal Recognition difficulty is not necessarily tied to planning difficulty. The Logistics domain took extraordinarily long compared to the Ipc-Grid and Navigation domains, yet all found the optimal solution set most of the time.

In future work, we wish to reevaluate with more coverage over more settings to pinpoint those settings where a domain becomes ‘easy’, as measured by how often the optimal solution set is found.

# 7 Conclusion

In applications with plentiful information or few complex observations, ignoring complexity may be preferred for faster results with little loss of answer quality. However, in areas with sparse information, more complex observations, or in domains known to be difficult, using complex information is vital, even if it takes longer to compute.

Our definitions for new observation types can be used for any goal recognition approach, and our compilation can be adapted for other planning-based approaches. In particular, we are interested in adapting this compilation for probabilistic plan recognition and multi-agent plan recognition.

For goal recognition to be used broadly, it needs to handle all types of information handed to it. From detective robots to ambiguous words in natural language, complex observations can come from many real-world scenarios, and this method lays the groundwork for leveraging them. We provide crisp definitions for partial-order ambiguous observations of both fluents and actions, then prove that our compilation produces satisfactory plans. While this work deals only with optimal solutions, this work can be extended to work with probabilistic goal recognition.

# References


[1] Sandra Carberry. *Plan recognition in natural language dialogue*. MIT press, Cambridge, MA, 1990.

[2] Rogelio E. Cardona-Rivera and R. Michael Young. Symbolic plan recognition in interactive narrative environments. In *Proceedings of the Joint Workshop on Intelligent Narrative Technologies and Social Believability in Games at the 11th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 16–22, Santa Cruz, CA, USA, November 2015.

[3] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[4] Neal Lesh, Charles Rich, and Candace L Sidner. Using plan recognition in human-computer collaboration. In *UM99 User Modeling*, pages 23–32. Springer Science, Berlin, Germany, 1999.

[5] William Pentney, Ana-Maria Popescu, Shiaokai Wang, Henry Kautz, and Matthai Philipose. Sensor-based understanding of daily life via large-scale use of common sense. In *Proceedings of the 21st national conference on Artificial intelligence-Volume 1*, pages 906–912, Boston, MA, USA, July 2006.

[6] Miguel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, Atlanta, GA, USA, July 2010.

[7] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *Twenty-First International Joint Conference on Artificial Intelligence*, Pasadena, CA, USA, July 2009.

[8] Shirin Sohrabi, Octavian Udrea, and Anton Riabov. Hypothesis exploration for malware detection using planning. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, Bellevue, WA, USA, July 2013.

[9] Shirin Sohrabi, Anton V Riabov, and Octavian Udrea. Plan recognition as planning revisited. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 63, New York, NY, USA, July 2016.

[10] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P. Goldman. *Plan, Activity, and Intent Recognition: Theory and Practice*. Morgan Kaufmann, Burlington, MA, 2014.

[11] Kartik Talamadupula, Gordon Briggs, Tathagata Chakraborti, Matthias Scheutz, and Subbarao Kambhampati. Coordination in human-robot teams using mental modeling and plan recognition. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2957–2962, Chicago, IL, USA, September 2014. IEEE.

[12] Ingrid Zukerman and Diane Litman. Natural language processing and user modeling: Synergies and limitations. *User modeling and user-adapted interaction*, 11(1-2):129–158, March 2001.

# Appendix A
# Sample PDDL

## A.1 Original DetectiveBot Domain

```
(define (domain detectiveBot)
    (:requirements
        :negative-preconditions
    )
    (:predicates
        (outside)
        (in-office)
        (in-backroom)
        (money-in-drawer)
        (key-in-drawer)
        (chest-empty)
        (holding-money)
        (holding-key)
        (holding-contents)
        (chest-unlocked)
        (contents-destroyed)
        (window-opened)
    )

    (:action enter-building
        :parameters ()
        :precondition
        (and
            (outside)
        )
        :effect
        (and
            (not (outside))
            (in-office)
        )
    )

    (:action enter-backroom
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-office)
        )
        :effect
```

```
        (and
            (in-backroom)
            (not (in-office))
        )
    )

    (:action enter-office
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-backroom)
        )
        :effect
        (and
            (in-office)
            (not (in-backroom))
        )
    )
    (:action exit-building
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-backroom)
        )
        :effect
        (and
            (not (in-backroom))
            (outside)
        )
    )

    (:action take-money
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-office)
            (money-in-drawer)
        )
        :effect
        (and
            (not (money-in-drawer))
            (holding-money)
        )
    )
    (:action take-key
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-office)
            (key-in-drawer)
```

```
        )
        :effect
        (and
            (not (key-in-drawer))
            (holding-key)
        )
    )

    (:action unlock-chest
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-backroom)
            (holding-key)
        )
        :effect
        (and
            (chest-unlocked)
        )
    )

    (:action take-contents-from-chest
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-backroom)
            (chest-unlocked)
            (not (chest-empty))
        )
        :effect
        (and
            (holding-contents)
            (chest-empty)
        )
    )

    (:action throw-out-window
        :parameters ()
        :precondition
        (and
            (not (outside))
            (in-backroom)
            (holding-contents)
        )
        :effect
        (and
            (not (holding-contents))
            (contents-destroyed)
            (window-opened)
        )
    )
)
```

## A.2 Compiled DetectiveBot Domain

```
(define
  (domain grounded-DETECTIVEBOT)
  (:requirements :strips :action-costs)
  (:predicates
    ( NOT-OUTSIDE )
    ( IN-OFFICE )
    ( IN-BACKROOM )
    ( HOLDING-MONEY )
    ( HOLDING-KEY )
    ( CHEST-UNLOCKED )
    ( HOLDING-CONTENTS )
    ( CHEST-EMPTY )
    ( CONTENTS-DESTROYED )
    ( WINDOW-OPENED )
    ( NOT-CHEST-EMPTY )
    ( KEY-IN-DRAWER )
    ( MONEY-IN-DRAWER )
    ( OUTSIDE )
    ( OBSERVATION_0 )
    ( MUTEX_1 )
    ( OBSERVATION_2 )
    ( OBSERVATION_3 )
    ( OBSERVATION_4 )
    ( OBSERVATION_5 )
  )
  (:functions (total-cost))
  (:action EXPLAIN_OBS_ENTER-BUILDING
    :parameters ()
    :precondition
    (and
      ( not ( OBSERVATION_0 ) )
      ( OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( NOT-OUTSIDE )
      ( IN-OFFICE )
      (not ( OUTSIDE ))
      ( OBSERVATION_0 )
    )
  )
  (:action EXPLAIN_OBS_TAKE-KEY
    :parameters ()
    :precondition
    (and
      ( not ( MUTEX_1 ) )
      ( KEY-IN-DRAWER )
      ( IN-OFFICE )
      ( NOT-OUTSIDE )
      ( OBSERVATION_0 )
    )
```

```
    :effect
    (and
      (increase (total-cost) 1)
      ( HOLDING-KEY )
      (not ( KEY-IN-DRAWER ))
      ( MUTEX_1 )
    )
  )
  (:action EXPLAIN_OBS_TAKE-MONEY
    :parameters ()
    :precondition
    (and
      ( not ( MUTEX_1 ) )
      ( MONEY-IN-DRAWER )
      ( IN-OFFICE )
      ( NOT-OUTSIDE )
      ( OBSERVATION_0 )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( HOLDING-MONEY )
      (not ( MONEY-IN-DRAWER ))
      ( MUTEX_1 )
    )
  )
  (:action EXPLAIN_OBS_ENTER-BACKROOM
    :parameters ()
    :precondition
    (and
      ( not ( OBSERVATION_2 ) )
      ( IN-OFFICE )
      ( NOT-OUTSIDE )
      ( MUTEX_1 )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( IN-BACKROOM )
      (not ( IN-OFFICE ))
      ( OBSERVATION_2 )
    )
  )
  (:action EXPLAIN_OBSERVATION_3
    :parameters ()
    :precondition
    (and
      ( not ( OBSERVATION_3 ) )
      ( WINDOW-OPENED )
      ( OBSERVATION_2 )
    )
    :effect
    (and
      (increase (total-cost) 0 )
```

```
      ( OBSERVATION_3 )
    )
  )
  (:action EXPLAIN_OBSERVATION_4
    :parameters ()
    :precondition
    (and
      ( not ( OBSERVATION_4 ) )
      ( CHEST-EMPTY )
      ( OBSERVATION_2 )
    )
    :effect
    (and
      (increase (total-cost) 0 )
      ( OBSERVATION_4 )
    )
  )
  (:action EXPLAIN_OBS_EXIT-BUILDING
    :parameters ()
    :precondition
    (and
      ( not ( OBSERVATION_5 ) )
      ( IN-BACKROOM )
      ( NOT-OUTSIDE )
      ( OBSERVATION_2 )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( OUTSIDE )
      (not ( IN-BACKROOM ))
      (not ( NOT-OUTSIDE ))
      ( OBSERVATION_5 )
    )
  )
  (:action THROW-OUT-WINDOW
    :parameters ()
    :precondition
    (and
      ( HOLDING-CONTENTS )
      ( IN-BACKROOM )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( CONTENTS-DESTROYED )
      ( WINDOW-OPENED )
      (not ( HOLDING-CONTENTS ))
    )
  )
  (:action TAKE-CONTENTS-FROM-CHEST
    :parameters ()
    :precondition
```

```
    (and
      ( NOT-CHEST-EMPTY )
      ( CHEST-UNLOCKED )
      ( IN-BACKROOM )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( HOLDING-CONTENTS )
      ( CHEST-EMPTY )
      (not ( NOT-CHEST-EMPTY ))
    )
  )
  (:action UNLOCK-CHEST
    :parameters ()
    :precondition
    (and
      ( HOLDING-KEY )
      ( IN-BACKROOM )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( CHEST-UNLOCKED )
    )
  )
  (:action TAKE-KEY
    :parameters ()
    :precondition
    (and
      ( KEY-IN-DRAWER )
      ( IN-OFFICE )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( HOLDING-KEY )
      (not ( KEY-IN-DRAWER ))
    )
  )
  (:action TAKE-MONEY
    :parameters ()
    :precondition
    (and
      ( MONEY-IN-DRAWER )
      ( IN-OFFICE )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
```

```
      ( HOLDING-MONEY )
      (not ( MONEY-IN-DRAWER ))
    )
  )
  (:action EXIT-BUILDING
    :parameters ()
    :precondition
    (and
      ( IN-BACKROOM )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( OUTSIDE )
      (not ( IN-BACKROOM ))
      (not ( NOT-OUTSIDE ))
    )
  )
  (:action ENTER-OFFICE
    :parameters ()
    :precondition
    (and
      ( IN-BACKROOM )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( IN-OFFICE )
      (not ( IN-BACKROOM ))
    )
  )
  (:action ENTER-BACKROOM
    :parameters ()
    :precondition
    (and
      ( IN-OFFICE )
      ( NOT-OUTSIDE )
    )
    :effect
    (and
      (increase (total-cost) 1)
      ( IN-BACKROOM )
      (not ( IN-OFFICE ))
    )
  )
  (:action ENTER-BUILDING
    :parameters ()
    :precondition
    (and
      ( OUTSIDE )
    )
    :effect
```

```
    (and
      (increase (total-cost) 1)
      ( NOT-OUTSIDE )
      ( IN-OFFICE )
      (not ( OUTSIDE ))
    )
  )

)
```

## A.3 Compiled DetectiveBot Problem

Compiled assuming `contents-destroyed` ∧ `outside` is the culprit's goal.

```
(define
  (problem grounded-DESTROY_AND_LEAVE)
  (:domain grounded-DETECTIVEBOT)
  (:init
    (= (total-cost) 0)
    ( NOT-CHEST-EMPTY )
    ( OUTSIDE )
    ( MONEY-IN-DRAWER )
    ( KEY-IN-DRAWER )
  )
  (:goal
    (and
    ( CONTENTS-DESTROYED )
    ( OUTSIDE )
    ( MUTEX_1 )
    ( OBSERVATION_0 )
    ( OBSERVATION_2 )
    ( OBSERVATION_3 )
    ( OBSERVATION_4 )
    ( OBSERVATION_5 )
    )
  )
  (:metric minimize (total-cost))
)
```