Journal of Artificial Intelligence Research 65 (2019) 209-269 Submitted 10/2018; published 07/2019

Goal Recognition Design in Deterministic Environments

Sarah Keren SKEREN@SEAS.HARVARD.EDU
Harvard University
School of Engineering and Applied Sciences
Cambridge, Massachusetts 02138, USA

Avigdor Gal AVIGAL@IE.TECHNION.AC.IL
Erez Karpas KARPASE@TECHNION.AC.IL
Technion — Israel Institute of Technology
Haifa 3200003, Israel

Abstract

Goal recognition design (*GRD*) facilitates understanding the goals of acting agents through the analysis and redesign of goal recognition models, thus offering a solution for assessing and minimizing the maximal progress of any agent in the model before goal recognition is guaranteed. In a nutshell, given a model of a domain and a set of possible goals, a solution to a *GRD* problem determines (1) the extent to which actions performed by an agent within the model reveal the agent's objective; and (2) how best to modify the model so that the objective of an agent can be detected as early as possible. This approach is relevant to any domain in which rapid goal recognition is essential and the model design can be controlled. Applications include intrusion detection, assisted cognition, computer games, and human-robot collaboration.

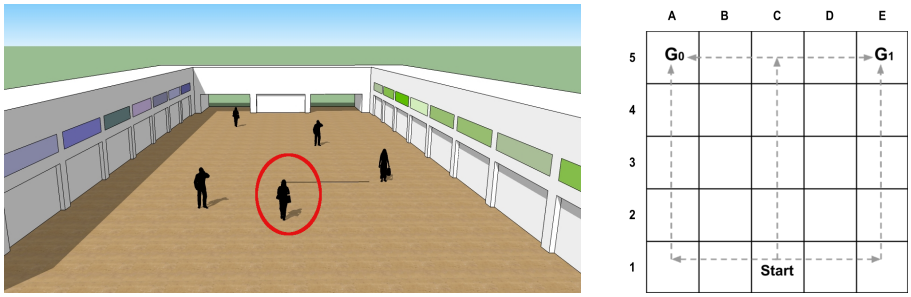
A *GRD* problem has two components: the analyzed goal recognition setting, and a design model specifying the possible ways the environment in which agents act can be modified so as to facilitate recognition. This work formulates a general framework for *GRD* in deterministic and partially observable environments, and offers a toolbox of solutions for evaluating and optimizing model quality for various settings.

For the purpose of evaluation we suggest the *worst case distinctiveness* (*WCD*) measure, which represents the maximal cost of a path an agent may follow before its goal can be inferred by a goal recognition system. We offer novel compilations to classical planning for calculating *WCD* in settings where agents are bounded-suboptimal. We then suggest methods for minimizing *WCD* by searching for an optimal redesign strategy within the space of possible modifications, and using pruning to increase efficiency. We support our approach with an empirical evaluation that measures *WCD* in a variety of *GRD* settings and tests the efficiency of our compilation-based methods for computing it. We also examine the effectiveness of reducing *WCD* via redesign and the performance gain brought about by our proposed pruning strategy.

1. Introduction

Goal recognition aims at discovering the goals of an agent based on observations of that agent's actions using data collected online (Kautz, 1987; Carberry, 2001; Ramirez & Geffner, 2010; Sukthankar, Geib, Bui, Pynadath, & Goldman, 2014).[1] We investigate the problem of

1. In the remainder of this paper we refer to agents interchangeably as "he," "she," or "it," depending on the context.



(a) Airport *GRD* illustration (b) Schematic *GRD* problem

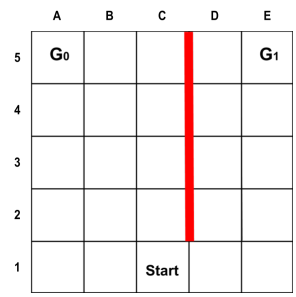
Figure 1: An example of a *GRD* problem

goal recognition design (*GRD*), that involves the analysis and optimization of goal recognition models. A *GRD* problem includes a description of a goal recognition model with a set of possible goals and a description of the available ways to modify the model. The analysis consists of two main stages: evaluation and optimization. In the evaluation stage we assess to what extent the goal of an agent acting in the system may remain unrecognized, while the optimization stage consists of finding the best way to modify the model so as to improve recognition.

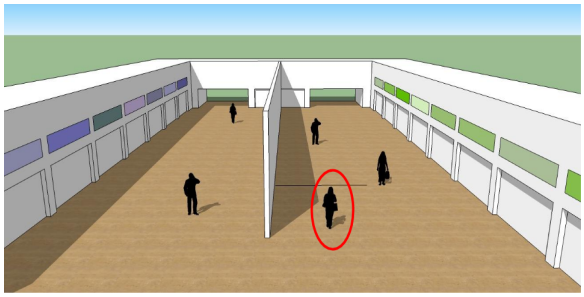
This research is motivated by developments in data science, which now provides tools for efficiently and effectively gathering, managing, analyzing, and visualizing user data. In turn, emerging applications in various domains, from urban transportation to medical informatics, require systems to automatically and rapidly analyze agents' behavior, identify their goals, and then help them reach (or, depending on the application, prevent them from reaching) those goals. In fact, *GRD* is relevant to any domain where rapid goal recognition is essential and where the model design can be controlled. Potential applications include intrusion detection (Jarvis, Lunt, & Myers, 2004; Kaluza, Kaminka, & Tambe, 2011; Boddy, Gohde, Haigh, & Harp, 2005), assisted cognition (Kautz, Etzioni, Fox, Weld, & Shastri, 2003), computer games (Kabanza, Bellefeuille, Bisson, Benaskeur, & Irandoust, 2010; Albrecht, Zukerman, & Nicholson, 1998; Ha, Rowe, Mott, & Lester, 2011), and human-robot collaboration (Levine & Williams, 2014; Freedman & Zilberstein, 2017).

Example 1 *A simple* GRD *problem is presented in Figure 1. We assume that, for security reasons, airport officials wish to track the movements of passengers (Figure 1(a)). The model consists of a simple room with a single entry point (Start) and two possible exit points, marked as G_0 (boarding gates for domestic flights) and G_1 (international flights). Agents can move vertically or horizontally from Start to either of these goals. Under the assumption that agents behave optimally, for each goal there are several possible paths (a subset of possible paths is marked by dashed lines in Figure 1(b)). As can be seen, paths to different goals may share a common prefix. In this model, the agent's goal becomes clear once she turns left or right.*

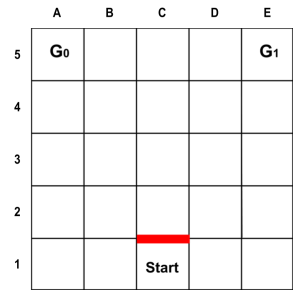
One measure of model quality (in terms of goal recognition) is how long an agent can operate in the model without observers being able to identify its goal; the longer the agent can operate, the worse the model. The *worst case distinctiveness* (*WCD*) measure represents the maximal cost a plan may accrue before an agent's goal can be recognized (when actions



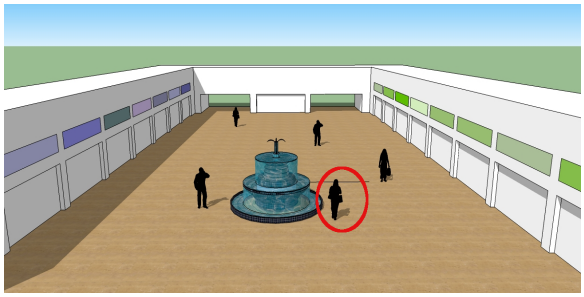
(a) A non-optimal solution



(b) Airport non-optimal solution



(c) An optimal solution



(d) Airport optimal solution

Figure 2: Improved *GRD* model

have uniform costs, the *WCD* counts the maximal number of actions an agent may perform before its goal becomes known). In Figure 1(a), the *WCD* is illustrated by the circled user, who can walk all the way up to the opposite side of the terminal (four steps in Figure 1(b)) before revealing her true intention.

Defining the quality of a model allows us to improve it. Towards this end, we need to define the means available for modifying a model. One possible way of introducing change to a model is by limiting the set of available actions an agent can perform. To maintain user comfort, we can require our solution to preserve the original solution cost for all goals. In addition, we wish to attain the maximal achievable reduction in *WCD* while minimizing the changes introduced and respecting any design constraints that may be specified. To illustrate, assume now that airport managers can place barriers in the terminal to control the flow of passengers (a common and effective solution for passenger control), but also wish to minimize obstruction to the ease of use of the terminal. Figure 2 presents two possible solutions, both of which reduce the *WCD* from 4 to 0 without increasing the minimal cost to goal. Assuming agents act optimally, the option in Figure 2(c) is intuitively preferable since it offers the same result by disallowing a single action and creating a single barrier.

Consider now a setting where we replace the assumption of agent optimality with a relaxed bounded suboptimality assumption, allowing agents to diverge from an optimal plan by at most two steps. The *WCD* in this case is 6, and the optimal design solution suggested above will leave the *WCD* unchanged. As depicted in Figure 3, the *WCD* can be reduced to 0 by using three barriers.

The discussion so far has assumed that agents are fully observable to the goal recognition system. However, many real-world applications must account for various forms of

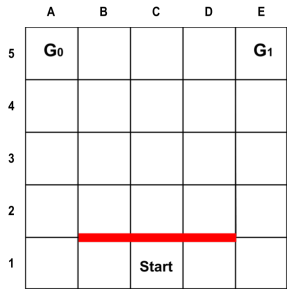
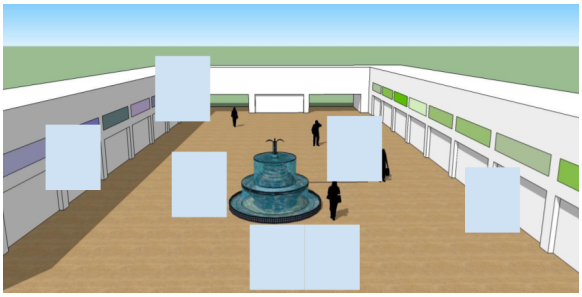
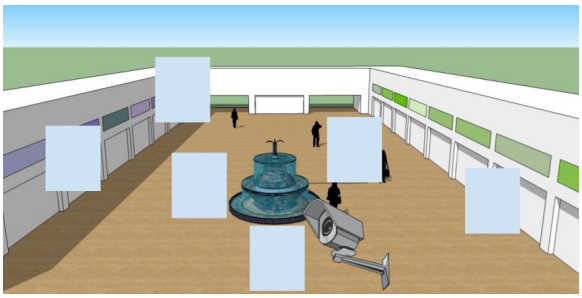


Figure 3: The airport design example with suboptimal agents



(a) Partially observable setting



(b) *WCD* reduction using sensors

Figure 4: *GRD* with non-observable actions (gray cells indicate missing sensors)

uncertainty. In particular, goal recognition systems often suffer from reduced and noisy observability due to lack of suitable sensors, insufficient sensor coverage, faulty sensors, inaccurate measurements, etc. Note that whereas in the fully observable setting goal recognition is hampered only if the agent's behavior could fit more than one goal, when observability is partial the agent's goal can remain unrecognized even if its behavior is goal-specific.

Example 2 *The setting depicted in Figure 4 differs from the one described in Example 1 by accounting for partial sensor coverage (a move action ending in a blank cell is non-observable). As can be seen, the modification applied to the fully observable setting does not offer the same benefit in the partially observable setting. Even with the obstacle placed in front of the entry point, optimal agents may advance one step before their goal is revealed (Figure 4(a)) – i.e.,* WCD = 1. *In Figure 4(b), a sensor is placed to the right of the entry*

point, thus guaranteeing recognition at the first step (to the left or right) and setting WCD *to* 0.

The ideas presented above can be applied to a variety of goal recognition settings. For example, consider a smart home in which the activity of a user with some physical or mental disability is tracked so as to help the user perform daily activities and avoid dangers. In such a setting, the environment may need to be redesigned (e.g., by repositioning furniture) so that the goal recognition system can detect dangerous situations as early as possible (e.g., if the user approaches a hot oven).

This work formulates a general framework for *GRD* in deterministic domains, and offers a toolbox of solutions for assessing and optimizing model quality. It continues and extends five previous conference papers that explored various aspects of the *GRD* problem. Our initial work on *GRD* (Keren, Gal, & Karpas, 2014) puts forward a model based on three simplifying assumptions, namely that the environment is fully observable both to the goal recognition system and agents; that the outcomes of agent actions are deterministic; and that agents, who are agnostic to the goal recognition system, act optimally. The model is modified by disallowing (removing) actions from the set of applicable actions. In Keren, Gal, and Karpas (2015) we account for suboptimal agents who have a budget for diverting from optimal plans. In this setting, agents behave suboptimally either naïvely, by following any plan within the specified bound, or intentionally, by following prefixes of optimal paths to other goals, thus obfuscating their true goal as far as possible within the cost bound. In Keren, Gal, and Karpas (2016a) we extend *GRD* to account for goal recognition systems with partial sensor coverage. In this setting, actions can be either observable or non-observable, and sensor placement is proposed to improve recognition. In Keren, Gal, and Karpas (2016b) we generalize the sensor model to account for non-deterministic and noisy sensor models. Sensor refinement – i.e., improving the system's sensor resolution – is added to the set of possible ways to modify the model. To assess these settings, novel compilations from *GRD* to classical planning are proposed. A breadth-first search (BFS) is used to identify optimal redesign strategies, and pruning is suggested as a way to increase efficiency. Finally, in Keren, Gal, and Karpas (2018), we focus on the redesign process and formulate it as a search within the space of modifications. The pruning approach suggested previously is extended and elaborated by specifying the conditions under which the pruning strategy is safe (i.e., such that at any point in the search, an optimal solution can be found in the unpruned search space). This positioning allows us to generalize the *GRD* framework and enrich the toolbox of redesign methods that can be applied to a goal recognition setting.

The extended *GRD* framework presented here provides three key contributions. First, we offer a comprehensive account of the theoretical framework suggested previously. This includes a complete description of the methods we apply and detailed proofs of the theorems we use to justify them. Second, we account for arbitrary agent action costs and specify a compilation-based *WCD* calculation method for this setting. Finally, in our empirical evaluation we analyze a variety of *GRD* models and examine different redesign settings for each problem, including new benchmarks that have not been examined previously.

In the remainder of this paper, we first review the necessary background on goal recognition and automated planning (Section 2). In Section 3, we formulate the *GRD* task. In section 4, we present methods developed for calculating *WCD* in settings where agents are

bounded-suboptimal. In section 5, we present methods for minimizing *WCD*. In Section 6 we provide an empirical evaluation of our methods, while in Section 7 we discuss related work. Finally, in Section 8, we summarize our contributions and suggest directions for future work.

2. Background

GRD creates a framework for analyzing various goal recognition settings. In this section, we first provide a brief overview of automated planning, which underlies the goal recognition settings we support in this work and most of the tools we have developed to solve the *GRD* problem. We then describe the goal recognition task and its relationship to automated planning.

2.1 Automated Planning

The basic form of automated planning, referred to as *classical planning*, is a model in which agents' actions are fully observable and deterministic. A common way to represent classical planning problems is by using the STRIPS formalism (Fikes & Nilsson, 1972): $P = \langle F, I, A, G, C \rangle$, where F is a set of fluents (and a state s is represented by the fluents that are true in s), $I \subseteq F$ is the initial state, $G \subseteq F$ represents the set of goal states, and A is a set of actions. Each action is a triple $a = \langle pre(a), add(a), del(a) \rangle$, which represents the precondition, add, and delete lists respectively, all subsets of F. An action a is applicable in state s if $pre(a) \subseteq s$. If action a is applied in state s, it results in a new state $s' = (s \setminus del(a)) \cup add(a)$. $C : A \to \mathbb{R}_0^+$ is a cost function that assigns each action a non-negative cost.

The objective of a planning problem is to find a plan $\pi = \langle a_1, \ldots, a_n \rangle$, a sequence of actions that brings an agent from I to a goal state. The cost $c(\pi)$ of a plan π is $\Sigma_{i=1}^{n}(C(a_i))$. Often, the objective is to find an optimal solution for P, meaning an optimal plan π^* that minimizes the associated cost. In settings where the problem input includes actions with uniform costs, a plan's cost is equivalent to its length, and the optimal plan is the shortest one.

The literature is rich with different approaches developed to solve the planning problem (Geffner & Bonet, 2013). One popular approach views the problem as a directed graph and uses graph-search algorithms to identify the optimal plan. Heuristic estimations, extracted automatically from the problem description, are used to guide the search (Bonet & Geffner, 2001; Helmert & Domshlak, 2009). Specifically, admissible heuristics are guaranteed to underestimate the cost to goal. Using admissible heuristics to guide search algorithms that first explore paths with the lowest estimated cost (e.g., A^*, Hart, Nilsson, & Raphael, 1968) is guaranteed to produce optimal solutions.

2.2 Goal Recognition

Goal recognition is the online analysis of an agent's perceived behavior with the aim of identifying the agent's objective. This task is closely related to yet distinct from plan recognition, which aims at identifying the complete plan being followed by an agent to

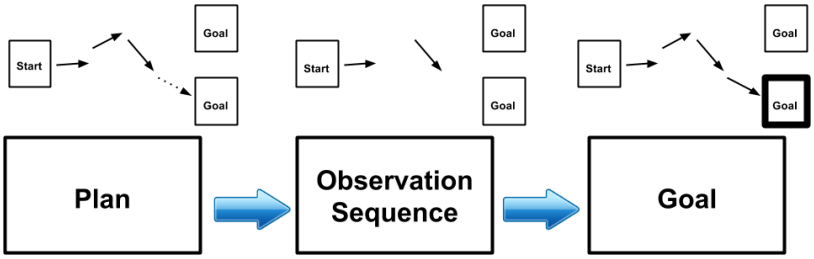


Figure 5: The goal recognition process

achieve its objective (Kautz & Allen, 1986; Cohen, Perrault, & Allen, 1981; Lesh & Etzioni, 1995; Ramirez & Geffner, 2009, 2010; Pattison & Long, 2010; Hong, 2001).[2]

Figure 5 illustrates a typical goal recognition setting. An agent enters the system at an initial state (labeled Start) and performs a sequence of actions that lead to a premeditated goal (represented in this case by the lower box marked Goal in the figure). The execution sequence emits an observation sequence that is perceived by the goal recognition system (hereafter referred to as the recognition system). The observation sequence, which is not necessarily either complete or accurate, is processed in order to reveal the agent's goal. Typically, the objective in such a setting is to identify the goal as early as possible.

A recognition setting can generally be classified as either *keyhole*, where the agent is unaware of or agnostic to the recognition process; *adversarial*, where the agent seeks to conceal its objective; and *intended*, where the agent helps the recognition system detect its objective (Carberry, 2001; Cohen et al., 1981). In this work we focus on the first of these settings, where the agent's behavior is not affected by the recognition process.

Due to its generic nature, goal recognition has been modeled and solved using various approaches. Examples include Bayesian networks (Bui, 2003; Han & Pereira, 2011), graph construction (Hong, 2001), and specialized procedures (Lesh & Etzioni, 1995). Most existing models rely on a specification of a plan library, which provides a representation of the set of plans an agent may execute to achieve each goal.

Despite the close relationship between goal recognition and automated planning, which aims to identify plans that lead to a desired goal, it was only less than a decade ago that Ramirez and Geffner (2009) established the connection between these fields. They present a compilation of plan recognition problems into *classical planning* problems that can be solved by any off-the-shelf planner. Several works followed and extended this approach by using various automated planning techniques to analyze and solve goal and plan recognition problems (Pattison & Long, 2010, 2011; Ramirez & Geffner, 2010, 2011; Yolanda, R-Moreno, Smith, et al., 2015; Sohrabi, Riabov, & Udrea, 2016; Freedman & Zilberstein, 2017; Pereira, Oren, & Meneguzzi, 2017; Kaminka, Vered, & Agmon, 2018).

In this work, we rely on models and tools from automated planning to model and solve *GRD* problems. Specifically, our approach uses planning-based tools to measure and minimize the maximal number of observations that need to be collected before recognition of an agent's goal is guaranteed.

2. See Sukthankar et al. (2014) for a recent survey of goal and plan recognition frameworks.

Table 1: Table of Notations for *GRD*

R	$\triangleq$	goal recognition model
D	$\triangleq$	design model
T	$\triangleq$	goal recognition design (GRD) model
$g \in G$	$\triangleq$	goal in a goal set
$\pi \in \Pi$	$\triangleq$	plan (full action sequence from an initial state to a goal) in a set of plans
$\vec{\pi} \in \vec{\Pi}$	$\triangleq$	path (plan prefix) in a path set
$\pi^{leg}(g) \subseteq \Pi^{leg}(G)$	$\triangleq$	the legal plans to goal g as a subset of all legal plans
$o \in O$	$\triangleq$	observation token in an observation token set
$o_\emptyset$	$\triangleq$	empty observation token (for non observable actions)
$\vec{o} \in \vec{O}$	$\triangleq$	observation sequence in a set of sequences
$op(\vec{\pi})$	$\triangleq$	the set of observable projections of path $\vec{\pi}$, i.e., the observation sequences that may be emitted when $\vec{\pi}$ is executed
$G^A(\vec{\pi})$	$\triangleq$	goals satisfied by path $\vec{\pi}$
$G^O(\vec{o})$	$\triangleq$	goals satisfied by observation sequence $\vec{o}$
$G^{\vec{O}}(\vec{\pi})$	$\triangleq$	goals satisfied by the observable projections of $\vec{\pi}$
$m \in \mathcal{M}$	$\triangleq$	atomic modification in a set of modifications
$\vec{m} \in \vec{\mathcal{M}}$	$\triangleq$	modification sequence in a set of sequences
ϕ	$\triangleq$	indicator of allowed modification sequences
δ	$\triangleq$	modification transition function
$app_\phi(R)$	$\triangleq$	the set of modifications applicable in R
$WCD(R)$	$\triangleq$	worst case distinctiveness (WCD) of goal recognition model R
$WCD^{min}(T)$	$\triangleq$	minimal WCD achievable in a GRD model T
$WCD_i(R)$	$\triangleq$	maximal non-distinctive path to goal g_i in goal recognition model R
$\mathcal{R}$	$\triangleq$	all goal recognition models
$\mathcal{R}^T$	$\triangleq$	all goal recognition models in GRD model T reachable from the initial goal recognition model R_0 via design

3. Goal Recognition Design (*GRD*)

Goal recognition design (*GRD*) analysis includes two key tasks. The first measures how efficiently and effectively the online goal recognition system performs in a given setting. The second optimizes the goal recognition setting via redesign. We devote this section to defining the *GRD* problem (Section 3.1) and the *worst case distinctiveness* (*WCD*) measure used to assess it (Section 3.2). In Section 3.3, we formulate the objective of the redesign process.

3.1 Model

The definition of a *GRD* problem has two components: the analyzed goal recognition setting, and a *design model* specifying possible ways to modify the goal recognition setting. After formulating the goal recognition task in Section 3.1.1 and the design model in Section 3.1.2, we integrate both components into the *GRD* model in Definition 5.

3.1.1 Goal Recognition Model

Typically, the definition of a goal recognition task (generally described in Section 2.2) includes a specific observation sequence that is analyzed (Ramirez & Geffner, 2009; Sohrabi et al., 2016; Pereira et al., 2017). The recognition task then involves mapping the perceived observation to a set of possible goals the agent may be trying to achieve.

In the context of *GRD*, we need to account for *all* possible sequences that may be observed. Accordingly, the *GRD* analysis needs to consider all aspects of the recognition setting that may affect goal recognition. To facilitate this analysis, in our definition we divide the goal recognition setting into three main elements, namely the *environment, agent strategy*, and *observability*.

The specification of the *environment* describes the dynamics of the setting in which agents act, including all aspects of the model that dictate the possible behaviors of agents within it. Instead of an explicit representation of agent plans, it is common to use a compact representation of the model that includes the set of possible goals G, the initial state I and the set of actions A that may be executed by an agent. In this context, a plan π is a sequence of actions that take an agent from the initial state to some goal. A path $\vec{\pi}$ is a plan prefix. While the actual representation formalism of the actions may vary, the description of each action includes its applicability, or the set of states in which the action may be applied; its possible outcomes; and its associated cost. This induces the set $\Pi(g)$ of possible plans to each goal g.

Given the set of possible plans to a goal, the *agent strategy* describes the set $\Pi^{leg}(g) \subseteq \Pi(g)$ of *legal plans* – i.e., the plans agents may choose to execute in order to achieve goal g. These plans are those allowed under the assumptions made about the behavior of the agent and how the agent chooses the action to execute at each stage. The set $\Pi^{leg}(G) = \bigcup_{g \in G} \Pi^{leg}(g)$ is the set of all legal plans for all goals. (See Table 1 for an account of our notations.)

The final element of a goal recognition model is the *observability* of agents and their actions. Observability as considered here describes how the activity of an agent is perceived by the recognition system, and is independent of how the agent perceives the environment.

Observability is defined via a *sensor model*, marked by S, which maps a path performed by an agent to an observation sequence $\vec{o}$ that may be emitted by the executed trajectory.

The sensor model can be expressed in various ways. The simplest sensor model corresponds to the fully observable setting, where the observation token emitted by the execution of an action is the unique action name. In such a setting, $\vec{o}$ corresponds to the actual performed path $\vec{\pi}$. A sensor model can also be expressed by specifying $S(a) \subseteq O$, where O is the set of observation tokens and $o \in S(a)$ means token o may be observed when a is performed (Geffner & Bonet, 2013). In this case, each path $\vec{\pi}$ is associated with a set of observation sequences that may be emitted by its execution (see Definition 2). The set $\vec{O}$ of possible observation sequences is therefore induced by the sensor model S and the set of legal plans $\Pi^{leg}(G)$. Given a goal recognition model and its sensor model, each path is associated with a set of goals *satisfied* by the observed behavior, i.e., the set of goals with legal paths that may produce the observed sequence.

We use the characterization above to specify the goal recognition models we support in this work, which comply with three underlying assumptions. The first is that system dynamics are deterministic; i.e., the outcome of each possible agent action is known in advance. The second states that an agent aiming at a given goal executes one of a predefined set of *legal plans* to that goal. The third states that while the agent's actions may be only partially observable to the recognition system, the agent has full knowledge of the state of the environment.

In accordance with the specification above, the agents in our setting are described using the classical planning model defined in Section 2.1, in which a planning domain is defined by the quadruple $\langle F, I, A, C \rangle$. As described in following sections, our framework supports the process of redesigning goal recognition models, and potentially modifying the agent's planning model. We therefore use universal sets of fluents and actions to describe the agent's models. Specifically, we let $\mathcal{F}$ represent the set of all fluents. In addition, we let $\mathcal{A}$ represent the set of all actions a, each represented by a triple $\langle pre(a), add(a), del(a) \rangle$, where $pre(a) \subseteq \mathcal{F}$ is the set of action preconditions, $add(a) \subseteq \mathcal{F}$ is the set of fluents a adds to the current state when executed, and $del(a) \subseteq \mathcal{F}$ is the delete list, representing the set of fluents a deletes from the current state.

Given that agent actions have deterministic outcomes and the state space is discrete, a plan $\pi = \langle a_0, ..., a_n \rangle, a_i \in A$ is a full execution that takes an agent from the initial state to a goal state and a *path* $\vec{\pi} = \langle a_0, ..., a_i \rangle, i \leq n$ is an execution sequence that is a prefix of a plan. Each path $\vec{\pi}$ emits a (possibly empty) observation sequence $\vec{o} = \langle o_1, .., o_j \rangle$, which is a sequence of observation tokens o_i perceived by the recognition system.

The definition of a goal recognition model is given below.

Definition 1 *A goal recognition model R is represented by the tuple*

$$R = \langle F, I, A, C_A, G, leg, O, S \rangle$$

where:

- $F \subseteq \mathcal{F}$ *is a set of fluents.*
- $I \subseteq F$ *is the initial state.*
- $A \subseteq \mathcal{A}$ *is a set of actions.*

- $C_A : \mathcal{A} \rightarrow \mathbb{R}$ *specifies the non-negative agent cost of performing each action. The cost of a path* $\vec{\pi} = \langle a_1, \ldots a_n \rangle$ *is the aggregated cost of its components* $\mathcal{C}_A(\vec{\pi}) = \sum_{i=1}^{n} C_A(a_i)$.
- G *is a set of possible goals* g *s.t.* $|G| \geq 2$ *and* $g \subseteq F$.
- $leg : \vec{\Pi} \times G \rightarrow \{0, 1\}$ *is an indicator that specifies the legal paths to each of the goals.*
- O *is a set of observation tokens, including the special observation token* $o_\emptyset$*, denoting that an action could be non-observable.*
- $S : \mathcal{A} \rightarrow 2^O \setminus \emptyset$ *is a sensor model, mapping each action* $a \in \mathcal{A}$ *into a set of observation tokens* $S(a) \subseteq O$ *that may be emitted when* a *is executed.*

We let $\mathcal{R}$ represent the set of goal recognition models that comply with Definition 1.

Corresponding to the elements described above that comprise a goal recognition model, the environment of a goal recognition model is described by the planning domain $\langle F, I, A, C_A \rangle$ and possible goals G. These induce the set of possible behaviors in the model. Specifically, they induce the set $\Pi(g)$ of plans to each goal g.

Agent strategy is described by the indicator leg, which, together with the planning domain $\langle F, I, A, C_A \rangle$ and goal set G, induce the set $\Pi^{leg}(g)$ of legal plans to each of the goals. This set can be described either explicitly or symbolically (e.g., the set of all optimal plans that do not make use of action a). An agent aiming at one of the goals in the set $g \in G$ enters the system at the initial state I and executes one of the legal plans $\pi \in \Pi^{leg}(g)$ from I to g. The set of legal paths $\vec{\Pi}^{leg}(g)$ is the set of prefixes of the plans in $\Pi^{leg}(g)$. Hereafter, since we assume agents follow only legal paths, whenever we refer to paths or plans we are implicitly referring to legal paths and legal plans, respectively.

Finally, observability is described by the sensor model S and token set O, which specify how each action is perceived by the recognition system. A state of partial observability implies a distinction between the agent's activity and how it is perceived by the system. When action a is performed by an agent, one of the possible observation tokens $o \in S(a)$ is emitted, with the special token $o_\emptyset$ denoting that the action is not observed by the recognition system.[3] Note that the sensor model refers to how the recognition system observes agent actions, while the world state is assumed to be fully observable to the agent. Also note that as opposed to typical goal recognition models (e.g., Ramirez & Geffner, 2009), which include a description of an observation sequence to be analyzed, the models we support require an account for the set $\vec{O}$ of all possible sequences that may be observed. Specifically, since each action a is mapped to a set of observation tokens $S(a) \subseteq O$ there are multiple possible observation sequences, any one of which could be emitted when a path is executed.

Next, we formally define the relationship between a path $\vec{\pi}$ (a prefix of a plan) and the set of observation sequences $op(\vec{\pi})$ it may emit.

3. The model presented above is a generalization of both the partially observable setting (Keren et al., 2016a), in which each action is mapped to either the empty token (non-observable) or to the action's name (observable), and the fully observable setting (Keren et al., 2014, 2015), in which each action is uniquely mapped to its name.

Definition 2 *Given a path $\vec{\pi} = \langle a_1, ..., a_n \rangle$, the set of possible* observable projections *of $\vec{\pi}$, denoted $op(\vec{\pi})$, is recursively defined as follows:*

$$op(\vec{\pi}) = \begin{cases} \langle\rangle & \vec{\pi} = \langle\rangle \\ S(a_1) \times op(\langle a_2, ..., a_n \rangle) & \vec{\pi} = \langle a_1, \ldots, a_n \rangle \wedge o_\emptyset \notin S(a_1) \\ (S(a_1) \setminus \{o_\emptyset\}) \times op(\langle a_2, ..., a_n \rangle) \bigcup op(\langle a_2, .., a_n \rangle) & \vec{\pi} = \langle a_1, \ldots, a_n \rangle \wedge o_\emptyset \in S(a_1) \end{cases}$$

The empty token $o_\emptyset$ is excluded from the observable projection of a path. This allows the model to account for settings where there is no way to know if and when some action has been performed.

3.1.2 Design Model

The design model, defined next, describes the modifications that can be applied to a goal recognition setting.

Definition 3 *A design model D is represented by the tuple*

$$D = \langle \mathcal{M}, \delta, \phi \rangle$$

where:

- *$\mathcal{M}$ is a finite set of atomic modifications a system can apply. A modification sequence is an ordered set of modifications $\vec{m} = \langle m_1, \ldots, m_n \rangle$ s.t. $m_i \in \mathcal{M}$ and $\vec{\mathcal{M}}$ is the set of all such sequences.*
- *$\delta : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{R}$ is a deterministic modification transition function, specifying the goal recognition model that results from applying a modification to a goal recognition model.*
- *$\phi : \vec{\mathcal{M}} \times \mathcal{R} \rightarrow \{0, 1\}$ is a constraint indicator that specifies the modification sequences that can be applied to a goal recognition model.*

The constraint indicator ϕ imposes the set of modifications that are applicable in a goal recognition model R. We represent this set as $app_\phi(R) = \{m \in \mathcal{M} \mid \phi(m, R) = 1\}$.

In general, each modification $m \in \mathcal{M}$ is associated with a design cost $C_D(m)$, and the cost of a sequence is the aggregated cost of its components ($\mathcal{C}_D(\vec{m}) = \sum_{i=1}^{n} C_D(m_i)$). For simplicity, we henceforth assume all modifications have a uniform cost, and the cost of a modification sequence is equal to its length.[4] Also, we assume a valid modification sequence cannot have an invalid prefix.

The result of applying a modification sequence to a model is defined as follows:

Definition 4 *Given a goal recognition model R, a design model D, and a modification sequence $\vec{m} = \langle m_1, \ldots, m_n \rangle$ s.t. $m_i \in \mathcal{M}$ and $\phi(\vec{m}, R) = 1$, $R^{\vec{m}}$ is the result of applying a modification sequence to R s.t.*

$$R^{\vec{m}} = \delta(m_n, \ldots, \delta(m_1, R))$$

4. As will be explained later, the model can be easily extended to support non-uniform design costs. This assumption is made for the sake of clarity, to prevent confusion between the costs assigned to agent actions in Definition 1 and the cost of modifications.

Similarly, the model R^m is the model that results from applying a single modification m to goal recognition model R. With a slight abuse of notation, we write $R^{\vec{m}} = \delta(\vec{m}, R)$ to represent the result of applying $\vec{m}$ to R.

The available ways to modify a goal recognition model and the different constraints that may be imposed on the design process vary between applications and settings. Our model supports this variety by accounting for arbitrary modifications that comply with the definitions above. In Section 1 we discussed two modification examples. Disallowing actions corresponds to removing actions from the action set, while sensor refinement, applied in partially observable settings, is expressed as a change in the recognition system's sensor model. A constraint function may, for example, impose a design budget, limiting the cost (or number) of allowed modifications. With the objective of maintaining usability, we can also bound the increase in optimal costs to the goals in the modified setting. Specifically, we may require the redesign process to leave the optimal cost for plans to all goals unchanged.[5]

Finally, a *GRD* model is defined as follows:

Definition 5 *A goal recognition design (*GRD*) model is given by the pair $T = \langle R_0, D \rangle$ where*

- *R_0 is an initial goal recognition model, and*
- *D is the* design model*, specifying the possible ways to redesign a goal recognition model.*

The design model D imposes a set $\mathcal{R}^T \subseteq \mathcal{R}$ of goal recognition models reachable from the initial model R_0 by applying a valid modification sequence.

3.2 Measure

In order to optimize goal recognition settings, we need to formulate the measure by which we assess how well goal recognition can be performed in a given model. In Section 1, we informally described the *worst case distinctiveness* (*WCD*) measure as the maximal cost of a path an agent can follow before its goal is revealed. To formally define this notion we first define the relationship between a path and a goal. We say that a path *satisfies* goal g if it is a prefix of a legal plan to g.

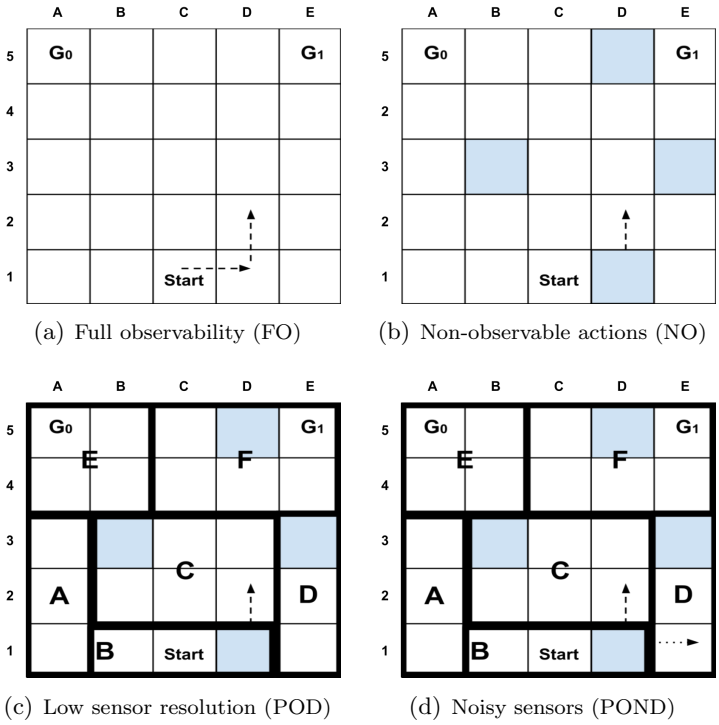
Definition 6 *A path $\vec{\pi}$* satisfies *a goal $g \in G$ if $\exists \pi \in \Pi^{leg}(g)$ s.t. $\vec{\pi}$ is a prefix of π.*

The relationship between a goal and the observation sequence emitted by an executed path is defined next.

Definition 7 *An observation sequence $\vec{o}$* satisfies *a goal $g \in G$ if $\exists \vec{\pi} \in \vec{\Pi}^{leg}(g)$ and $\vec{o} \in op(\vec{\pi})$.*

We denote by $G_R^A(\vec{\pi})$ the set of goals satisfied by path $\vec{\pi}$ in a goal recognition model R. The set of goals satisfied by observation sequence $\vec{o}$ is denoted by $G_R^O(\vec{o})$. Finally, for every $\vec{\pi}$, the set of goals satisfied by at least one of its observable projections is marked by $G_R^{\vec{O}}(\vec{\pi})$ s.t. $G_R^{\vec{O}}(\vec{\pi}) = \bigcup_{\vec{o} \in op(\vec{\pi})} (G_R^O(\vec{o}))$. When R is clear from the context we use $G^A(\vec{\pi})$, $G^O(\vec{o})$ and $G^{\vec{O}}(\vec{\pi})$, respectively.

5. By supporting arbitrary modifications and constraints we extend previous *GRD* models, which supported only specific constraints and modifications.



(a) Full observability (FO) (b) Non-observable actions (NO)

(c) Low sensor resolution (POD) (d) Noisy sensors (POND)

Figure 6: Different sensor models for a goal recognition setting.

Our analysis is based on the discovery of behaviors for which the observable projection may not reveal the goal of the executing agent (i.e., paths for which the observable projection satisfies more than one goal). We define *non-distinctive observation sequences* as those that satisfy more than one goal, and *non-distinctive paths* as those for which at least one observable projection is non-distinctive.

Definition 8 *An observation sequence* $\vec{o}$ *is* non-distinctive *if* $|G^O(\vec{o})| > 1$. *Otherwise, it is* distinctive.

Definition 9 *A path* $\vec{\pi}$ *is* non-distinctive *if* $|G^A(\vec{\pi})| \geq 1$ *(it is legal for some goal) and* $|G^{\vec{O}}(\vec{\pi})| > 1$ *(at least one of its observable projections is non-distinctive). Otherwise, it is* distinctive.

A non-distinctive path is a legal path that leads to some goal, and for which at least one of its observable projections $\vec{o} \in op(\vec{\pi})$ is non-distinctive and shared with a path to a different goal. Note that according to Definition 9, the empty zero-cost path $\vec{\pi}_\emptyset = \langle\rangle$, which is a legal path to all goals, is non-distinctive.

To illustrate the definitions given above, Figure 6 depicts the goal recognition setting described in Example 1. In this example we assume agents are optimal. The figure describes various sensor models and the observation sequences collected for a specific agent who enters

the system and performs two move actions (moving right and then upward). The executed path corresponds to the sequence $\vec{\pi} = \langle \text{move}(C1, D1), \text{move}(D1, D2) \rangle$.

Under the assumption that agents act optimally in a fully observable (FO) setting (Figure 6(a)), the emitted observation sequence corresponds to the performed sequence and $\vec{\pi}$ satisfies a single goal, which is G_1 on the right. In this setting, $op(\vec{\pi}) = \vec{o} = \vec{\pi}$ and $G^A(\vec{\pi}) = G^O(\vec{o}) = G^{\vec{O}}(\vec{\pi}) = \{G_1\}$. This is because, under the full observability assumption, the set $G^A(\vec{\pi})$ of goals satisfied by the path is equal to the set $G^O(\vec{o})$ satisfied by its observable projection $\vec{o}$. In addition, since the sensor model is deterministic, $\vec{o}$ is the only possible observation sequence and $G^O(\vec{o}) = G^{\vec{O}}(\vec{\pi})$. This means that $\vec{\pi}$ is distinctive and the recognition system can infer the agent's goal after $\vec{\pi}$ is executed. In fact, recognition can occur as early as the first step since there is no legal path to G_0 that starts by moving right.

Figure 6(b) depicts a similar setting with partial sensor coverage (NO). The lack of sensors near the initial state causes a delay in recognition of the second move action, where $\vec{\pi}$ emits the observation sequence $\vec{o} = \langle \text{move}(D1, D2) \rangle$. As in the fully observable setting, $G^A(\vec{\pi}) = G^O(\vec{o}) = G^{\vec{O}}(\vec{\pi}) = \{G_1\}$.

Finally, consider the two partially observable settings depicted in Figures 6(c) (POD) and 6(d) (POND), where the recognition system perceives the agent via a low-resolution or noisy sensor model, respectively. The non-observable actions from the NO setting are still non-observable here, and the observation tokens correspond to the zones A-F depicted in the figure. In Figure 6(c), sensor resolution is poor and all move actions in a zone emit the same token (each zone is depicted by a rectangle in the figure, and each move action emits the token corresponding to the zone which includes the movement's destination cell). In this setting, referred to as POD (for Partially Observable Deterministic sensor model), the execution of the path $\vec{\pi}$ emits the observation sequence $\vec{o} = \langle \text{move(ZoneC)} \rangle$ (the first action is non-observable). This observation sequence is non-distinctive since it may be emitted by legal paths to both goals. In fact, the agent may almost reach its goal without being recognized. Accordingly, while $G^A(\vec{\pi}) = \{G_1\}$, the set $G^O(\vec{o})$ includes both goals (i.e., $G^O(\vec{o}) = \{G_0, G_1\}$), since there is a legal path to G_0 that may emit $\vec{o}$. Since the sensor model is deterministic, any path may emit a single observation sequence and $G^O(\vec{o}) = G^{\vec{O}}(\vec{\pi})$.

A similar situation occurs in the noisy (non-deterministic) sensor model setting depicted in Figure 6(d), and referred to as POND (for Partially Observable Non-Deterministic sensor model). Here, after an agent performs move$(D1, D2)$, the perceived observation is either $\vec{o} = \langle \text{move(ZoneC)} \rangle$ or $\vec{o}' = \langle \text{move(ZoneD)} \rangle$, which correspond to the tokens the move action is mapped to by the non-deterministic sensor model. This is an example of a setting where the set $G^O(\vec{o})$ of goals satisfied by a possible observation sequence for a path differs from the set $G^{\vec{O}}(\vec{\pi})$ of goals satisfied by the path. In particular, $G^O(\vec{o}) = G^{\vec{O}}(\vec{\pi}) = \{G_0, G_1\}$ while $G^O(\vec{o}') = \{G_1\}$.

The examples above demonstrate the type of goal recognition models we wish to support. However, while the descriptions referred to a single observation sequence, our analysis needs to reflect an aggregated account of all possible agent behaviors and the possible ways the behavior may be perceived by the system. Accordingly, the measure by which we evaluate a goal recognition model is *worst case distinctiveness* (WCD), which represents the maximal cost of a path an agent can take in a system without its goal being revealed. We let $\vec{\Pi}^{nd}(R)$

represent the *non-distinctive path set* of a model R, which includes all the non-distinctive paths in R, and define *WCD* as follows.

Definition 10 *The* worst case distinctiveness *(*WCD*) of a model R, denoted by* $\mathrm{WCD}(R)$*, is:*

$$\mathrm{WCD}(R) = \max_{\vec{\pi} \in \vec{\Pi}^{nd}(R)} \mathcal{C}_A(\vec{\pi})$$

The cost of a path in this setting signifies the degree to which an agent advances in the environment. When action costs are uniform, we can replace $\mathcal{C}_A(\vec{\pi})$ with $|\vec{\pi}|$ to measure the maximal number of actions (steps) on a non-distinctive path. The empty path $\vec{\pi}_\emptyset$ is considered non-distinctive (Definition 9) and therefore the minimal *WCD* of a goal recognition model is 0.

Returning to the example depicted in Figure 6, in the fully observable setting (Figure 6(a)) the *WCD* is 4 since an (optimal) agent aiming for either goal can move up 4 steps before turning left or right. In the partially observable settings (figures 6(b)–6(d)) *WCD* is 5, since an agent aiming for G_1 can move up 4 steps and then one step to the right (to a cell without a sensor) without revealing its goal. Note that if agents are suboptimal and have a budget of at least 4 steps for diverging from optimal behavior, *WCD* is increased by 2 in each model.

3.3 Redesign

After formulating the *WCD* measure (i.e., the measure by which we assess a goal recognition model), we turn to the second objective of *GRD*: redesigning the model so as to minimize *WCD*. For this purpose, we must define a *design model* specifying possible ways to modify the goal recognition model, along with any constraints that the design process must respect.

Given a *GRD* model $T = \langle R_0, D \rangle$, our objective is to find a modification sequence $\vec{m}^* \in \vec{\mathcal{M}}$ to apply to R_0 which will minimize *WCD* under the constraints specified by ϕ. We let $WCD^{min}(T)$ represent the minimal *WCD* achievable in T when applying an optimal modification sequence, and we let $R_0^{\vec{m}}$ represent the model that results from applying $\vec{m}$ to R_0. We formulate our objective as follows:

$$WCD^{min}(T) = \underset{\vec{m} \in \vec{\mathcal{M}} | \phi(\vec{m}, R_0) = 1}{\text{minimize}} \quad WCD(R_0^{\vec{m}}) \tag{1}$$

In particular, given the minimal *WCD* possible, we prefer solutions with minimal length.

In the fully observable setting described in Example 1, physical barriers can be placed to direct the flow of passengers to distinctive paths by disallowing specific move actions. The constraints, specified by ϕ, may require the optimal cost to any goal to remain unchanged in the modified model. In addition, a budget of available modifications may be defined. The solution reveals that under the assumption agents are optimal, by placing a single barrier *WCD* is reduced from 4 in the original setting to 0 in the redesigned model (Figure 2(c)). However, even with this barrier, in the partially observable setting depicted in Example 2, the agent can advance one step without her goal being recognized ($WCD = 1$). Here, sensors are used to improve agents' visibility to the system, and *WCD* is minimized to 0 by positioning a single camera at the entry point (Figure 4(b)).

4. Calculating *WCD*

The calculation of *WCD* differs from both planning problems, where the aim is to find any legal path to a goal, and goal recognition problems, where the aim is to find any legal path that fits a perceived observation sequence. In contrast, the analysis of *GRD* must take into account all possible observation sequences corresponding to all possible paths to all goals. After describing a general approach for *WCD* calculation, we show that *WCD* can be computed using a compilation from *GRD* to classical planning for settings where the set of legal plans to a goal are cost-bounded.

Given a *GRD* model, the naïve approach to *WCD* calculation consists of exhaustively examining each possible legal path (legal plan prefix) and checking whether it is non-distinctive (i.e., whether it is mapped to more than one goal by the recognition system). The value of *WCD* is then the maximal non-distinctive path. However, in most cases, the set of possible plans is large and the set of legal plans is not given explicitly, but instead compactly represented via the relationship between actions, states and goals. Hence, exploring the entire plan set exhaustively becomes impractical for large problems.

One common way to compactly specify agent behavior is by using a planning domain (formally defined in Section 2.1) to describe the environment dynamics and to set a bound on the cost of plans agents may execute to achieve a goal. Accordingly, in this section we present methods for *WCD* calculation in goal recognition models where divergence from optimal behavior is assumed to be cost-bounded. In such models, which we refer to as *cost-bounded goal recognition* (*CB-GR*), for each goal $g_i \in G$ agents have a bound θ_i such that the set $\pi \in \Pi(g_i)$ of legal plans to g_i includes all plans π to g_i with cost $\mathcal{C}_A(\pi) \leq \mathcal{C}^*_A(g_i) + \theta_i$, where $\mathcal{C}^*_A(g_i)$ represents the optimal cost to goal g_i. The set of legal plans to each goal is represented by $\vec{\Pi}^{\theta}(g)$. As a special case, when agents heading for goal g_i are assumed to behave optimally, $\theta_i = 0$, and the set $\vec{\Pi}^{leg}(g_i)$ of legal plans to goal g_i is the set $\Pi^*(g_i)$ of optimal plans that minimize the cost to the goal.

In the following sections we present methods to calculate *WCD* of a *CB-GR* model, based on the following observations:

Theorem 1 *If $\vec{\pi}$ is non-distinctive, any prefix of $\vec{\pi}$ is non-distinctive.*

Proof: Let $\vec{\pi}_{pre}$ be a prefix of $\vec{\pi}$. According to Definition 9, $\vec{\pi}_{pre}$ is non-distinctive if $|G^A(\vec{\pi}_{pre})| \geq 1$ (the set $G^A(\vec{\pi}_{pre})$ of goals satisfied by the actual path includes at least one goal) and $|G^{\vec{O}}(\vec{\pi}_{pre})| > 1$ (the set $G^{\vec{O}}(\vec{\pi}_{pre})$ of goals satisfied by its observable projections includes at least two goals).

The first condition is satisfied by the fact that $\vec{\pi}$ is a legal path to some goal g and $\vec{\pi}_{pre}$ is a prefix of $\vec{\pi}$, thus satisfying g (Definition 6). The second condition is satisfied by the fact that since $\vec{\pi}$ is non-distinctive, $\exists \vec{o} \in op(\vec{\pi})$ s.t. $|G^O(\vec{o})| > 1$ (at least one of $\vec{\pi}$'s observable projections is non-distinctive). This means that $\exists g' \in G$ and $\vec{\pi}'$ s.t. $g \neq g'$, $\vec{\pi}' \in \vec{\Pi}^{leg}(g')$ and $\vec{o} \in op(\vec{\pi}')$ (Definition 7). Definition 2 guarantees that $\exists \vec{o}_{pre}$ that is a prefix of $\vec{o}$, s.t. $\vec{o}_{pre} \in op(\vec{\pi}_{pre})$, and that there is a prefix $\vec{\pi}'_{pre}$ of $\vec{\pi}'$ s.t. $\vec{o}_{pre} \in op(\vec{\pi}'_{pre})$, ensuring $\vec{\pi}_{pre}$ is non-distinctive. ∎

Corollary 1 *If $\vec{\pi}$ is distinctive, any path $\vec{\pi}' \in \vec{\Pi}^{leg}(G)$ for which $\vec{\pi}$ is a prefix, is distinctive.*

Proof: Assume to the contrary that $\vec{\pi}$ is distinctive and is a prefix of a non-distinctive path $\vec{\pi}'$. However, according to Theorem 1, if $\vec{\pi}'$ is non-distinctive, any prefix of $\vec{\pi}'$, including $\vec{\pi}$, is non-distinctive, which serves as a contradiction. ■

The above observations assure us that any agent in a *CB-GR* setting will start its progress in the system by following a (possibly empty) non-distinctive path and end with a (possibly empty) distinctive path leading to its goal. This allows us to establish a relationship between a *CB-GR* model, which offers an online analysis of a single observation sequence $\vec{o}$ emitted by the path performed by an agent (e.g., Ramirez & Geffner, 2009), and its corresponding *GRD* model, which provides an offline account of the set $\vec{O}$ of all possible observation sequences that may be generated by agents in the system. To formulate this relationship we start by defining the *action set* of a token. Given a sensor model S and a token $o \in O$, the action set of o, marked by $A_S[o]$, is the set of actions that may emit o when executed.

Definition 11 $A_S[o] = \{a' | o \in S(a')\}$

Similarly, we define the *non-distinctive action set* of action $a \in A$ as the set of actions (excluding a) that share a common observation token with a. We notate this as $A_S[a]$.

Definition 12 $A_S[a] = \{a' | a' \neq a \ , \ S(a) \cap S(a') \neq \emptyset\}$

We let ${C_A}^{min}(\vec{o})$ represent the minimal cost of a sequence of actions that may emit $\vec{o}$ and define it as follows:

Definition 13 *Let $\vec{o} = \langle o_1, ..., o_n \rangle$ be an observation sequence. Then,*

$$ {C_A}^{min}(\vec{o}) = \Sigma_{i=1}^{n}(\min_{a \in A_S[o_i]} C_A(a)) $$

Note that ${C_A}^{min}(\vec{o})$ does not necessarily represent the cost of a legal path, since a path may include non-observable actions that are not accounted for by the sequence. We use ${C_A}^{min}(\vec{o})$ as a lower bound on the cost of the possible paths that may emit $\vec{o}$ to state the following:

Theorem 2 *Given a goal recognition model R and an observation sequence $\vec{o}$, if ${C_A}^{min}(\vec{o}) >$ WCD(R) then $|G^O(\vec{o})| = 1$.*

Proof: We assume $\vec{o}$ is emitted by an agent following a legal path $\vec{\pi} \in \Pi^{leg}(G)$. Definition 2 assures us that while not all executed actions in $\vec{\pi}$ emit a token, each token $o \in \vec{o}$ corresponds to an executed action. Definition 13 assures us that among the paths $\vec{\pi}$ for which $\vec{o} \in op(\vec{\pi})$ there is no path with a lower cost than ${C_A}^{min}(\vec{o})$. According to Definition 10, $WCD(R)$ is the maximal cost of a non-distinctive path. Therefore, if ${C_A}^{min}(\vec{o}) > WCD(R)$, then the underlying path that generated $\vec{o}$ is distinctive. Definition 9 classifies a path as distinctive if $\max_{\vec{o}' \in op(\vec{\pi})} |G^O(\vec{o}')| \leq 1$. In particular, this applies to the actual generated sequence $\vec{o} \in op(\vec{\pi})$, and therefore $|G^O(\vec{o})| = 1$. ■

Theorem 2 provides a guarantee that any observation sequence $\vec{o}$ for which ${C_A}^{min}(\vec{o})$ exceeds $WCD(R)$ represents distinctive behavior. For each observation, the value of ${C_A}^{min}(\vec{o})$

can be directly computed from the sensor model S and cost function C_A by considering for each token the cost of the least expensive action in its action set $A_S[o]$. For the sake of computational efficiency, the recognition system may exploit this observation to fully analyze only observations that are guaranteed to represent distinctive paths. Therefore, the process of minimizing the *WCD* of a model, described in Section 5, provides a way to guarantee improved online recognition by minimizing the maximal non-distinctive paths agents can follow before recognition is guaranteed.

Note that the above theorems are not guaranteed for all goal recognition models. In particular, they do not apply to goal recognition settings based on the probabilistic measure suggested by Ramirez and Geffner (2010). For each observation sequence $\vec{o}$, Ramirez and Geffner (2010) suggest computing for each goal g the minimal cost difference between a plan to g that satisfies $\vec{o}$ and a plan to g that does not. The observation sequence is then mapped to the goal that minimizes this difference. In such goal recognition settings, a non-distinctive path may have a distinctive prefix and vice versa. Accordingly, the methods for *WCD* calculation we suggest next, which are based on the theorems presented above, are not guaranteed to find the *WCD* for such settings.

4.1 BFS-Based Method for *WCD* Calculation

A naïve way to discover all paths to all goals within a *CB-GR* model is to perform an iterative and exhaustive exploration of the state space using, for example, a breadth-first search (BFS) over the action space. The search starts at the initial state and explores at each level all states that can be reached from the previous level. The search continues up to the level at which the most expensive legal plans are found. The result is a tree depicting all legal plans to all goals in G. In order to reveal the *WCD* value of the model, we need to find the set of goals that share the most expensive non-distinctive path. We can do this by maintaining a priority queue and performing a backward search starting at the most expensive leaf, and advancing one level at time. We stop once a node that represents a non-distinctive path is discovered.

The remaining question is how to identify distinctive paths. According to Definition 9, a path $\vec{\pi}$ is distinctive if $|G^A(\vec{\pi})| = 0$ (it is illegal) or if $|G^{\vec{O}}(\vec{\pi})| = 1$ (all of its observable projections are distinctive). Both criteria can be verified using one of the many goal recognition techniques available, depending on the specific model at hand. The recognition technique is used to identify the number of goals a path (and its observable projections) satisfies.

Although this method is sound, it is highly inefficient, especially in scenarios where there are many legal paths to each goal. The *wcd-bfs*, whose pseudocode is given in Algorithm 1, is a variation of the BFS presented above — only instead of blindly exploring all paths in the model we trim the search by pruning nodes representing distinctive paths. For *CB-GR* models, Corollary 1 assures us that further exploring such nodes is futile, since any path that has a distinctive prefix is also distinctive. Successors of non-distinctive paths are generated by appending every applicable action to the current path and adding the resulting path to a queue. It is worth noting that nodes represent paths rather than states to support the task of accounting for *all* legal paths that may lead to a certain state. The

search continues as long as the queue is not empty. The *WCD* value of the model is the maximal-cost path among the expanded paths.

Algorithm 1 *wcd-bfs*

```
1: create a (priority) queue Q = ∅
2: WCD= 0
3: enqueue ⟨⟩(empty path) onto Q
4: while Q is not empty do
5:    π⃗ ← Q.dequeue()
6:    if π⃗ is non-distinctive then
7:       enqueue successors(π⃗) onto Q
8:       if C_A(π⃗) > WCD then
9:          WCD= C_A(π⃗)
10:      end if
11:   end if
12: end while
13: return WCD
```

Comparing this method to the baseline method presented above, it is clear that pruning can prevent many unnecessary states from being expanded, especially in a domain with a large branching factor. However, as shown by Keren et al. (2014), using goal recognition several times for each node is expensive and impractical for large domains. In the next section we will present compilation-based techniques for *WCD* calculation.

4.2 Calculating *WCD* via Compilations to Planning

BFS-based approaches for finding the *WCD* value of a model exhaustively explore the state space and require solving a separate goal recognition problem for each examined path to determine if it is distinctive. In this section we find the *WCD* value of a *CB-GR* model by using a compilation to classical planning that can be solved using any off-the-shelf planner.

A key difference between goal recognition (design) and planning is that while a goal recognition problem includes a set of at least two possible goals, a classical planning problem has a single goal. The basic idea underlying our compilation is to transform a goal recognition problem with multiple goals into a single planning problem with multiple agents, each with a different goal. The single objective of the compiled problem is then for all agents to reach their respective goals. By manipulating the cost function, we incentivize agents to prefer non-distinctive paths and reveal the *WCD* value of the model (see Figure 7 for an illustration).

This idea is implemented in the *common-declare* compilation (described below) to find the *WCD* of a *CB-GR* model with two goals. After presenting the compilation, we show how *WCD* of a model with multiple goals is found by taking the maximal value over all ordered goal pairs. In addition, we describe variations of the compilation, tailored to more restricted forms of partial observability for which the computation can be performed more efficiently.

To define the measure found by our methods, we let $\vec{\Pi}_i^{nd}(R) \subseteq \vec{\Pi}^{nd}(R)$ represent the set of non-distinctive paths that are legal to goal g_i but share at least one observation sequence with a path to some other goal g_j. Accordingly, $WCD_i(R)$ represents the maximal cost of a path in $\vec{\Pi}_i^{nd}(R)$.

Definition 14

$$\vec{\Pi}_i^{nd}(R) = \{\vec{\pi} | g_i \in G^A(\vec{\pi}) \text{ and } \exists g_j \text{ s.t. } g_i, g_j \in G^{\vec{O}}(\vec{\pi})\}$$

$$\text{WCD}_i(R) = \max_{\vec{\pi} \in \vec{\Pi}_i^{nd}(R)} \mathcal{C}_A(\vec{\pi})$$

Recall from Section 3.2 that the zero-cost empty path $\vec{\pi}_\emptyset$ is shared by any two goals. Accordingly, the set $\vec{\Pi}_i^{nd}(R)$ includes $\vec{\pi}_\emptyset$ and $0 \leq WCD_i(R)$ for any goal g_i.

Our model supports settings in which an agent may reach its goal without being observed. Assuming agents are bounded suboptimal, we let $C_A^{max}(g_i)$ represent the maximal cost to goal g_i, i.e., the maximal cost of a plan $\pi \in \vec{\Pi}^\theta(g_i)$. The value of $WCD_i(R)$ is bound by $C_A^{max}(g_i)$.

Lemma 1 *Given a goal recognition model R, for any goal $g_i \in G$*

$$\text{WCD}_i(R) \leq C_A^{max}(g_i)$$

Proof: According to Definition 14, $WCD_i(R)$ represents the cost of a legal path to g_i bounded by the cost of a complete legal plan, thus creating a bound on $WCDi(R) \leq C_A^{max}(g_i)$. ∎

We let $R_{0,1}$ represent a goal recognition model with two goals $G = \{g_0, g_1\}$. The *common-declare* compilation, whose formal definition is given in Appendix A, is used to calculate $WCD_0(R_{0,1})$, which is the maximal non-distinctive path that is legal for an agent heading to g_0 in $R_{0,1}$. Note that for any two goals, $WCD_0(R_{0,1})$ is not necessarily equal to $WCD_1(R_{0,1})$ (and $\vec{\Pi}_0^{nd}(R_{0,1})$ is not necessarily equal to $\vec{\Pi}_1^{nd}(R_{0,1})$). This is due to the environment's partial sensor coverage and is exemplified in Figure 6(d), where $WCD_0(R_{0,1}) = 4$, since an agent can only reach cell $C5$ while following optimal paths to both goals, but $WCD_1(R_{0,1}) = 5$, since an optimal agent heading to G_1 can remain unrecognized even after turning right due to the missing sensor at cell $D5$.

The *common-declare* compilation involves two agents, $agent_0$ and $agent_1$, each with a copy f_i of each fact $f \in F$. The agents start at the same initial state but are aiming for different goals (g_0 and g_1, respectively). Each agent has a separate copy A_i of the action set A, where each action emits an observation token. Token emission is represented by setting (assigning the value *true* to) the predicate $performed_i^k$. To support non-deterministic sensor models, each agent has a separate copy a_i^k of the original action a for each token k the action may emit (including the empty token $o_\emptyset$). This allows the planner to choose among the different observation tokens by executing the corresponding action.

Each agent needs to account for the observation sequence of its path by "declaring" emitted tokens. Each action deletes the predicate $declared_i$, which needs to be reset (by performing one of the "declare" actions) before the agent can perform the next action.

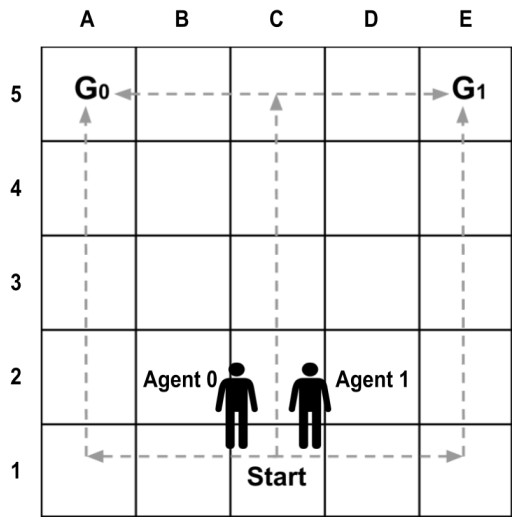


Figure 7: The *common-declare* compilation

While the execution cost for $agent_1$ is associated with the executed actions, $agent_0$ "pays" for its actions when declaring the action's token. This is done using one of the Pay_0 actions: the Pay_0^c action pays the full cost of the action, while the corresponding $PayDiscounted_0^c$ pays a slightly discounted cost.

Agents can declare their observation tokens either separately (by performing $Declare_i^k$ or $Declare_0^\emptyset$), or jointly (by performing $Declare_{0,1}^k$) if both emit the same observation token. Token declaration is "free" for both agents, but joint declarations $Declare_{0,1}^k$ and empty token declarations $Declare_0^\emptyset$, applicable only so long as the *exposed* flag is false, allow $agent_0$ to execute $PayDiscounted_0^c$ and pay a discounted fee for its last performed action. The flag is initially false and is set to true once and for all by the first separate declaration.

At the initial state, both agents have no pending tokens and can start execution. The goal description requires both agents to reach their goal with no pending tokens. The solution to the problem is a plan where both agents interleave execution and declaration (and payment for $agent_0$) until they achieve their goals. We note that although the compilation includes two agents, the proposed model serves a single agent with the objective of finding a plan to follow that maximizes the non-distinctive prefix of $agent_0$. The two-agent model is merely used to account for different agent behaviors within a single search.

To account for bounded suboptimal agents, we force each agent to follow a plan with length $T_i = \mathcal{C}_A^*(g_i) + \theta_i$, which represents the maximal cost of a legal plan. To constrain the plan lengths, we maintain a timer for each agent. Each action a advances the time step from t to $t + C_A(a)$ for the acting agent. The goal specification requires that each agent reach time step $time_T^i$ in addition to achieving its original goal. To support legal paths with a cost smaller than T_i, we add to the model idle actions, $Idle_i^c$, which cost c and advance the counter by c time stamps. Idle actions set the *exposed* flag to true, ensuring that they are not considered part of the non-distinctive prefix (Appendix A provides implementation details for this mechanism).

Accounting for suboptimal agents results in a numeric planning problem, which can be solved by any off-the-shelf numeric planner, e.g., Aldinger, Mattmüller, and Göbelbecker (2015). However, due to the computational limitations of current state-of-the-art numeric planners, we transform the compiled problem into a classical planning problem (as first presented by Keren et al., 2015). We let $dom(C_A)$ represent the set of possible action costs in the model. We then add a sequence of time steps $\langle time_i^0, ..., time_i^{T_i} \rangle$ and a new predicate $next(time_i^t, time_i^{t+c})$ for every time stamp $0 \leq t \leq T_i$ and $c \in dom(C_A)$. The predicates enforce action execution durations and are added to the preconditions of every action in a_1^k that has $time_i^t$ as its starting point and $time_i^{t+c}$ as its end point. In addition, a separate $Idle_i^c$ action is created for each $c \in dom(C_A)$. Clearly, such a solution does not scale as $|dom(C_A)|$ increases, but it allows the use of any off-the-shelf planner to find the *WCD* value in *GRD* settings where $|dom(C_A)|$ is relatively small.

To demonstrate the compilation, consider the partially observable (POND) example depicted in Figure 6(d), where agents have no budget for diverging from optimal behavior (e.g., $\theta_i = 0 \; \forall g_i \in G$). If we apply the *common-declare* compilation to find $WCD_0(R_{0,1})$ and solve it using an optimal solver, the path found by the solver will consist of four movements upward for both agents, each followed by a shared declare (e.g., $move_0^{ZoneC}(C1, C2)$ and $move_1^{ZoneC}(C1, C2)$ followed by a $Declare_{0,1}^{ZoneC,1}$). These joint movements allow $agent_0$ to enjoy the discounted payment for its tokens and apply $PayDiscounted_0^1$. After reaching the wall (cell $C5$ in Zone F), the agents can no longer declare their tokens together without diverging from optimal behavior. At this point the goal of $agent_0$ is revealed after it performs $move_0^{ZoneE}(C5, B5)$, which emits the token $ZoneE$. This forces the agent to declare its token alone, setting the flag *exposed* to true and *WCD* to 4. Henceforth, declare actions entail the full cost.

When measuring $WCD_1(R_{0,1})$, a higher *WCD* of 5 is revealed. This is because $agent_1$, which is now the agent that must pay for emitted tokens, will choose to move to the right, $move_1^{\emptyset}(C5, D5)$, upon reaching the wall. Since no token is emitted the agent can apply $Declare_1^{\emptyset}$ without exposing its action. Only the last action, $move_1^{ZoneF}(D5, E5)$, exposes the agent by forcing it to perform a separate declaration of its final move action, $move_1^{ZoneF}(E5, F5)$, and applying $Declare_1^{ZoneF}$.

It is worth noting that a diversion budget of at least 4 is needed to increase $WCD_1(R_{0,1})$ to 6. This is because in such a setting $agent_0$ is forced to perform 10 actions. The optimal plan for the compiled problem would then be for both agents to advance together towards G_1, allowing $agent_1$ to pay a discounted rate for declare (and remain unrecognized) until reaching its goal.

To prove the correctness of the compilation, we let $\pi_{P'}$ represent a plan found by a planner for the compiled problem P', and we let $\pi^*_{P'}$ represent a solution found by an optimal planner.[6] The plan, which consists of actions for both agents, is divided into two parts. The first part consists of actions declared by both agents together (using $Declare_{0,1}^k$), along with non-observable actions performed by $agent_0$, declared using $Declare_0^{\emptyset}$. In the second part, each agent declares its tokens separately. The two parts of the plan are divided by the first time the flag *exposed* is set, thenceforth allowing tokens to be declared

6. Recall that an optimal solution for the compiled problem may represent suboptimal agent behavior.

only separately. The value of $WCD_0(R_{0,1})$ is the accumulated cost of "real" actions A_0 (including non-observable actions) performed by $agent_0$ before exposure.

Given a solution $\pi_{P'}$ to P', we denote the *projection* of $\pi_{P'}$ on each agent i by $proj(\pi_{P'}, g_i)$, which includes all actions in A_i that appear in $\pi_{P'}$ (excluding the declare and payment actions). Accordingly, the projection of the optimal solution $\pi^*_{P'}$ to P' for each agent is denoted as $proj(\pi^*_{P'}, g_i)$. In addition, we let $unexposed(\pi^*_{P'}, g_0)$ be the unexposed prefix of $proj(\pi_{P'}, g_0)$, i.e., the actions performed by $agent_0$ up to (but not including) the first action declared separately by $agent_0$.

The proof of correctness starts by showing that $unexposed(\pi_{P'}, g_0)$ (of any plan $\pi_{P'}$ found by the compilation) represents a non-distinctive path of $agent_0$. We then provide the conditions under which $\pi^*_{P'}$ is guaranteed to yield legal plans for both agents, and show that $unexposed(\pi^*_{P'}, g_0)$ represents $WCD_0(R_{0,1})$. We conclude by proving that the *WCD* value of the entire model can be found by taking the maximal value over individual *WCD* values of all pairs.

To show that $unexposed(\pi_{P'}, g_0)$ is non-distinctive we rely on the compilation definition, which guarantees that those actions performed before the first separately-declared action either emit a token that is shared by an action on a path to a different goal, or are non-observable.

Lemma 2 *$unexposed(\pi_{P'}, g_0)$ is non-distinctive.*

Proof: The compilation forces the agent to account for the observable projection of its path by declaring each token emitted by its performed actions. A $declared_i$ flag, initially set to false, indicates whether that agent has a pending token. An action is only applicable after $declared_i$ is set to true. The action sets the flag to false, forcing the agent to declare its pending token before the next action is performed or the goal can be achieved.

The compilation guarantees that any action in $unexposed(\pi_{P'}, g_0)$ represents either a non-observable action, which emits the empty token $o_\emptyset$ and is followed by a $Declare^{\emptyset}_0$, or an observable action which emits a token k and is followed by a $Declare^{k}_{0,1}$ action where both agents declare their pending token together. Both types of declare actions can be performed only before the flag *exposed*, initially set to false, is set to true. This happens when the first separate declare is performed by $agent_0$, and cannot thereafter be changed. This means that actions which appear in $unexposed(\pi_{P'}, g_0)$ form an observation sequence that produces an observable projection shared by both goals, and is therefore non-distinctive. ∎

We guarantee that $proj(\pi^*_{P'}, g_i)$ yields a legal plan for both agents by bounding the discount $agent_0$ may receive for token declaration. As detailed in Definition 20 (Appendix A), ϵ represents the discount for payment actions s.t. for every action a that costs c, the full payment for an action using Pay^c_0 is c, while the discounted rate is $c(1-\epsilon)$. To guarantee that agents follow legal paths in the compiled problem, we set ϵ such that the maximal accumulated discount over an entire execution is lower than the smallest possible diversion by any agent from a legal path.

Given a *CB-GR* model R, we let $\delta^{min}(R)$ represent the minimal cost difference between any two paths in R.[7] We show that when ϵ is smaller than $\frac{\delta^{min}(R_{0,1})}{\mathcal{C}^*_A(g_0)}$, both agents follow legal paths in $\pi^*_{P'}$, i.e., $proj(\pi^*_{P'}, g_i)$ represents bounded suboptimal behavior in the original model for both agents. Recall that $\mathcal{C}^{max}_A(g_i)$ represents the maximal cost of a legal path to goal g_i in the original goal recognition model.

Theorem 3 *Given a* CB-GR *model* $R_{0,1}$ *with two goals* $G = \{g_0, g_1\}$ *and a transformed model* P', $proj(\pi^*_{P'}, g_i) \in \Pi^{leg}(g_i)$ *(both agents follow legal plans) if*

$$\epsilon < \frac{\delta^{min}(R_{0,1})}{\mathcal{C}^{max}_A(g_0)}$$

Proof: To ensure both agents choose a path that is legal in the original model $R_{0,1}$, we require that the difference between the cost of achieving g' in P' (denoted by $\mathcal{C}'_A(\pi^*_{P'}(g'))$) and the joint cost of achieving g_0 and g_1 in $R_{0,1}$ ($\mathcal{C}^{max}_A(g_0)$ and $\mathcal{C}^{max}_A(g_1)$, respectively) be smaller than the minimal cost of diversion from a legal path in $R_{0,1}$. Under the assumptions that an action is associated with a non-negative cost and that the cost of diversion from a legal path is at least $\delta^{min}(R_{0,1})$, we require the difference between the optimal cost of the solution of the compiled problem and the sum of both agents' maximal plan costs to be at most $\delta^{min}(R_{0,1})$:

$$\mathcal{C}'_A(\pi^*_{P'}(g')) - [\mathcal{C}^{max}_A(g_0) + \mathcal{C}^{max}_A(g_1)] < \delta^{min}(R_{0,1})$$

According to C'_A, the difference between the costs of achieving the goals in the original setting and in the compiled problem P' is due only to the discount $agent_0$ receives for actions in $unexposed(\pi^*_{P'}, g_0)$. Lemma 2 showed that $unexposed(\pi^*_{P'}, g_0)$ represents non-distinctive behavior. The cost of $unexposed(\pi^*_{P'}, g_0)$ is bound by $WCD_0(R_{0,1})$ (Definition 14), the maximal cost of a non-distinctive legal path to g_0. We therefore need to ensure that

$$\epsilon \cdot WCD_0(R_{0,1}) < \delta^{min}(R_{0,1})$$

and thus when

$$\epsilon < \frac{\delta^{min}(R_{0,1})}{WCD_0(R_{0,1})}$$

agents will follow legal plans.

According to Lemma 1, $WCD_0(R_{0,1}) \leq C^{max}_A(g_0)$. Therefore, when

$$\epsilon < \frac{\delta^{min}(R_{0,1})}{C^{max}_A(g_0)}$$

the agents will follow legal plans.

■

Theorem 4, next, shows that the optimal solution to P' yields $WCD_0(R_{0,1})$.

7. We assume all costs are non-negative rational numbers, and therefore it is possible to scale all costs by some factor such that all costs are integers and the minimal cost difference is 1.

Theorem 4 *Given a goal recognition model $R_{0,1}$ with two goals $G = \{g_0, g_1\}$ and a planning problem P', created according to the* common-declare *compilation with $\epsilon < \frac{\delta^{min}(R)}{C_A^{max}(g_0)}$, then*

$$\text{WCD}_0(R_{0,1}) = \mathcal{C}_A(unexposed(\pi^*_{P'}, g_0)))$$

Proof: The bound on ϵ (Theorem 3) guarantees that apart from the declare and payment actions, the solution to P' consists solely of actions that form a pair of legal plans to each of the goals. Lemma 2 shows that $unexposed(\pi^*_{P'}, g_0)$ represents a non-distinctive path. The only way to minimize the cost of the solution to P' is by maximizing the cost of any actions $agent_0$ performs before the first separate declaration. This guarantees that $\pi^*_{P'}$ is the solution to P' which maximizes $\mathcal{C}_A(unexposed(\pi^*_{P'}, g_0))$ and therefore represents $WCD_0(R_{0,1})$. ∎

Note that throughout the execution no discount is assigned to $agent_1$'s actions, guaranteeing maximization of the non-distinctive prefix of $agent_0$ rather than choosing a path that maximizes the cost of non-observable actions performed by both agents.

As a final stage of validating our approach, we observe that $WCD(R)$ represents the maximal non-distinctive path shared between at least a pair of goals. Therefore, by finding WCD for all ordered pairs, we are guaranteed to find the maximal non-distinctive path of a model.

Lemma 3 *Given a goal recognition problem R,*

$$\text{WCD}(R) = \max_{g_i, g_j \in G | g_i \neq g_j} (\text{WCD}_i(R_{i,j}))$$

Proof: According to Definition 9, any path $\vec{\pi}_{wcd}$ that is a non-distinctive path with maximal cost in R has at least a pair of goals $g, g' \in G$ for which $g_i, g_j \in G^{\vec{O}}(\vec{\pi}_{wcd})$ and $g_i \in G^A(\vec{\pi}_{wcd})$. Therefore, $\vec{\pi}_{wcd} \in \vec{\Pi}_i^{nd}(R)$. Since $\vec{\pi}_{wcd}$ is maximal among non-distinctive paths and according to Definition 14, $WCD_i(R_{i,j}) = WCD(R)$ and there is no other pair that shares a more costly non-distinctive path. ∎

We use Lemma 3 to justify our all-pairs approach. We find the WCD value of a GRD problem with $n > 2$ by finding the WCD value shared between all goal pairs, and we assign WCD to be the maximal WCD of all pairs. This method involves solving n^2 planning problems, each with a branching factor of $2|A||O|$. An alternative approach would be to incorporate all goals into a single planning problem with n agents. This, however, will lead to a planning problem with 2^n versions of each action, since one needs to account for all possible agent combinations.

As a final note, while the *common-declare* compilation (formally described in Definition 20) could admit many permutations of the same plan, we can optimize the process by disallowing some permutations with additional constraints. In particular, after *exposed* becomes true, we can force $agent_1$ to wait until $agent_0$ achieves its goal. This can be implemented by adding a no-cost action that sets a flag $done_0$. The flag allows $agent_1$ to declare its token separately but disallows separate declare actions for $agent_0$. This in fact disables actions performed by $agent_0$, forcing it to achieve its goal before setting $done_0$ and allowing $agent_1$ to achieve its goal.

4.2.1 Variations of *common-declare*

Following the general idea used for the *common-declare* compilation, we now describe two variations of the compilation that apply to two special cases of observability. Although the general compilation applies to both settings, the specially tailored variations of the compilation are preferable as they are more compact and more computationally efficient.

Fully observable agents A special case of partial observability is a setting with perfect sensing where the sensor function deterministically maps each action to a unique token (i.e., the action's name). Using the *common-declare* compilation in this setting is possible, but would add unnecessary complexity to the solution. An alternative approach is to exploit the correspondence between the executed action and the emitted token to incorporate token declarations into the action's description.

Similarly to the *common-declare*, the *latest-split* compilation, described in detail in Keren et al. (2014), consists of two agents, each aiming at its respective goal. The agents have two types of actions: *separate* actions, performed by one or the other agent alone, and *together* actions, where both agents advance together. We guarantee the plan prefix that agents follow together represents the maximal non-distinctive path by offering a small discount ϵ for joint actions allowed only before a flag *split*, initially set to false, becomes true. After splitting, represented by a one-time setting of *split* to true, only separate actions are allowed. The value of $WCD_0(R_{0,1})$ (which is equal to $WCD_1(R_{0,1})$ in the fully observable setting) is the cost of the sequence of actions agents perform together before splitting.

Acting together in the *latest-split* compilation is similar to performing an action and then declaring its token together in the *common-declare* compilation. The benefit of the *latest-split* compilation is that it avoids the need to specify the sensor model in the problem description. As shown in our empirical evaluation, the compact representation results in a more efficient solution.

Non-observable actions Another special case of partial observability is when the action set $A = A^o \cup A^{no}$ is partitioned into observable (A^o) and non-observable (A^{no}) actions. In this setting, first presented in Keren et al. (2016a), when an agent performs an observable action it is correctly perceived by the observer, while the execution of a non-observable action goes unnoticed.

Similarly to the *common-declare* compilation, the *latest-expose* compilation, fully described in Keren et al. (2016a), consists of two agents, each aiming at its respective goal. The solution is divided into two parts by a common *exposure point*. The prefix of the plan up to the exposure point represents a non-distinctive path, and may consist of actions performed by both agents simultaneously in addition to non-observable actions performed by $agent_0$. After the exposure point, each agent acts separately to achieve its goal. To discover $WCD_0(R_{0,1})$ we discount the actions $agent_0$ performs before exposure.

The *latest-expose* compilation differs from *common-declare* by integrating the execution of actions and the declaration of their tokens, which can either be the action itself or the empty token for non-observable actions. This reduces the size of the action space by not including token declaration actions. In our empirical evaluation section we show the benefits of this compact representation.

5. Redesign - Minimizing *WCD*

The design process formally described in Section 3.3 takes as input a goal recognition setting and uses the available modifications to minimize the *WCD* value of the model. Typically, the number of possible modifications is large, making it impossible to exhaustively explore all possibilities. We thus need ways to efficiently search for an optimal redesign sequence. In this section, we start by describing a general approach to design and then show how pruning can be applied to reduce the computational effort of the design process. We then characterize a class of *GRD* models for which the pruning approach we suggest (and presented previously in Keren et al., 2018) is *safe*, i.e., guaranteed to yield an optimal modification sequence. Such positioning allows us not only to justify our pruning approach for previously suggested *GRD* models and reduce the computational overhead of design for these model, but also to enrich the *GRD* framework with new modifications for which this approach is guaranteed to yield optimal results.

With the above objective in mind, we view the *GRD* task as a search in the space of modification sequences $\vec{m} \in \vec{\mathcal{M}}$ (and their corresponding goal recognition models) for a sequence that minimizes *WCD*. The operators are modifications in $\mathcal{M}$ that transition between goal recognition models. A node in the search tree represents a modification sequence $\vec{m} \in \vec{\mathcal{M}}$ that in turn represents a goal recognition model $R_0^{\vec{m}}$, which is the result of applying $\vec{m}$ to the initial model R_0. The constraint function ϕ induces the set $app_\phi(R)$ of applicable modifications of each model (node) $R \in \mathcal{R}^T$ (where $\mathcal{R}^T$ are the goal recognition models reachable from R_0 via redesign). The search continues until an optimal modification sequence is found.

A basic search method that can be used for *WCD* reduction is a breadth-first search (BFS) through the space of allowed modification sequences. The root node of the search is the empty sequence $\vec{m}_\emptyset$ and the initial model R_0. Each successor node appends a single modification to the sequence applied to the parent node. If the sequence is valid, it is added to a queue of sequences. Otherwise it is pruned, relying on the assumption that a valid modification sequence cannot have an invalid prefix. For each explored node we calculate *WCD* of the corresponding goal recognition model, updating the minimal *WCD* and optimal modification sequence when relevant. The search continues, at each level of the tree increasing the size of applied modification sequences until a model in which $wcd = 0$, or until there are no more nodes to explore. The result is a modification sequence $\vec{m}^{min} \in \vec{\mathcal{M}}$ that minimizes *WCD*. This approach is depicted in Figure 8. The upper portion of the figure shows the design process as a search in the modification space, while the lower part depicts the goal recognition models corresponding to each modification sequence.

In addition to satisfying the main objective of minimizing *WCD* under the specified constraints, this approach fulfills our secondary requirement, guaranteeing that among all the valid sequences which minimize *WCD*, the sequence selected is one with minimal length. This is thanks to the iterative nature of the proposed algorithm, which adds at each stage a separate node for each of the possible modifications appended to the parent node. We note that in the case of non-uniform modification costs, we are interested in a modification sequence with minimal cost. In this case, we can replace the BFS with a Dijkstra-based exploration. Instead of the queue described above, the sequences are maintained in a priority queue that returns at each stage the minimal-cost sequence.

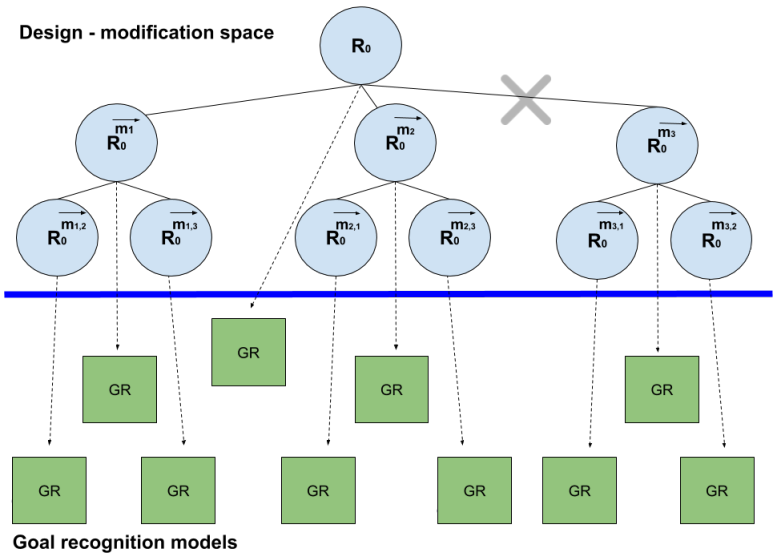


Figure 8: Design as a search in the modification space

The key question remaining is what modifications should be considered at each stage. A naïve approach, denoted *exhaustive-reduce*, considers all possible combinations of modifications. In the worst case, *exhaustive-reduce* examines all valid modification sequences.

Seeking improvement through pruning, we observe that the *WCD* value of a model cannot decrease if at least one maximal non-distinctive path remains non-distinctive in the modified model. Using $\vec{\Pi}^{nd}(R)$ to represent the set of non-distinctive paths in R, we let $\vec{\Pi}^{wcd}(R) \subseteq \vec{\Pi}^{nd}(R)$ represent the set of WCD *paths*, the set of non-distinctive paths with maximal cost. Recall that given a *GRD* model T, $\mathcal{R}^T$ represents the set of models reachable from R_0 in T. Lemma 4 guarantees that the *WCD* value of a model cannot decrease if at least one non-distinctive path in $\vec{\Pi}^{wcd}(R)$ remains non-distinctive in the modified model.

Lemma 4 *Given a* GRD *model* T *and two goal recognition models* $R, R' \in \mathcal{R}^T$*, if* $\exists \vec{\pi} \in \vec{\Pi}^{wcd}(R)$ *s.t.* $\vec{\pi} \in \vec{\Pi}^{nd}(R')$ *then* $\text{WCD}(R) \leq \text{WCD}(R')$*.*

Proof: Definition 10 sets *WCD* of a model as the maximal cost over non-distinctive paths $\vec{\Pi}^{nd}(R)$, which is the cost of $\vec{\pi}$. If $\vec{\pi}$ is non-distinctive in R' then *WCD* is at least the cost of $\vec{\pi}$ and $WCD(R) \leq WCD(R')$. ■

Typically, the *WCD* calculation of a goal recognition model, which can be performed by (for example) any of the compilation-based methods discussed in Section 4, reveals one of the *WCD* paths of the model. We refer to this path as the WCD *path* of model R and denote it by $\vec{\pi}^{WCD}(R)$. The *WCD* plans of model R, denoted by $\Pi^{WCD}(R)$, are a pair of complete plans that lead to two different goals, and at least one has the *WCD* path $\vec{\pi}^{WCD}(R)$ as its prefix, while the other has a prefix that shares an observable projection with $\vec{\pi}^{WCD}(R)$.

We exploit Lemma 4 to avoid unnecessary computations by pruning sequences that are guaranteed not to have an effect on the set of *WCD* plans of a model (in Figure 8

pruning is represented by the crossed-out branch). Accordingly, we propose the *pruned-reduce* algorithm. This algorithm extends *exhaustive-reduce* by pruning modifications that do not affect any action in the pair $\Pi^{WCD}(R)$.

In this work we focus on modifications that change the model's actions and observability. Accordingly, to formally characterize the effect a modification may have on a path, we let $A(m, R)$ represent the set of *affected actions* of modification m in model R. The set is comprised of two sets s.t. $A(m, R) = A^{mod}(m, R) \cup A^{S}(m, R)$: the set $A^{mod}(m, R)$ of actions whose implementation is changed by m, and the set $A^{S}(m, R)$ of actions whose observability changes as a result of applying m to R. Changing the implementation of action a is expressed by a change in either $pre(a)$, $add(a)$ or $del(a)$, which represent the preconditions, add effects and delete effects of a, respectively. Changing the observability of action a is expressed via the sensor model S as a change in the set of tokens $S(a)$ that may be emitted when a is executed, or in the set that may be emitted by any action which shares a token with a. For any node $R_0^{\vec{m}_{cur}}$, which represents the goal recognition model that results from applying $\vec{m}_{cur}$ to the initial model R_0, the *pruned-reduce* algorithm prunes modifications m for which the set of affected actions $A(m, R_0^{\vec{m}_{cur}})$ includes no action from the set of *WCD* plans.

Both the *exhaustive-reduce* and *pruned-reduce* algorithms are presented in Algorithm 2. When the flag b_{prune} is set to $false$ the iterative exploration is exhaustive, corresponding to *exhaustive-reduce*. Otherwise, modifications that do not affect any action in the pair $\Pi^{WCD}(R)$ are pruned (Line 12), which corresponds to *pruned-reduce*.[8]

We devote the next section to specifying the conditions under which the pruning performed by the *pruned-reduce* algorithm is guaranteed to preserve completeness and produce a valid modification sequence that minimizes the *WCD*.

5.1 Safe Pruning for *GRD* Using Generalized Strong Stubborn Sets

To justify the pruning performed by *pruned-reduce* we observe that it can be viewed as a form of partial order reduction used to reduce the size of the search space. Specifically, we show that for every node, the modifications not pruned form a strong stubborn set (Valmari, 1989). Originally introduced in the realm of computer-aided verification, strong stubborn sets guarantee that the subset of modifications applied at each node in the search are chosen in a way that preserves completeness. In our setting, a strong stubborn set of a node in the search is a subset of modifications which includes the first modification in a sequence that minimizes the *WCD* of the goal recognition model represented by the node.

Specifically, we use the formulation of *generalized strong stubborn sets* (GSSS) (Wehrle & Helmert, 2014), which considers planning tasks, and adapt it to our optimization task of finding a valid modification sequence to apply to the initial goal recognition model, with the aim of minimizing *WCD*.

Given a *GRD* model T, our objective is to find a *strongly optimal* modification sequence. We consider a modification sequence $\vec{m}$ to be strongly optimal for model R if it is a minimal-

8. In the algorithm description, we use the $\cdot$ operator to represent the concatenation of a single modification to a sequence. Therefore, $\vec{m} \cdot m$ is the modification sequence resulting from appending modification m to $\vec{m}$.

Algorithm 2 Redesign($T = \langle R_0, D \rangle$,$b_{prune}$)
(the b_{prune} flag distinguishes the *pruned-reduce* and *exhaustive-reduce* approaches)

```
1:  WCD^min = ∞ (init)
2:  m⃗^min = m⃗_∅ (init with empty sequence)
3:  Create a queue Q initialized to m⃗_∅ (Initialize queue with the empty sequence)
4:  while Q is not empty: do
5:     m⃗_cur ← Q.dequeue()
6:     if  WCD(R_0^{m⃗_cur}) < WCD^min then
7:        WCD^min = WCD(R_0^{m⃗_cur})
8:        m⃗^min = m⃗_cur
9:     end if
10:    for all m ∈ M_D do
11:       if {φ(m⃗_cur · m, R_0) = 1} (Check if the sequence is valid) then
12:          if  {not b_prune} or { ∃a ∈ A(m, R_0^{m⃗_cur}) s.t. a ∈ Π^{WCD}(R_0^{m⃗_cur})}  then
13:             enqueue m⃗_cur · m onto Q
14:          end if
15:       end if
16:    end for
17: end while
18: return  m⃗^min
```

length sequence that minimizes the *WCD* value of R.[9] A *terminal node* either minimizes *WCD* or has no valid successor modifications. The set $Opt(T)$ represents the strongly optimal solutions for model R_0 in T. Finally, a successor function $\succ$: $\mathcal{R} \rightarrow 2^{\mathcal{M}}$ yields for every node R a set of successor modifications.

We let $\succ_{pr}$ represent the successor function of the *pruned-reduce* algorithm, which yields for every node R a set of successor modifications $\succ_{pr}(R) \subseteq app(R)$. According to Algorithm 2, $\succ_{pr}$ prunes transitions that either violate any constraints or that have no effect on the set of *WCD* plans of the current model.

To minimize *WCD*, we need to make sure our pruning method is *safe*; i.e., it is guaranteed that an optimal solution for the given *GRD* task can still be found in the pruned search tree. Specifically, a successor function is safe if for every non-terminal model R, $\succ_{pr}(R)$ includes at least one operator that starts an optimal solution for R. As described by Wehrle and Helmert (2014), assuming all modifications have a non-zero uniform cost, this is a necessary criterion for safety.

For the sake of readability, the complete and formal definition of a GSSS in the context of *GRD* is given in Appendix B. We also show there that any pruning function which is guaranteed to yield a GSSS for each node encountered in the search is guaranteed to be safe. We devote the remainder of this section to a description of this formulation.

Given a *GRD* model $T = \langle R_0, D \rangle$, we let T_R represent the model that is the same as T but with R as its initial goal recognition model (instead of R_0). The set $\succ_{pr}(R)$ is a GSSS in R if it complies with three requirements. First, for every modification $m \in \succ_{pr}(R)$

9. When design costs are non-uniform, a strongly optimal sequence is an optimal sequence that minimizes design costs.

that is not applicable in R, the set contains a *necessary enabling set* for m and $Opt(T_R)$, which are the modifications that are applied before m in all sequences in $Opt(T_R)$ where m is applied. Second, for every modification $m \in \succ_{pr}(R)$ that is applicable in R, the set contains all modifications that *interfere* with m in R. Two modifications, $m_1, m_2 \in \mathcal{M}$, interfere in R if either one *disables* the other, or if they *conflict* in R. Modification m_1 disables m_2 in R if both are applicable in R but $m_2 \notin app(\delta(m_1, R))$. Two applicable modifications conflict in R if applying them in any order to R is valid but yields different non-distinctive paths (*i.e.*, $\vec{\Pi}^{nd}(R^{m_1,m_2}) \neq \vec{\Pi}^{nd}(R^{m_2,m_1})$). The final condition requires that $\succ_{pr}(R)$ contain at least one modification from at least one strongly optimal solution for T_R (all these concepts are exemplified in Appendix B).

The key difference between the original formulation by Wehrle and Helmert (2014) and our formulation of GSSSs for *GRD* is that the former requires non-interfering operators to yield the same state for different orders of application. Our focus is on the non-distinctive path set of a model. We therefore only require non-interfering modifications to yield the same set of non-distinctive paths for different orders of application. This is enough to guarantee that both models share the same *WCD* value.

The definition of GSSSs guarantees that for every non-terminal node, at least one permutation of a strongly optimal solution is not pruned. Note that verifying a given set to be a GSSS does not require complete knowledge of the sets of strongly optimal solutions. If the specified conditions can be verified for an over-approximation of the sets, they hold for the actual sets. Accordingly, in the next section (and Theorem 8 in particular), we show how the pruning performed by the *pruned-reduce* algorithm uses an over-approximation of the optimal solutions for every explored node while ensuring that successor pruning based on GSSSs is safe.

5.2 Independent, Persistent, Monotonic-nd *GRD* Models

We are now ready to present a characterization of a class of *GRD* models that can take advantage of the GSSS characterization and perform effective pruning when using the *pruned-reduce* algorithm. Clearly, to make use of GSSSs, an efficient method for their computation is required. We show that, in the case of *pruned-reduce*, the computation of a GSSS is given with a low computational overhead, as part of the *WCD* computation (Line 6 in Algorithm 2).

A useful observation here is that a *GRD* model and its constraint function induce a space of valid modification sequences. To guarantee that $\succ_{pr}$, the successor pruning function applied by *pruned-reduce*, is safe (i.e., that the space state induced by $\succ_{pr}$ includes a strongly optimal sequence), we require the *GRD* model to be *independent*, *persistent*, and *monotonic-nd*. Independence ensures that no two modifications interfere with each other (i.e., modifications of a sequence can be applied in any order to yield the same model). Persistence ensures that model constraints are not violated under any modification permutations. Finally, monotonicity ensures that no new non-distinctive paths are added as a result of modifying a model. After introducing these characterizations formally, we show this categorization is sufficient to guarantee the safety of $\succ_{pr}$. We then describe a variety of *GRD* models that comply with the specified requirements.

Definition 15 *A* GRD *model* T *is* independent *if for any modification* $m \in \mathcal{M}$*:*

- *The necessary enabling set of m and $Opt(T)$ is empty; and*
- *There are no modification m' and goal recognition model R s.t. m' interferes with m in R.*

Definition 16 *A* GRD *model T is* persistent *if for any goal recognition model $R \in \mathcal{R}^T$ and modification sequences $\vec{m}, \vec{m}' \in \vec{\mathcal{M}}$:*

- *If $\vec{m}$ is a prefix of $\vec{m}'$ and $\phi(\vec{m}, R) = 0$, then $\phi(\vec{m}', R) = 0$; and*
- *If $\vec{m}'$ is a permutation of $\vec{m}$ and $\phi(\vec{m}, R) = 1$, then $\phi(\vec{m}', R) = 1$.*

Persistence and independence are sufficient for pruning invalid sequences or sequences that are guaranteed to have a permutation which is not pruned. To prune modifications that do not affect the pair of *WCD* plans when no new non-distinctive paths can be added to the model, we define *monotonic-nd* models as follows:

Definition 17 *A* GRD *model T is* monotonic-nd *if for any goal recognition model $R \in \mathcal{R}^T$ and modification $m \in \mathcal{M}$, $\vec{\Pi}^{nd}(R^m) \subseteq \vec{\Pi}^{nd}(R)$.*

In a monotonic-nd model, valid modifications may only remove paths from the set of non-distinctive paths. Therefore, applying them cannot increase the *WCD* value of a model.

Lemma 5 *Given a* GRD *model T, if T is monotonic-nd then for every goal recognition model $R \in \mathcal{R}^T$ and modification $m \in \mathcal{M}$ s.t. $\phi(m, R) = 1$,*

$$\mathrm{WCD}(R^m) \leq \mathrm{WCD}(R)$$

Proof: According to Definition 17, modifications in a monotonic-nd model do not add non-distinctive paths. In particular, there are no non-distinctive paths with a cost higher than $WCD(R)$ that are added to the model and $WCD(R^m) \leq WCD(R)$. ∎

In a monotonic-nd model, applying a modification that does not modify the pair of *WCD* plans of the current model leaves *WCD* unchanged.

Corollary 2 *Let T be a monotonic-nd* GRD *model. For every modification $m \in \mathcal{M}$ and goal recognition model $R \in \mathcal{R}^T$ s.t. $\phi(m, R) = 1$, if $\forall a \in A(m, R)$ $a \notin \Pi^{\mathrm{WCD}}(R)$ then*

$$\mathrm{WCD}(R^m) = \mathrm{WCD}(R)$$

Proof: According to Lemma 5, in a monotonic-nd *GRD* model *WCD* cannot increase as a result of applying a valid modification, i.e., $WCD(R^m) \leq WCD(R)$. Since no action in the pair of plans $\Pi^{WCD}(R)$ is affected by m, both remain applicable in R^m and the *WCD* path $\vec{\pi}^{WCD}(R)$ remains non-distinctive in R^m. Moreover, since T is monotonic-nd no non-distinctive paths are added to the model. This, according to Lemma 4, means that *WCD* cannot decrease, i.e., $WCD(R^m) \geq WCD(R)$, leaving *WCD* unchanged. ∎

We note that the scope of models for which our suggested pruning approach is guaranteed to be safe is restricted to monotonic-nd models. However, as we will show in the next section, there are a variety of *GRD* models that comply with the above requirement, such as models in which sensor placement is applied to improve recognition. In particular, various

models suggested previously in the literature (e.g., Keren et al., 2014, 2015, 2016b; Son, Sabuncu, Schulz-Hanke, Schaub, & Yeoh, 2016; Mirsky, Stern, Gal, & Kalech, 2018) all support monotonic-nd models for which our suggested pruning can be applied to enhance performance.

Finally, we use the definitions above to specify the conditions under which the *pruned-reduce* algorithm is guaranteed to produce an optimal solution, i.e., the conditions under which $\succ_{pr}$ yields a GSSS for every model encountered in the search.

Theorem 5 *Given a* GRD *model* $T = \langle R_0, D \rangle$ *where* R_0 *is the initial model and* $D = \langle \mathcal{M}, \delta, \phi \rangle$*, if* T *is independent, persistent and monotonic-nd then for every model* $R \in \mathcal{R}^T$*, the set* $\succ_{pr}(R)$ *is a GSSS.*

The full proof for Theorem 5 can be found in Appendix C.

Theorem 5 specifies conditions that guarantee $\succ_{pr}$ generates a GSSS for every node $R \in \mathcal{R}^T$. According to Theorem 8 (in Appendix B), this guarantees that $\succ_{pr}$ is safe and the search will find an optimal modification sequence.

5.3 Modifications for Independent, Persistent, and Monotonic-nd *GRD* Models

Equipped with the conditions under which pruning performed by the *pruned-reduce* algorithm is safe, we now describe a general *GRD* model that complies with these conditions. We do this by specifying the constraints (ϕ) and modifications ($\mathcal{M}$) of this model and prove their compliance with the requirements.

The *GRD* model we characterize here, which was first presented by (Keren et al., 2018), is a generalization of multiple *GRD* models presented in previous works (Keren et al., 2014, 2015, 2016b) and for which $\succ_{pr}$ has been shown to produce optimal results. By positioning our pruning techniques in the context of GSSSs, we can support existing *GRD* models and provide a method for understanding the impact of previously presented design modifications. In addition, we can extend the *GRD* framework by offering new redesign modifications that preserve the completeness of the *pruned-reduce* algorithm.

We assume the analyzed recognition system is a *Cost-Bounded Goal Recognition* (*CB-GR*) model (as defined in Section 4), in which agents follow bounded sub-optimal plans to their goals. This means agents have a limited budget for diverging from optimal behavior (or no diversion budget if they are optimal). The constraint function we support here is *budget preserving*, enforcing a design budget which limits the number of modifications that can be applied. Such a budget can either limit the overall number of modifications or represent a separate budget for each modification type. The constraint function is also *cost preserving*, requiring that the maximal cost of a legal plan to any of the goals $g \in G$ does not increase.

The modification set consists of four modification types. Two modifications types, namely *action conditioning* (Keren et al., 2018) and its extreme special case, *action removal* (Keren et al., 2014), affect the applicability of agents' actions. The other two, namely *single-action sensor refinement* (Keren et al., 2018) and its extreme special case, *sensor placement* (Keren et al., 2015), modify the recognition system's sensor model.

Action conditioning, defined next, restricts the applicability of an action by adding preconditions necessary for its execution. Recall that $\mathcal{A}$ represents the set of all actions and $pre_R(a)$ is the set of preconditions of action a in model R (Section 3.1). Also, R^m is the goal recognition model that results from applying m to R.

Definition 18 *A modification m is an* action conditioning *modification if for every goal recognition model $R \in \mathcal{R}$, R^m is identical to R except that for every action $a \in \mathcal{A}$, $pre_R(a) \subseteq pre_{R^m}(a)$.*

Action conditioning can only remove paths from the set of applicable paths in the model, as we show next.

Lemma 6 *Let $T = \langle R_0, D \rangle$ be a* GRD *model where R_0 is the initial goal recognition model and $D = \langle \mathcal{M}, \delta, \phi \rangle$ is the design model. If $m \in \mathcal{M}$ is an action conditioning modification, then for every model $R \in \mathcal{R}^T$, $\vec{\Pi}^{leg}(R^m) \subseteq \vec{\Pi}^{leg}(R)$.*

Proof: Assume to the contrary that there is a path $\vec{\pi} = \langle a_1, \cdots, a_n \rangle$ s.t. $\vec{\pi} \in \vec{\Pi}^{leg}(R^m)$ but $\vec{\pi} \notin \vec{\Pi}^{leg}(R)$. Let i represent the first index s.t. the prefix $\langle a_1, \cdots, a_{i-1} \rangle$ is applicable in both models but $\langle a_1, \cdots, a_i \rangle$ is applicable in R^m but not in R. Since the effect of all actions is the same in both models, the state $s_{\langle a_1, \cdots, a_{i-1} \rangle}$ reached by the application of $\langle a_1, \cdots, a_{i-1} \rangle$ from the initial state is the same in both models. According to Definition 18, $pre_{R^m}(a_i) \subseteq pre_R(a_i)$. Therefore, if a_i is applicable in $s_{\langle a_1, \cdots, a_{i-1} \rangle}$ in R^m it is also applicable in R, contradicting our choice of i. ∎

When an action is conditioned (i.e., preconditions are added to it), some of the paths that include it may become invalid. Action conditioning can therefore be used to disallow specific paths from the model. Specifically, it can be used to disallow specific permutations of paths by forcing a partial order between actions. In Example 1, we described an extreme special case of action conditioning where actions can be removed from the model by applying *action removal* modifications. The barrier placed in Figure 2(c) reduces *WCD* from 4 in the original model to 0. Removing an action from the model is equivalent to adding an unsatisfiable precondition. Applying action conditioning to the same setting can be implemented by forcing an agent aiming at one of the goals to visit a specific location (e.g., to collect a key) before reaching her goal. For example, if we place a key in position $A1$ for G_0 or $E1$ for G_1, *WCD* is reduced to 0. As a generalization of action removal, action conditioning modifications extend the toolbox of design solutions that can be applied to a goal recognition setting and often correspond to modifications that are cheaper and easier to implement in practice.

As we show above, action conditioning may potentially decrease *WCD* by removing non-distinctive paths. In general, however, action conditioning may increase the optimal cost to a goal (and *WCD*) by eliminating all the optimal plans that lead to it. Returning to Example 1, if we apply action conditioning so as to force both agents to visit cell $D1$ before reaching their destinations, the optimal cost and *WCD* increase. We show next that in models with a cost-preserving constraint function, action conditioning modifications cannot add non-distinctive paths to the model.

Lemma 7 *Let $T = \langle R_0, D \rangle$ be a* GRD *model where R_0 is the initial goal recognition model and $D = \langle \mathcal{M}, \delta, \phi \rangle$ is the design model. If ϕ is cost preserving, then for any action conditioning modification $m \in \mathcal{M}$ and model $R \in \mathcal{R}^T$, $\vec{\Pi}^{nd}(R^m) \subseteq \vec{\Pi}^{nd}(R)$.*

Proof: Let $\Pi^{leg}_R(g) \subseteq \Pi_R(g)$ represent the legal plans to goal g in model R. Assume to the contrary that there is an action conditioning modification m s.t. there exists a goal recognition model R with a path $\vec{\pi} \in \vec{\Pi}^{nd}(R^m)$ but $\vec{\pi} \notin \vec{\Pi}^{nd}(R)$ (non-distinctive in R^m but distinctive in R). This implies that there are at least two legal plans $\pi^{'} \in \Pi^{leg}_{R^m}(g^{'})$ and $\pi^{''} \in \Pi^{leg}_{R^m}(g^{''})$ to two different goals $g^{'} \neq g^{''}$, where one has $\vec{\pi}$ as its prefix and the other shares the observable projection of $\vec{\pi}$ in R^m but not in R (w.l.o.g. we say that $\vec{\pi}$ is a prefix of $\pi^{'}$).

Under the assumption that m is an action conditioning modification, Lemma 6 ensures that both plans are valid in R (i.e., $\pi^{'} \in \Pi_R(g^{'})$ and $\pi^{''} \in \Pi_R(g^{''})$). Moreover, since ϕ is cost-preserving we know that both plans are legal in R ($\pi^{'} \in \Pi^{leg}_R(g^{'})$ and $\pi^{''} \in \Pi^{leg}_R(g^{''})$). The sensor models of R and R^m are the same. Therefore, $\vec{\pi}$ is a prefix of $\pi^{'}$ and shares an observable projection with a prefix of $\pi^{''}$. Hence $\vec{\pi}$ is non-distinctive in R, contradicting our choice of $\vec{\pi}$, and concluding our proof. ∎

Next, we consider modifications that improve the recognition system's ability to observe agent behavior by changing the system's sensor model. Specifically, we consider here *single-action sensor refinement* modifications (Keren et al., 2018), which determine the observability of a single action. Letting $A_S[a] \subseteq \mathcal{A}$ represent the set of actions (excluding action a) that share an observation token with action a according to sensor model S (Definition 12), we define single-action sensor refinement as follows:

Definition 19 *A modification m is a* single-action sensor refinement *modification if for every goal recognition model $R \in \mathcal{R}$, $A^{mod}(m, R) = \emptyset$ and:*
(1) For every action $a \in \mathcal{A}$, if $o_\emptyset \in S_{R^m}(a)$ then $o_\emptyset \in S_R(a)$; and
(2) There exists an action $a_m \in \mathcal{A}$ s.t. for every action $a \in \mathcal{A}$, $A_{S_{R^m}}[a] = A_{S_R}[a] \setminus a_m$.

Following Definition 19, we say that sensor model S' is an *s-refinement* of sensor model S if there is a sequence $\vec{m} = \langle m_1, \ldots, m_n \rangle$ of single-action sensor refinement modifications (hereafter referred to as sensor refinement) s.t. given any goal recognition model R with S as its sensor model, $R^{\vec{m}}$ is the same as R except for its sensor model S'.

The key to single-action sensor refinement is that some action is uniquely mapped to a (not necessarily new) token. Also, the token set of a given action can include the empty token in the refined model only if it was included in the original model. This formalism is meant to support both noisy and low-resolution sensors. To illustrate, consider the POND setting depicted in Figure 6(d), where the sensor model is noisy and non-deterministic, and movement from $D1$ to $D2$ can be perceived as movement to either $ZoneC$ or $ZoneD$. Sensor refinement can be applied to guarantee that the only token emitted in this case is the token corresponding to the action name.

A special case of sensor refinement is *sensor placement* (Keren et al., 2016a), which involves *exposing* a non-observable action (previously mapped to the null token) and mapping it to its unique token. To illustrate sensor placement, consider the NO setting depicted in Figure 6(b), where WCD can be reduced by exposing move$(C5, D5)$ (i.e., positioning a

sensor to observe this action). While in the original model the non-observable action produced the null token $o_\emptyset$, in the modified setting the action becomes observable (and emits a token corresponding to the action's name). Note that in this setting, while the *exhaustive-reduce* approach will consider all possible sensor placement and refinement modifications, our proposed *pruned-reduce* algorithm will prune modifications that have no effect on the current *WCD* path, such as the sensor placement modification that exposes move$(C3, B3)$.

Another example which illustrates both sensor refinement and action conditioning is given below.

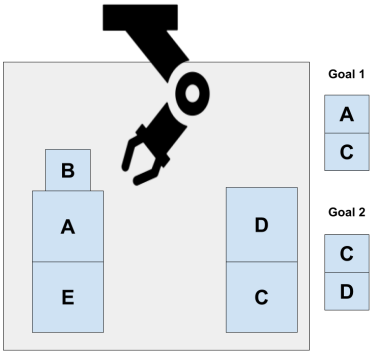


Figure 9: Action conditioning example

Example 3 *As an example of a controllable environment where action conditioning can be applied, consider Figure 9, which depicts a variation of the well-known BlockWords domain. In the example, a robot uses a gripper to move blocks so as to achieve one of two possible configurations. In our setting, in order to lift a block, the gripper needs to be adjusted to the width of the block. The blocks and gripper are hidden from the recognition system, which knows the initial setting (depicted on the left) and possible goals (depicted on the right), but only knows when some action is performed and not which one (the recognition system can hear when the gripper is performing some action, but cannot distinguish between different actions).*

In the original setting, the only way to guarantee correct goal recognition is by counting the number of performed actions, setting WCD *to* 3 *as the cost of the optimal plan to* Goal 1 *(i.e., the sequence* $\langle putOnTable(B), putOnTable(D), stackOn(A,C)\rangle$*). Placing a sensor on* Block B *(indicating when the block is picked up by the gripper) reduces* WCD *to* 1*, since optimal plans to both goals can start with placing* Block D *on the table (putOnTable(D)). If moving* Block B *is the next action performed, then the goal is* Goal 1 *(and* Goal 2 *otherwise). If, in addition to placing the sensor, we limit the robot's movement so it can only expand its gripper (initially closed), the only way to achieve* Goal 1 *is by first lifting* Block B*. In this case* $WCD = 0$*, and the first action is guaranteed to reveal the goal. Note that we cannot disallow moving any of the blocks while still guaranteeing that both goals can be achieved, exemplifying the importance of supporting the more general and versatile action conditioning modifications.*

To demonstrate the effect of pruning, the exhaustive-reduce *approach examines all modifications, including those that affect* Block E *(i.e., constraining it from being lifted or placing a sensor on the block to detect when it is moved). In contrast, the* pruned-reduce *algorithm prunes these modifications since no action that affects* Block E *is ever part of the* WCD *plans of the model.*

Sensor refinement (and placement) modifications never add non-distinctive paths. We prove this by showing that for every path, the number of paths that may share its observable projection cannot increase due to sensor refinement. Recall that $\vec{\Pi}^{leg}_R(G)$ represents the set of legal plans to all goals in a model R, and $G^O_R(\vec{o})$ is the set of goals satisfied by the observation sequence $\vec{o}$.

Theorem 6 *Given two goal recognition models $R, R' \in \mathcal{R}$ and a sensor refinement modification m, if $R' = m(R)$ and $\phi(m, R) = 1$ then*

$$\forall \vec{\pi} \in \vec{\Pi}^{leg}_R(G), \max_{\vec{o'} \in op_{R'}(\vec{\pi})} |G^O_{R'}(\vec{o'})| \leq \max_{\vec{o} \in op_R(\vec{\pi})} |G^O_R(\vec{o})|$$

The full proof for Theorem 6 can be found in Appendix D.

Corollary 1 *Given a sensor refinement modification m and two goal recognition models R and R', if $R' = m(R)$ and $\phi(m, R) = 1$ then $\vec{\Pi}^{nd}(R') \subseteq \vec{\Pi}^{nd}(R)$.*

Proof: Let $\vec{\pi}$ be a non-distinctive path in $\vec{\Pi}^{nd}(R')$. Recall that sensor refinement modifies only the sensor model and leaves the set of agent paths unchanged. Therefore, R and R' differ only in their sensor model and $\vec{\pi} \in \vec{\Pi}^{leg}_R(G) \cap \vec{\Pi}^{leg}_{R'}(G)$. From Theorem 6,

$$\max_{\vec{o'} \in op_{R'}(\vec{\pi})} |G^O_{R'}(\vec{o'})| \leq \max_{\vec{o} \in op_R(\vec{\pi})} |G^O_R(\vec{o})|$$

Therefore, $\vec{\pi} \in \vec{\Pi}_{nd}(R)$ (Definition 9). Since this holds for any non-distinctive path in R', we obtain that $\vec{\Pi}^{nd}(R^m) \subseteq \vec{\Pi}^{nd}(R)$. ∎

Corollary 1 guarantees that if the recognition system's sensor model is improved, the *WCD* value cannot increase. In particular, it establishes the relationship between any partially observable *GRD* model, where an action can share its emitted tokens with other actions (i.e., NO, POD and POND), and its fully observable (FO) version, where each action is mapped to a single and unique token. The *WCD* value of the model with full observability is therefore a lower bound on the *WCD* value of the same model with a partial or noisy sensor model.

Before concluding our description, we show that our model complies with the three requirements, described in Section 5.2, that guarantee the safety of $\succ_{pr}$. We start by showing that a model with only action conditioning and sensor refinement modifications and a cost-preserving constraint function is guaranteed to be monotonic-nd.

Lemma 8 *Given a* GRD *model $T = \langle R_0, D \rangle$ where R_0 is the initial model and $D = \langle \mathcal{M}, \delta, \phi \rangle$, if ϕ is cost-preserving and the modification set $\mathcal{M}$ consists only of action conditioning and sensor refinement modifications, then T is monotonic-nd.*

Proof: According to Lemma 7, under the assumption that the constraint function ϕ is cost-preserving, action conditioning modifications never add paths to the model, and non-distinctive paths in particular. Corollary 1 shows that this is true for sensor refinement modifications in any *GRD* model. Therefore, for any modification $m \in \mathcal{M}$ and model $R \in \mathcal{R}^T$, $\vec{\Pi}^{nd}(R^m) \subseteq \vec{\Pi}^{nd}(R)$ and T is monotonic-nd according to Definition 17. ∎

We now show that a *GRD* model which supports only action conditioning and sensor refinement modifications is independent.

Lemma 9 *Let $T = \langle R_0, D\rangle$ be a* GRD *model where R_0 is the initial goal recognition model and $D = \langle \mathcal{M}, \delta, \phi\rangle$ is the design model. If $\mathcal{M}$ consists only of sensor refinement and action conditioning modifications, then T is independent.*

The full proof for Lemma 9 is given in Appendix E.

The third condition requires the *GRD* model to be persistent, which depends on its constraint function.

Lemma 10 *Given a* GRD *model $T = \langle R_0, D\rangle$ where R_0 is the initial model and $D = \langle \mathcal{M}, \delta, \phi\rangle$, if ϕ is cost- and budget-preserving, and the modifications set $\mathcal{M}$ consists only of action-constraining and sensor refinement modifications, then T is persistent.*

Proof: To show the model is persistent, we consider a sequence $\vec{m}$ in T. If $\vec{m}$ is pruned because it violates the design budget, then adding modifications is not allowed, and there is no valid sequence with $\vec{m}$ as its prefix. If $\vec{m}$ is valid then the sequence and all of its permutations respect the budget constraints.

According to Definition 18, the set of actions affected by a modification is the same for any model $R \in \mathcal{R}$ (for sensor refinement the preconditions of all action are unchanged). Therefore, if sequence $\vec{m}'$ is a permutation of $\vec{m}$, it will induce the same set of paths, and legal paths in particular. If applying $\vec{m}$ maintains for every goal g the maximal cost of a legal plan, denoted by $C_A^{max}(g)$, so will the application of any of its permutations $\vec{m}'$. Similarly, if $\vec{m}$ causes $C_A^{max}(g)$ for one of the goals g to increase, appending modifications to $\vec{m}$ will not reduce these costs. Therefore any sequence that has $\vec{m}$ as its prefix will be invalid. This concludes our proof that T is persistent under the specified conditions. ∎

Finally, we combine the ingredients specified above to describe a *GRD* model for which the *pruned-reduce* algorithm is guaranteed to find an optimal modification sequence.

Theorem 7 *Given a* GRD *model $T = \langle R_0, D\rangle$, where R_0 is the initial model and $D = \langle \mathcal{M}, \delta, \phi\rangle$, if ϕ is both cost-preserving and budget-preserving and the modifications set $\mathcal{M}$ includes only action-constraining and sensor refinement modifications, then $\succ_{pr}(R)$ is safe.*

Proof: According to Theorem 8 (Appendix B), if a pruning function is guaranteed to produce a GSSS for every node in the search, it is safe. According to Theorem 5, if R is independent, persistent and monotonic-nd, the set $\succ_{pr}(R)$ is a GSSS for every model $R \in \mathcal{R}^T$. Lemma 8 guarantees that, under the specified constraints, T is monotonic-nd, while Lemma 9 guarantees it is independent. Finally, Lemma 10 concludes our proof by guaranteeing the model is persistent. ∎

6. Empirical Evaluation

In our empirical evaluation we instantiate a variety of independent, persistent and monotonic-nd *GRD* models that comply with the requirements specified above and show the effect of design on *WCD*.

GRD analysis consists of two core tasks, namely calculating *WCD* and minimizing it. Accordingly, we divide our evaluation into two main parts. In the first, we measure *WCD* in different goal recognition settings, and evaluate the methods we have suggested to calculate it. In particular, we compare our compilation-based approaches, and examine their efficiency for the different *GRD* settings. The second part of our analysis focuses on the design task, and evaluates *WCD* reduction achieved through redesign in various settings, using the different modification methods suggested in Section 5.3. We also examine the benefits of pruning using the *pruned-reduce* algorithm, and compare it to *exhaustive-reduce*.

All our examined problems represent independent, persistent, and monotonic-nd *GRD* models (as defined in Section 5.2). They vary both in the recognition system's sensor model, and in agents' bounds for diversion from optimal behavior. This allows us to examine how *WCD* increases when moving from fully observable to partially observable settings, and from settings with optimal agents to settings that support bounded-suboptimal agents. We first describe the dataset used for the evaluation, followed by the experimental setup and results for each of the empirical questions.

6.1 Dataset

Our dataset[10] consists of four uniform cost goal recognition domains adapted from Ramirez and Geffner (2009), namely Grid-Navigation (GRID), IPC-Grid$^+$ (GRID+), Blocks-words (BLOCK), and Logistics (LOG). We also examined three uniform cost domains adapted from Pereira et al. (2017), namely Intrusion Detection (I-DET), Depots (DEP), and Campus (CAM). All benchmarks are based on PDDL domains from the deterministic track of the International Planning Competitions (IPC).[11] Due to planner limitations, we created smaller instances for I-DET, DEP and LOG for which the corresponding planning problem could be solved optimally within 5 minutes.

For non-uniform action costs we evaluated the ISS-CAD (ISS) domain (E-Martín, R-Moreno, & Smith, 2015), which describes a space exploration setting where a robot needs to recognize the goal of an astronaut performing maintenance tasks in a spaceship. The domain operators involve moving between spaceship areas, inspecting components, measuring air and humidity levels, etc. The model propositions describe the location of astronauts, subsystems, tools, and components, the availability of instruments and components, the state (on, off, enabled, disabled) of the electrical equipment, etc.

We considered four observability settings (all described in Section 3), expressed via the recognition system's sensor model.

- Fully Observable (FO) - agent actions are fully observable by the recognition system.
- Non-Observable (NO) - agent actions are either observable or non-observable.

10. A full code base and dataset together with a *GRD* task generator can be found at https://github.com/sarah-keren/goal-recognition-design
11. http://icaps-conference.org/index.php/main/competitions

- Partially Observable Deterministic (POD) - each agent action is mapped to exactly one token (but many actions can be mapped to the same token). Non-observable actions are mapped to the null token.
- Partially Observable Non-Deterministic (POND) - a generalization of POD that allows each agent action to be mapped to one or more tokens (non-observable actions are mapped to the null token).

Each problem contains a domain description, a template for a problem description (without a goal), and a set of possible goals. For each benchmark we generated a separate FO problem for each pair of goals. For each FO problem we used a generator to create three NO settings, where the non-observable actions were randomly selected. For each NO setting, we created three POD settings, where the non-observable actions from the PO model remain non-observable and the remaining actions are randomly mapped to an observation token, chosen from an automatically generated set of possible tokens. The variety of tokens is generated by mapping each (grounded) action (e.g., $move(C1, D1)$) to its corresponding action name (e.g., $move$), to a subset of its parameters (e.g., $C1$), or to one of a set of random tokens (e.g., O_8). For each POD setting, we created three POND settings, where noise is added to the sensor model. This is implemented by adding to the set of possible observation tokens an additional token (each token emitted by an action in a POD setting may be emitted by the action in its corresponding POND settings). The set of possible tokens is the same as described for the POD setting.

This setup creates a dataset where each FO setting R_{FO} has a set of NO settings R_{NO} such that $S_{R_{FO}}$, the sensor model of R_{FO}, is an s-refinement (see Definition 19) of $S_{R_{NO}}$, the sensor model of R_{NO}. Similarly, each NO setting has a set of POD settings R_{POD} s.t. $S_{R_{NO}}$ is a s-refinement of $S_{R_{POD}}$, the sensor model of R_{POD}. Finally, each POD setting R_{POD} has a set of POND settings R_{POND}, s.t. $S_{R_{POD}}$ is a s-refinement of $S_{R_{POND}}$. Corollary 1 guarantees that WCD will not decrease as we move from FO to NO, from NO to POD, and from POD to POND (i.e., $WCD(R_{FO}) \leq WCD(R_{NO}) \leq WCD(R_{POD}) \leq WCD(R_{POND})$).

	Non-observable action file	Action tokens mapping file
FO	Empty file	Each action mapped to a unique token
NO	Random selection of non-observable actions	Observable actions mapped to their name
POD	NA	Observable actions mapped to a single token. Non-observable actions mapped to the null token.
POND	NA	Observable actions mapped to two distinct tokens. Non-observable actions mapped to the null token.

Table 2: Implementation of the various observability settings for GRD

To implement the variety of observability settings, we have two file types describing the recognition system's sensor model. For each NO setting we include a file specifying

the set of non-observable actions (with observable actions being the complementary set of actions). For the POD and POND settings, each problem description includes a file specifying the tokens emitted by each action. The variety of generated sensor models and their corresponding files is described in Table 2, where each column represents the type of file, and the rows are the different observability settings ('NA' stands for 'non-applicable'). Note that since we wanted to evaluate the efficiency of our calculation methods (as described in the next section), we implemented both the FO and NO settings using the action token files in addition to the more specialized implementation of each setting. In total, we evaluated 660 GRID, 528 GRID+, 264 BLOCK, 620 LOG, 264 I-DET, 824 DEP, 220 CAM, and 352 ISS problems.

6.2 Setup

***WCD* calculation:** We measure *WCD* in different *GRD* settings and evaluate the efficiency of our *WCD* calculation methods, presented in Section 4.2.

- *latest-split* (LS) - a compilation to classical planning for fully observable agents.
- *latest-expose* (LE) - a compilation to classical planning for non-observable actions.
- *common-declare* (CD) - a compilation to classical planning for partial and non-deterministic sensor models.

	LS	LE	CD	ASP
FO	+	+	+	+
NO	-	+	+	-
POD	-	-	+	-
POND	-	-	+	-

Table 3: Calculation methods examined for each observability setting

We performed a separate evaluation for each observability setting presented above, examining for each model the relevant calculation methods. As shown in Table 3 (where + indicates performed evaluations), for FO models we examined all three methods. For the NO settings we examined the LE and CD compilation techniques. The POD and POND settings were evaluated using CD. To examine the effect of a diversion bound on the *WCD* in *GRD* settings with bounded-suboptimal agents, we evaluated each setting with a bound of 1-3. In addition to our approaches, we examined the approach suggested by Son et al. (2016), where answer set programming is used to find and reduce *WCD*. This approach, which we denote as ASP, was used only for the fully observable models with optimal agents (the only setting supported by ASP).

For the solution of the compiled planning problems, we used the Fast Downward planning system (Helmert, 2006), running A^* with the LM-CUT heuristic (Helmert & Domshlak, 2009) for all but the ISS domain, for which the IPDB heuristic (Haslum, Botea, Helmert, Bonet, Koenig, et al., 2007) was used. Experiments were run on Intel(R) Xeon(R) CPU

X5690 machines, with a time limit of 30 minutes and memory limit of 2 GB.

Setup - *WCD* reduction: We implemented the four modification methods described in Section 5.3, namely action removal (AR), action conditioning (AC), sensor placement (SP), and single-action sensor refinement (SR). AR and AC were implemented by adding a special parameterized predicate *not-allowed* to the domain description. This predicate is specific to an action and its parameters. For example, in the GRID domain, disallowing movement from cell $C1$ to $C2$ is represented by the predicate *not-allowed-move*$(C1, C2)$.

We implemented AR, in which certain actions are disallowed altogether, by adding the corresponding *not-allowed* predicate to the initial state. We implemented AC by forcing pairs of actions to occur in a certain order. This was done by adding *not-allowed* predicates to the effects of actions. For example, to disallow the execution of *move*$(C2, C3)$ after *move*$(C1, C2)$, *not-allowed-move*$(C2, C3)$ is added to the effects of *move*$(C1, C2)$.

	AR	AC	SP	SR
FO	+	+	-	-
NO	+	+	+	-
POD	+	+	+	+
POND	+	+	+	+

Table 4: Modification methods examined for each setting

We implemented SP by removing an action from the non-observable action file for NO settings, and by setting the token of a non-observable action (previously mapped to the null token) to its corresponding action name for POD and POND settings. Similarly, SR was implemented by mapping an action to its name. The modification methods examined for each observability model are specified in Table 4 (+ indicates a method was used, and − otherwise).

To evaluate the effect of design on *WCD*, and particularly the effect of specific modification types, we examined all instances with a design budget of 4 assigned once for each modification type and once as an overall budget. The constraint function required the optimal cost to any of the goals to remain unchanged. To evaluate the power of pruning, each problem was run with both the *exhaustive-reduce* and *pruned-reduce* algorithms. During execution, we kept track of intermediate results, allowing us to examine the maximal reduction achieved by each design budget allocation.

6.3 Results

***WCD* calculation:** Table 5 shows the average running time (in seconds) of the different compilation techniques, as specified in Table 3, for *WCD* calculation for optimal agents. The shortest running time (indicating the most efficient method) is highlighted (for POD and POND, CD was the only method examined). The results show that where the more

specialized compilations can be applied, they perform better. Nevertheless, the most generic compilation, CD (*common-declare*), which can account for all settings, managed to solve all instances in the different observability settings, with a running time up to four times higher than with the most tailored approach.

	FO			NO		POD	POND
	LS	LE	CD	LE	CD	CD	CD
GRID	**0.7**	1.7	2.1	**1.5**	2.1	**2.56**	**7.22**
GRID+	**1.1**	1.8	1.9	**2.2**	2.4	**3.4**	**4.1**
BLOCK	**3.8**	6.2	8.1	**51.4**	52.5	**104.1**	**160.2**
LOG	**1.4**	1.7	1.9	**12.1**	22.2	**99.2**	**109.4**
I-DET	**0.3**	0.7	1.2	**1.5**	2.3	**2.4**	**2.4**
DEP	**4.2**	5.3	8.3	**8.9**	9.5	**9.7**	**9.9**
CAM	**0.8**	1.2	2.2	**2.3**	2.4	**2.3**	**2.3**
ISS	**2.7**	3.9	5.5	**4.9**	7.3	**13.4**	**13.5**

Table 5: Average running time (in seconds) of different *WCD* calculation methods

Not shown in the table are the results comparing the *latest-split* and ASP methods. This is because the only available benchmarks for running ASP comprised the complete goal sets presented by Ramirez and Geffner (2009), while our dataset included pairs of goals from different sources. For these problems, which we examined separately using both methods, ASP outperformed *latest-split* with a time reduction of up to 30%. However, this method is limited to the fully observable setting and could therefore not be used to evaluate our extended partially observable settings.

	FO				NO				POD				POND			
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
GRID	10.1 (1.0)	10.1 (1.0)	12.1 (1.0)	12.1 (1.0)	13.9 (1.0)	13.9 (1.0)	16.5 (1.0)	16.5 (1.0)	14.9 (1.0)	14.9 (1.0)	17.6 (1.0)	17.6 (1.0)	15.9 (1.0)	15.9 (1.0)	18.0 (1.0)	18.0 (1.0)
GRID+	9.5 (1.0)	10.1 (1.0)	11.2 (1.0)	12.1 (1.0)	9.6 (1.0)	10.7 (1.0)	11.2 (1.0)	12.7 (1.0)	9.51 (1.0)	10.03 (1.0)	11.8 (1.0)	12.6 (0.4)	10.5 (0.22)	NA	NA	NA
BLOCK	7.01 (1.0)	7.6 (1.0)	9.3 (1.0)	10.1 (1.0)	9.5 (0.8)	10.9 (0.8)	11.6 (0.4)	13.2 (0.3)	10.5 (1.0)	10.9 (0.8)	11.4 (0.4)	12.9 (0.2)	11.7 (0.14)	NA	NA	NA
LOG	0 (1.0)	3.5 (1.0)	6.2 (1.0)	8.3 (1.0)	9.5 (0.6)	11.3 (0.3)	12.2 (0.3)	13.0 (0.3)	11.7 (0.2)	11.9 (0.2)	12.3 (0.16)	13.2 (0.16)	13.4 (0.1)	NA	NA	NA
I-DET	1.1 (1.0)	2.7 (1.0)	3.8 (1.0)	4.9 (1.0)	3.5 (1.0)	4.5 (1.0)	3.3 (1.0)	5.1 (1.0)	3.1 (1.0)	3.6 (1.0)	4.8 (0.9)	5.5 (0.8)	6.3 (0.3)	NA	NA	NA
DEP	1.2 (1.0)	4.5 (1.0)	6.7 (1.0)	8.5 (1.0)	2.3 (0.8)	2.4 (0.7)	4.7 (0.3)	5.1 (0.2)	7.0 (0.2)	7.8 (0.17)	8.1 (0.17)	NA	7.5 (0.11)	NA	NA	NA
CAM	1.2 (1.0)	2.8 (1.0)	4.3 (1.0)	5.3 (1.0)	2.3 (1.0)	2.8 (1.0)	4.3 (1.0)	5.3 (1.0)	2.9 (0.7)	3.1 (0.4)	4.5 (0.4)	5.4 (0.4)	6.2 (0.2)	NA	NA	NA
ISS	42.01 (0.5)	43.02 (0.4)	NA	NA	45.67 (0.3)	46.1 (0.3)	NA	NA	45.67 (0.2)	NA	NA	NA	45.92 (0.2)	NA	NA	NA

Table 6: Average *WCD* value for different observability settings and diversion bounds

Table 6 shows the average *WCD* for each domain, based on observability and diversion bound. The ratio of solved instances is given in parentheses. We examined a diversion bound range of $0-3$, where 0 corresponds to optimal agents. Settings for which no instance was solved are marked 'NA'. The table shows the results acquired when using the most tailored approach for each observability setting (i.e., LS for FO, LE for NO and CD for

POD and POND). Note that *WCD* in the ISS domain is substantially higher since this is a non-uniform cost domain, and the indicated value reflects path cost rather than path length.

As expected, *WCD* increases for all domains as the diversion bound rises and/or observability worsens. It is noteworthy that the diversion bound and the introduction of noise to the sensor model had different effects in each domain. This is apparent when comparing the FO setting with optimal agents (FO-0) to the partially observable setting with optimal agents ($PONG$-0) and the fully observable setting for agents with the maximal examined diversion bound (FO-3). For example, in the GRID domain, partial observability had a stronger effect on *WCD*, while for $GRID+$, it was the diversion bound that caused a greater increase in *WCD*.

Considering the overhead caused by the extended observability and diversion bound, our compilation-based techniques were able to compute *WCD* for most examined instances. Failure in all instances of a particular setting (e.g., diversion bound 2 and 3 for the ISS domain) mainly reflected failure to complete the task before reaching the allocated time limit. The results indicate that the diversion bound had a stronger effect on computation than the introduction of noise to the sensor model (as seen in a comparison between observability models).

Results - *WCD* reduction

	FO				NO				POD				POND			
	Exhaustive		Pruned		Exhaustive		Pruned		Exhaustive		Pruned		Exhaustive		Pruned	
	Solved	Expanded	Solved	Expanded	Solved	Expanded	Solved	Expanded	Solved	Expanded	Solved	Expanded	Solved	Expanded	Solved	Expanded
GRID	148	135.25	160	48.5	53	240.1	72	168.3	142	341.3	148	198.4	232	437.42	288	252.57
GRID+	71	161.25	75	36.75	70	173.4	73	52.4	63	234.4	68	54.3	63	212.66	68	50.04
BLOCK	15	136.1	20	47.6	16	142.3	24	58.2	20	132.6	23	20.4	32	136.75	48	21.33
LOG	41	194.6	45	11.66	40	202.3	42	50.5	32	346.5	35	170.1	19	580.01	22	172.66
I-DET	42	145.5	42	35.5	39	192.3	41	54.2	38	243.5	40	197.5	31	580.01	33	214.65
DEP	47	134.3	51	94.6	30	145.3	34	103.2	25	245.4	23	145.6	21	580.01	24	187.6
CAM	57	142.3	62	62.2	57	246.5	63	98.4	45	145.3	48	101.3	32	203.4	42	84.01
ISS	60	40.0	120	7.9	53	54.4	119	50.1	68	73.6	195	64.3	74	168.75	224	156.63

Table 7: Efficiency gain of pruning

Table 7 compares the performance of the *exhaustive-reduce* (Exhaustive) and *pruned-reduce* (Pruned) algorithms. For each method and setting, the table shows the number of problems solved to completion within the time bound (Solved), and, for those solved by both methods, the average number of modification sequences expanded by each method (Expanded). Evaluation shows that for all domains, *pruned-reduce* solves more problems and expands fewer nodes than the exhaustive search, showing the efficiency gain brought about by *pruned-reduce*. This computational gain is emphasized in domains with large branching factors, such as ISS, where *pruned-reduce* examines as few as 20% of the nodes examined by the exhaustive approach.

Table 8 shows the average *WCD* reduction achieved in each domain and observability setting for optimal agents using each relevant modification method separately (AR, AC, SP and SR), and a combination of all modifications (TOT). In parentheses we show the ratio of problems for which *WCD* was reduced.

	FO			NO				POD				POND			
	AR	AC	TOT	AR	AC	SP	TOT	AR	AC	SR	TOT	AR	AC	SR	TOT
GRID	1.27	1.06	**1.29**	1.27	1.06	1.35	**1.37**	1.54	1.12	2.11	**2.21**	3.95	1.23	3.54	**4.1**
	(0.67)	(0.71)	(0.82)	(0.67)	(0.71)	(0.84)	(0.84)	(0.22)	(0.35)	(0.67)	(0.72)	(0.21)	(0.3)	(0.28)	(0.43)
GRID+	**0.57**	**0.57**	0.57	0.57	0.57	**1.1**	1.1	0.61	0.61	**3.2**	3.2	1.73	1.5	**3.67**	3.67
	(0.5)	(0.5)	(0.5)	(0.5)	(0.5)	(0.81)	(0.81)	(0.31)	(0.31)	(0.73)	(0.73)	(0.27)	(0.33)	(0.67)	(0.71)
BLOCK	0.43	0.44	**0.52**	1.11	0.5	1.82	**1.82**	0.45	0.32	**1.79**	1.79	0.55	0.31	**1.81**	1.81
	(0.4)	(0.4)	(0.61)	(0.5)	(0.65)	(0.76)	(0.76)	(0.8)	(0.3)	(0.85)	(0.85)	(0.51)	(0.32)	(0.85)	(0.85)
LOG	1.34	1.32	**1.78**	1.67	1.64	**2.53**	2.53	3.51	1.42	3.82	**3.91**	3.54	1.23	3.95	**4.17**
	(0.53)	(0.57)	(0.58)	(0.5)	(0.55)	(0.79)	(0.79)	(0.24)	(0.32)	(0.41)	(0.44)	(0.21)	(0.3)	(0.28)	(0.32)
I-DET	0	0	0	0	0	**1.73**	1.73	0	0	**1.9**	1.9	0	0	**1.65**	1.65
	(0)	(0)	(0)	(0)	(0)	(0.71)	(0.71)	(0)	(0)	(0.74)	(0.74)	(0)	(0)	(0.66)	(0.66)
DEP	0.15	**0.25**	0.25	0.15	0.25	**0.35**	0.35	0.11	0.12	**1.92**	1.92	0.11	0.12	**2.1**	2.1
	(0.11)	(0.23)	(0.23)	(0.11)	(0.23)	(0.5)	(0.5)	(0.1)	(0.19)	(0.61)	(0.61)	(0.1)	(0.21)	(0.63)	(0.63)
CAM	0.44	0.46	**1.01**	0.89	0.38	**0.92**	0.92	0.82	0.35	**2.1**	2.1	0.43	0.23	**1.9**	1.9
	(0.22)	(0.23)	(0.5)	(0.68)	(0.4)	(0.79)	(0.79)	(0.17)	(0.15)	(0.65)	(0.65)	(0.14)	(0.1)	(0.63)	(0.63)
ISS	0	0	0	0	0	**1.5**	1.5	0	0	**9.5**	9.5	0	0	**9.57**	9.57
	(0)	(0)	(0)	(0)	(0)	(0.17)	(0.17)	(0)	(0)	(0.15)	(0.15)	(0)	(0)	(0.15)	(0.15)

Table 8: Comparing modifications in different observability settings

The results point to the feasibility of reducing *WCD* through redesign. In fact, with the exception of the FO settings of ISS and I-DET, *WCD* can be reduced for at least 10% of the instances in each domain. For most FO domains, a combination of AR and the more refined AC approach yielded the best results. In addition, while sensor placement and refinement were the dominating approaches in most partially observable settings, for GRID and LOG a combined approach was better then any of the individual modifications. Moreover, for the domains where reduction was achievable, all of our examined approaches reduced *WCD* in some problems. This is important since in any realistic setting, the set of possible modifications depends on the application and the available design resources. The results show that even in domains where AR cannot be applied, the more refined AC approach can achieve *WCD* reduction in all cases where AR is beneficial.

7. Related Work

GRD involves the analysis of goal recognition models. As such it is closely related to and complements goal recognition research efforts, where the objective is to recognize the goals (or plans for plan recognition) of agents on the basis of their observed behavior. As mentioned in Section 2.2, there are many different models for goal and plan recognition, and many different tools have been developed for their solutions.

However related, *GRD* is a different task. While goal recognition aims at discovering the goals (and sometimes plans) of an agent by analyzing a specific observation sequence, *GRD* analyzes the goal recognition setting and offers a solution for facilitating online recognition. As such, the principles of *GRD* analysis can be adopted to any goal and plan recognition setting where the environment can be controlled. In particular, it can be applied to continuous domains where, as in discrete domains, the recognition system may fail because plan prefixes are shared by multiple goals (Vered & Kaminka, 2017). In such settings, redesign can be applied to reduce ambiguity and improve recognition. In particular, sensor refinement (Definition 19) can be viewed as a form of discretization (Kaminka et al., 2018), where the recognition system's sensor resolution is improved. Given a limited bud-

get of possible modifications, *GRD* analysis finds the optimal set of refinement operators to apply to the model so as to minimize *WCD* (i.e., to support early recognition).

GRD is also related to the problem of transparent (or explainable) planning (MacNally, Lipovetzky, Ramirez, & Pearce, 2018; Chakraborti, Sreedharan, Zhang, & Kambhampati, 2017), where acting agents behave in ways meant to help the observer recognize their objectives. Minimizing *WCD* guarantees a bound on the number of observations that need to be collected before recognition is achieved, thus enhancing collaboration between agents. Equally, *GRD* can be applied to the problem of finding behaviors that obfuscate an agent's goal. This is relevant to both adversarial agents (Kabanza et al., 2010; Keren et al., 2015; Masters & Sardina, 2017) and privacy-preserving agents (Keren et al., 2016b). Specifically, Keren et al. (2016b) show that agents can use *GRD* tools to identify *WCD* paths that lead to their destination while keeping it ambiguous as long as possible.

The first *GRD* model, suggested in Keren et al. (2014), accounts for fully observable deterministic settings where agents are assumed to behave optimally. This model can be modified by disallowing actions. Several variations of this setting have since been suggested. Son et al. (2016) modeled and solved the *GRD* setting specified in Keren et al. (2014) using Answer Set Programming (ASP). Keren et al. (2015) accounted for suboptimal agents, while Keren et al. (2016a, 2016b) supported a recognition system with partial and noisy sensor models that could be improved via sensor refinement. Mirsky et al. (2018) used plan libraries to represent agent behavior. Wayllace, Hou, Yeoh, and Son (2016, 2017) extended *GRD* to account for stochastic agent actions. In that work, instead of seeking the maximal non-distinctive path, the model accounts for probabilistic behavior by measuring the maximal expected cost an agent may incur before its goal is revealed. This approach is extended by Wayllace, Keren, Yeoh, Gal, and Karpas (2018) to account for partial observability in stochastic models. Finally, Ang, Chan, Jiang, and Yeoh (2017) support a game-theoretic version of *GRD*, in which a recognition system can alter the environment to facilitate the early detection of attacks by strategic adversarial agents that obfuscate their targets. In this work, we focus on *GRD* in settings where agents are either optimal or suboptimal, actions are deterministic, and the recognition system's sensor coverage (i.e., observability of agents and their movements) is possibly partial and noisy. For such settings, we present techniques for efficient *WCD* calculation based on compilations to planning, and provide an efficient design method while specifying the conditions under which it is guaranteed to yield optimal redesign sequences.

GRD can be viewed as a form of *environment design* (Zhang, Chen, & Parkes, 2009), which provides a framework for utility maximization where an interested party seeks optimal modifications to an environment. The agent is assumed to have a fixed form of decision making and the system can influence the agent's decisions via limited changes to the environment. Several special cases of this model have been suggested, including policy teaching (Zhang & Parkes, 2008), where the system aims to influence an agent's decisions by modifying its reward function; and equi-reward utility maximizing design (ER-UMD) (Keren, Pineda, Gal, Karpas, & Zilberstein, 2017), where design is used to maximize agent utility. In *GRD*, the recognition system is penalized for non-distinctive behavior and modifications are applied to minimize *WCD*.

GRD can also be seen as a form of mechanism design, where we want to influence future interactions between an agent and the goal recognition system. Specifically, our approach of

reducing *WCD* by eliminating legal actions can be seen as an aspect of social law (Shoham & Tennenholtz, 1995), with eliminated actions viewed as being made illegal. Agotnes, Van der Hoek, and Wooldridge (2012) use Kripke structures as their model and apply social laws to achieve a specified objective by eliminating some available transitions. A distance measure between structures is defined to assess the quality of a given law. However, that work offers no method for finding the social laws that fulfill the specified objective. We describe concrete methods for finding the optimal set of modifications to apply to a model so as to minimize *WCD*.

Our work, with its *GRD* framework, is not the first to highlight the influence of common prefixes on the task of goal recognition. Geib (2004) theoretically discusses the complexity of performing plan recognition by analyzing the prefixes of plans in a library. However, no concrete measure is given that can be used in our model. In the work of Geib (2009), the analysis of plan heads is proposed as a way to increase the efficiency of goal recognition given an observation set. Our work relates to a different setting that does not commit to a specific observation sequence and does not rely on a plan library being supplied.

Most of the tools we have created for assessing the *GRD* framework are based on compilations to classical planning. This approach is inspired by the work of Ramirez and Geffner (2009), which established the connection between goal recognition and automated planning. Similarly to Ramirez and Geffner (2009, 2010), we use planning domains to model the goal recognition environment. However, instead of analyzing a given observation, our work uses compilations to planning in order to assess a goal recognition setting. In fact, Ramirez and Geffner (2016) showed that the domain formulation subsumes the plan library approach through a compilation of libraries into STRIPS theories, making our approach applicable to any description of the set of legal plans using plan libraries.

As for *WCD* reduction, we apply a BFS search in the space of modifications, and offer ways to prune the search space. Son et al. (2016) use a declarative approach based on Answer Set Programming (ASP) to assess and reduce *WCD* in fully observable settings. As detailed in Section 6, this approach outperforms our compilation to planning approach, but fails to account for the partial observability of our model.

8. Concluding Remarks

We described the problem of goal recognition design (*GRD*) and presented techniques for assessing and redesigning different goal recognition settings for improved recognition. The overall objective of this work is to propose a generic framework for *GRD* in deterministic environments, and to offer efficient ways of both analyzing and optimizing a given goal recognition setting.

As a measure of model quality, we presented the worst case distinctiveness measure (*WCD*), which represents the maximal cost of a path an agent may follow before its goal is revealed. After formulating the *GRD* framework and providing a formal definition of the *WCD* measure, we offered methods for calculating *WCD*, mostly based on compilations of *GRD* problems to classical planning.

For optimizing *GRD* models, we offer a BFS-like search in the space of modifications, and suggest pruning as a way to reduce the size of the search space. Framing our pruning methods in the context of strong stubborn sets, we specified conditions under which our

pruning is safe. This allowed us not only to account for previously suggested *GRD* models, but also to extend the framework with new models that comply with these requirements.

Our empirical evaluation examined the value of *WCD* in different *GRD* settings, and the techniques we propose for its calculation and optimization. In particular, we examined the efficiency gain brought about by our pruning approach, and *WCD* reduction that can be achieved via design. We showed that our techniques can be used to efficiently evaluate and minimize *WCD* in a variety of goal recognition settings.

Looking forward, there are many ways to extend the *GRD* framework. One such direction involves accounting for agents with partial knowledge. To date, all existing models of *GRD* are based on the assumption that agents enjoy perfect observability of the environment and its dynamics. Enriching the *GRD* model to account for agents with partial information is a promising direction for future work, relevant to many realistic applications of *GRD*.

Another possible direction explores and enhances the design process. In this work, we used pruning to increase efficiency. It may be beneficial to combine different search space reduction techniques and heuristics in order to further reduce the computational overhead of the search in the modification space. In particular, while we offered methods for finding optimal redesign sequences, compromising optimality and applying approximate solutions can be useful in real-world applications, which tend to be extremely complex. Also, this work focused on modifications that change either the set of applicable actions at each state or the recognition system's sensor model. It may be valuable to examine other types of modifications, including modifications designed to change the initial state, adjust the position of the goals, *etc.*

Finally, while we evaluated *GRD* using goal recognition settings based on standard benchmarks from the automated planning literature, it would be useful to enrich the *GRD* framework by applying it to various real-world applications and examine the effect *GRD* may have in such settings.

Appendix A. The *common-declare* Compilation

The definition of the compilation *common-declare* described and justified in Section 4.2 is given below. For clarity, we highlight the parts of the compilation that deal with sub-optimal agents and that can be omitted for settings where agents are assumed to behave optimally.

The input to the model is a *CB-GR* model as described in Section 4. We let θ_i represent the diversion bound of each agent and $\vec{\Pi}^{\theta}(g_i)$ represent the set of legal paths to goal g_i, including all paths with cost up to $T_i = \mathcal{C}^*_A(g_i) + \theta_i$, where $\mathcal{C}^*_A(g_i)$ is the optimal cost to g_i. We define the *common-declare* compilation as follows:

Definition 20 (*common-declare* compilation) *For a* CB-GR *model* $R_{0,1} = \langle F, I, A, C_A, G = \{g_0, g_1\}, O, \langle \theta_0, \theta_1 \rangle, S \rangle$, *we create a planning problem* $P' = \langle F', I', A', G', C'_A \rangle$ *where:*

- $F' = \{f_0, f_1 \mid f \in F\} \cup \{declared_i \mid i \in \{0,1\}\} \cup \{performed_i^k \mid k \in O, i \in \{0,1\}\} \cup \{cost^c \mid c \in dom(C_A)\} \cup \{exposed\} \cup \{discounted_0\} \cup \{paid_0\} \cup \{\mathbf{time_i^t} \mid \mathbf{i} \in \{\mathbf{0}, \mathbf{1}\}\}$
- $I' = \{f_0, f_1 \mid f \in I\} \cup \{declared_0\} \cup \{declared_1\} \cup \{paid_0\} \cup \{\mathbf{time_0^0}, \mathbf{time_1^0}\}$

- $A' = \{A_i \mid i \in \{0,1\}\} \cup \{Declare^k_{0,1} \mid k \in O\} \cup \{Declare^{o^\emptyset}_0\} \cup \{Declare^k_i \mid k \in O, i \in \{0,1\}\} \cup \{Pay_0\} \cup \{PayDiscounted_0\} \cup \{\mathbf{Idle_i} \mid \mathbf{i} \in \{\mathbf{0},\mathbf{1}\}\}$ *where*
  - $A_i = \{a^k_i \mid a \in A, k \in S(a)\}$ *where*
    $a^k_0 = \langle \{f_0 \mid f \in pre(a)\} \cup \{declared_0\} \cup \{paid_0\} \cup \{\mathbf{time^t_0}\},$
    $\{f_0 \mid f \in add(a)\} \cup \{performed^k_0\} \cup \{cost^{C_A(a)}_0\} \cup \{\mathbf{time^{t+C_A(a)}_0}\},$
    $\{f_0 \mid f \in del(a)\} \cup \{declared_0\} \cup \{paid_0\} \cup \{\mathbf{time^t_0}\}\rangle$
    $a^k_1 = \langle \{f_1 \mid f \in pre(a)\} \cup \{declared_1\} \cup \{\mathbf{time^t_1}\},$
    $\{f_1 \mid f \in add(a)\} \cup \{performed^k_1\} \cup \{\mathbf{time^{t+C_A(a)}_1}\},$
    $\{f_1 \mid f \in del(a)\} \cup \{declared_1\} \cup \{\mathbf{time^t_1}\}\rangle$
  - $Declare^k_{0,1} = \langle \{performed^k_0, performed^k_1, \neg exposed\}, \{declared_0, declared_1, discounted_0\}, \{performed^k_0, performed^k_1\}\rangle$
  - $Declare^\emptyset_0 = \langle \{performed^{o^\emptyset}_0, \neg exposed\}, \{declared_0, discounted_0\}, \{performed^{o^\emptyset}_0\}\rangle$
  - $Declare^k_i = \langle \{performed^k_i\}, \{declared_i, exposed\}, \{performed^k_i\}\rangle$
  - $Pay^c_0 = \langle \{cost^c_0, declared_0\}, \{paid_0\}, \{cost^c_0\}\rangle$
  - $PayDiscounted^c_0 = \langle \{discounted_0, cost^c_0, declared_0\}, \{paid_0\}, \{discounted_0, cost^c_0\}\rangle$
  - $\mathbf{Idle^c_i} = \langle \{\mathbf{time^t_i}\}, \{\mathbf{time^{t+c}_i}, \mathbf{exposed}\}, \{\mathbf{time^t_i}\}\rangle$
- $g' = \{f_0 | f \in g_0\} \cup \{f_1 | f \in g_1\} \cup \{declared_0\} \cup \{paid_0\} \cup \{declared_1\} \cup \{\mathbf{time^{T_i}_i}\}$
- $C'_A(a) = \begin{cases} 0 & \text{if } Declare^k_1, Declare^k_{0,1}, Declare^{o^\emptyset}_0, A_0 \\ C_A(a) & \text{if } a \in A_1 \\ c & \text{if } Pay^c_0 \\ c(1-\epsilon) & \text{if } PayDiscounted^c_0 \\ \mathbf{c} & \text{if } \mathbf{Idle^c_i} \end{cases}$

Appendix B. Generalized Strong Stubborn Sets for *GRD*

Given a *GRD* model T, our objective is to find a *strongly optimal* modification sequence. A sequence $\vec{m}$ is optimal for model R if it is valid according to the specified constraints and minimizes *WCD*. A sequence $\vec{m}$ is strongly optimal for model R if it is a minimal cost (length) optimal sequence of R. A *terminal node* either minimizes *WCD* or has no valid successor modifications. Finally recall from Section 3.1 that $\mathcal{R}^T \subseteq \mathcal{R}$ represents the set of goal recognition models reachable from the initial model R_0 by applying a valid modification sequence.

Definition 21 (optimal solution, strongly optimal solution, terminal node) *Let $T = \langle R_0, D\rangle$ be a* GRD *model where R^0 is the initial goal recognition model and $D = \langle \mathcal{M}, \delta, \phi\rangle$ is the design model.*

An optimal solution *for a node $R \in \mathcal{R}^T$ and its corresponding* GRD *model $T_R = \langle R, D\rangle$, is a finite sequence $\vec{m} = \langle m_1, \cdots, m_n\rangle$ of modifications $m_i \in \mathcal{M}$ s.t. $\phi(\vec{m}, R) = 1$ and* $\text{WCD}(\delta(\vec{m}, R)) = \text{WCD}^{min}(T_R)$.

A strongly optimal solution *for node R is a sequence $\vec{m}$ s.t. for any optimal solution $\vec{m}'$ of node R, $|\vec{m}| \leq |\vec{m}'|$.*

A terminal node *is a node with no valid successors.*

The set $Opt(T)$ represents the strongly optimal solutions for a *GRD* model T. Next, we define a *successor pruning function* that yields for every node a set of successor nodes to be explored in the search. Recall from Section 3 that $app(R)$ represents the set of modifications that are applicable in a goal recognition model R.

Definition 22 (successor pruning function) *Let $T = \langle R^0, D\rangle$ be a* GRD *model where R^0 is the initial goal recognition model and $D = \langle \mathcal{M}, \delta, \phi\rangle$ is the design model. A* successor pruning function *for T is a function $\succ\colon \mathcal{R} \to 2^{\mathcal{M}}$ such that $\succ(R) \subseteq app(R)$ for all $R \in \mathcal{R}^T$.*

In *GRD*, a pruning function induces a subset of the applicable modifications at every encountered goal recognition model. The pruning performed by the *pruned-reduce* algorithm induces a successor pruning function, denoted by $\succ_{pr}$, which takes a node of the search (represented by a goal recognition model) and the set $\mathcal{M}$ of all possible transitions (modifications) and prunes transitions that either violate the constraints or have no effect on the set of *WCD* plans of the current model.

Since we want to minimize *WCD*, we need to guarantee our pruning method $\succ_{pr}$ preserves completeness, i.e., strongly optimal solutions can still be found in the search tree in spite of the pruning performed. Accordingly, a successor function is considered *safe* if is guaranteed that an optimal solution for the given *GRD* task can still be found in the pruned search tree.

Definition 23 (safe) *Let $\succ$ be a successor pruning function for a goal recognition design model T. We say that $\succ$ is safe if for every model $R \in \mathcal{R}^T$, the value of the optimal solution for R is the same when using the pruned state space induced by $\succ$ as when using the full state space induced by app.*

In general, it may be difficult to determine if a given successor pruning function is safe. In order to make the definition operational, a successor function is safe if for every non-terminal model R, the function includes at least one modification that starts an optimal solution for R . As described by Wehrle and Helmert (2014), under the assumption the model does not include zero-cost modifications, this is a necessary criterion for safety.

Proposition 1 *Let $\succ$ be a successor pruning function such that for every non-terminal node R, $\succ(R)$ contains at least one modification which is the first operator in an optimal solution for R. Then $\succ$ is safe.*

After defining the pruning method applied at each node in the search and the conditions that guarantee an optimal solution is found in spite of the pruning performed, we now describe strong stubborn sets in the context of *GRD*. Specifically, we need to prove $\succ_{pr}$ is safe by showing it produces a strong stubborn set for each encountered node. For this, we first define *interfering modifications* as pairs of modifications that affect each other's applicability. In the context of *GRD*, non-interfering modifications yield the same set of non-distinctive paths given any order of application. According to Definition 10, this is enough to guarantee the two models have the same *WCD* value.

Definition 24 (interference) *Let $m_1, m_2 \in \mathcal{M}$ be modifications of a* GRD *model T, and let R be a goal recognition model in $\mathcal{R}^T$. We say that m_1 and m_2 interfere in R if they are both applicable in R (i.e. $m_1, m_2 \in app_\phi(R)$), and*

- m_1 disables m_2 *in* R*, i.e.* $m_2 \notin app(\delta(m_1, R))$*, or*
- m_2 *disables* m_1*, or*
- m_1 *and* m_2 conflict *in* R*, i.e., both* $R^{m_1,m_2} = \delta(m_2, \delta(m_1, R))$ *and* $R^{m_2,m_1} = \delta(m_1, \delta(m_2, R))$ *are defined but* $\vec{\Pi}^{nd}(R^{m_1,m_2}) \neq \vec{\Pi}^{nd}(R^{m_2,m_1})$.

In our description of the fully observable setting in Example 1, we considered modifications that create physical barriers, corresponding to the removal of actions from the model. In this setting no modification can interfere with any other, since each modification relates to a different position in the grid. If we extend this setting by considering also the removal of barriers (i.e., extending the set of applicable actions), some modifications may interfere with each other. In this case, the order of applications becomes important.

The second ingredient for computing strong stubborn sets is *enabling sets*. An enabling set of a modification m and sequence set $\vec{\mathcal{M}}$ consists of those modifications that are applied before m in all sequences in $\vec{\mathcal{M}}$ where m is applied.

Definition 25 (necessary enabling set) *Let* $T = \langle R_0, D \rangle$ *be a goal recognition design model where* R_0 *is the initial goal recognition model and* $D = \langle \mathcal{M}, \delta, \phi \rangle$ *is the design model. In addition, let* $m \in \mathcal{M}$ *be one of the modifications, and* $\vec{\mathcal{M}}$ *be a set of modification sequences that are applicable in* R_0*.*

A necessary enabling set *(NES) for modification* m *and* $\vec{\mathcal{M}}$ *is a set* M *of modifications such that every modification sequence in* $\vec{\mathcal{M}}$ *which includes* m *as one of its operators also includes some operator* $m' \in M$ *before the occurrence of* m*.*

In Example 1 and Example 2, no modifications have an NES. As an example of a setting where modifications may have an NES, consider a variation of Example 2 where a nearby power outlet needs to be installed before a sensor can be placed.

Based on Definitions 24 and 25, we define a modification as *independent* if it can be applied to any goal recognition model without affecting the applicability of other modifications.

Definition 26 (independent modification) *Let* $T = \langle R_0, D \rangle$ *be a* GRD *model where* R_0 *is the initial goal recognition model and* $D = \langle \mathcal{M}, \delta, \phi \rangle$ *is the design model. We say that a modification* $m \in \mathcal{M}$ *is* independent *in* T *if:*

- *The necessary enabling set of* m *and* $Opt(T)$ *is empty; and*
- *For every goal recognition model* $R \in \mathcal{R}^T$*, there is no modification* m' *s.t.* m' *interferes with* m *in* R*.*

Finally, the formal definition of a generalized strong stubborn set (GSSS) is given below. The definition uses an *envelope* (Wehrle & Helmert, 2014) to describe a subset of the modification set $\mathcal{M}$ that is considered at each step and is known to be sufficient in the sense that it is safe to ignore all modifications which are not within the envelope. A GSSS is then a set of modifications that is guaranteed to contain a modification on a strongly optimal sequence, all modifications that are preconditions to the application of other modifications, and all modifications that interfere with a modification in the set.

Definition 27 (generalized strong stubborn set) *Let* $T = \langle R_0, D \rangle$ *be a goal recognition design model where* R_0 *is the initial goal recognition model and* $D = \langle \mathcal{M}, \delta, \phi \rangle$ *is the*

design model. A generalized strong stubborn set *(GSSS) in $R \in \mathcal{R}^T$ is a set of modifications $G \subseteq \mathcal{M}$ with the following properties.*

If R is a terminal model, every set $G \subseteq \mathcal{M}$ is a GSSS. Otherwise, G has an associated set of operators $E \subseteq \mathcal{M}$ (called the envelope of G). Let $T_R^E = \langle R, D^E \rangle$ be the goal recognition design model with an initial goal recognition model R and a design model $D^E = \langle E, \delta, \phi \rangle$, which is the model that results from replacing the set of all modifications $\mathcal{M}$ with the set $E \subseteq \mathcal{M}$. In addition, let $Opt(T_R^E)$ be the set of strongly optimal solutions for T_R^E, and $\mathcal{R}_{Opt}(T_R^E)$ be the set of goal recognition models that are visited by at least one sequence in $Opt(T_R^E)$. The following conditions must be true for G to be a $GSSS$:

- *E includes all operators from at least one strongly optimal solution for R (in the original model T).*
- *G contains at least one modification from at least one strongly optimal solution for T_R^E.*
- *For every modification $m \in G$ that is not applicable in R, G contains a necessary enabling set for m and $Opt(T_R^E)$.*
- *For every $m \in G$ that is applicable in R, G contains all operators in E that interfere with m in any model $R' \in \mathcal{R}_{Opt}(T_R^E)$.*

The definition above ensures that for every non-terminal node, at least one permutation of an optimal sequence is not pruned. Note that verifying a given set is a $GSSS$ does not require complete knowledge of the sets of strongly optimal solutions. If the specified conditions can be verified for an over-approximation of the sets, they hold for the actual sets. Accordingly, we show that the pruning performed by the *pruned-reduce* algorithm uses an over-approximation of the optimal solutions to yield a $GSSS$ for every explored node.

Concluding the description we show that successor pruning based on GSSSs is safe.

Theorem 8 *Let $\succ$ be a successor pruning function defined as $succ(R) = G(R) \cap app(R)$, where $G(R)$ is a generalized strong stubborn set in R. Then $\succ$ is safe.*

Proof: Let R be a goal recognition model and G be a GSSS in R. We show that if R is a non-terminal model, then G contains a modification which is the first modification in a strongly optimal solution for R. The claim then follows from Proposition 1.

Let E denote the envelope of G, and let T_R^E, $Opt(T_R^E)$ and $\mathcal{R}_{Opt}(T_R^E)$ be defined as in Definition 27. In the following, we refer to the four condition of Definition 27 as $C1 - C4$.

Every solution of T_R^E (for its initial model R) is a solution for model R of T because the two tasks only differ in their initial models and in the set of modifications, and the modification set E of T_R^E is a subset of operator set $\mathcal{M}$. Further, because of $C1$ at least one strongly optimal solution for R in T is also present in T_R^E. It follows that strongly optimal solution for T_R^E are also strongly optimal for model R in T. Hence, it is sufficient to show that G contains the first operator of some strongly optimal solution for T_R^E.

Let $\vec{m} = \langle m_1, \ldots, m_n \rangle$ be a strongly optimal solution for T_R^E of which at least one modification is contained in G. Such a solution must exist because of $C2$. Let $k \in \{1 \ldots n\}$ be the minimal index for which $m_k \in G$.

We show by contradiction that m_k is applicable in R. Assume it is not applicable. Because $o_k \in G$, $C3$ guarantees that G contains a necessary enabling set for m_k and $Opt(T_R^E)$. $Opt(T_R^E)$ contains the sequence $\vec{m}$, so by the definition of necessary enabling sets,

G must contain one of the operators in $\vec{m}$ that occur before m_k . This is a contradiction to the choice of k. It follows that m_k is applicable in R.

Let $R^0, \ldots, R^n$ be the sequence of models visited by $\vec{m} : R^0 = R$, and $R^i = \delta(m_i, R^{i-1})$ for all $1 \leq i \leq n$. Because $\vec{m}$ is strongly optimal, all these nodes are contained in $R_{Opt}(T_R^E)$. It follows that m_k does not interfere with any of the operators $m_1, \ldots, m_{k-1}$ in any of the models R_j: if it did, then from $C4$ (with $m = m_k$), the interfering operators would be contained in G, again contradicting the minimality of k.

We now show that if m_k is not already the first operator in $\vec{m}$, we can use the method suggested by Wehrle and Helmert (2014) to shift m_k to the front of $\vec{m}$. Consider the case where $k > 1$ (otherwise m_k is already at the front). We already know that m_k is applicable in R^0; also, m_1 is applicable in R^0 (or $\vec{m}$ would not be applicable in R). Because m_1 and m_k do not interfere in R^0 , it follows that m_1 does not disable m_k, and hence m_k is also applicable in R^1. This argument can be repeated to show that m_k is applicable in all models R^j with $j < k$. In particular, it is applicable in R^{k-2}, the node right before the one in which m_k is applied in $\vec{m}$. Therefore, in this node, m_{k-1} and m_k are both applicable and do not interfere, so they can be applied in either order, leading to a model with the same set of non-distinctive paths: $\vec{\Pi}^{nd}(R^k) = \vec{\Pi}^{nd}(\delta(m_k, \delta(m_{k-1}, R^{k-2}))) = \vec{\Pi}^{nd}(\delta(m_{k-1}, \delta(m_k, R^{k-2})))$. Hence we can swap m_{k-1} and m_k in $\vec{m}$ and still have a valid sequence.

This argument can be repeated to swap m_k to position $k-2$, $k-3$ and so on, until we end up with the sequence $\vec{m}' = \langle m_k, m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_n \rangle$.

Because $\vec{m}'$ is a permutation of $\vec{m}$, it has the same cost as $\vec{m}$ and consists of the same set of operators. It also induces the same set of non-distinctive paths and the same *WCD* value and is therefore also a strongly optimal solution for T_R^E. Its first operator, m_k, belongs to G. We have found a strongly optimal solution for T_R^E whose first operator is in G, concluding the proof. ∎

Appendix C. Full Proof for Theorem 5

Below is the full proof for Theorem 5 given in Section 5.2 to show that under the specified conditions, $\succ_{pr}$ yields a GSSS for every model encountered in the search.

Proof: In the following, we refer to the four conditions of Definition 27 (given in Appendix B) as $C1 - C4$. Assume, by way of contradiction, there exists a model $R \in \mathcal{R}^T$ s.t. $\succ_{pr}(R)$ is not a *GSSS*.

If R is a terminal node, then all subsets of $\mathcal{M}$ form a GSSS for R, so R must be non-terminal and violate one of the conditions $C1 - C4$.

Condition $C1$ requires that the envelope of R include at least one strongly optimal plan for R. As described in Line 10 of Algorithm 2, *pruned-reduce* considers all modifications at every iteration. The envelope E for every node is the same and includes all actions and therefore $C1$ is not violated. According to Definition 26, since T is independent the necessary enabling set of any $m \in \mathcal{M}$ and $Opt(T)$ is empty, and condition $C3$ is not violated. Similarly, condition $C4$ is not violated since there are no two modifications that interfere in an independent model.

To show that $C2$ cannot be violated under the specified assumptions, we show that for every strongly optimal sequence that is pruned we can use the method suggested by

Wehrle and Helmert (2014) to construct a strongly optimal sequence that is not pruned. We let $\vec{m} = \langle m_1, \ldots, m_n \rangle$ represent a strongly optimal solution for R s.t. $m_1 \notin \succ_{pr}(R)$ (the sequence is pruned). According to the *pruned-reduce* algorithm, modification m_1 is not in $\succ_{pr}(R)$ either because $\phi(\delta(m_1, R)) = 0$ (applying m_1 to R is invalid) or because m_1 has no effect on the *WCD* plans of R.

If applying m_1 is invalid, then, under the assumption T is persistent, applying $\vec{m}$ to R is also invalid, contradicting our choice of $\vec{m}$. Otherwise, let k be the index of the first modification in $\vec{m}$ such that m_k affects the set of *WCD* plans of R (i.e., $\exists a \in A(m_k, R)$ s.t. $a \in \Pi^{WCD}(R)$). If no such modification exists in $\vec{m}$ then the pair $\Pi^{WCD}(R)$ remains unchanged after executing $\vec{m}$. According to Corollary 2, under the assumption T is monotonic-nd, $WCD(R) = WCD(R^{\vec{m}})$ and $\vec{m}$ is not strongly optimal, again contradicting our choice of $\vec{m}$.

If such a modification exists, then, under the assumption T is persistent, we know that m_k is applicable at R (the beginning of the sequence $\vec{m}$). If m_k is the first operator in $\vec{m}$ it will be included in $\succ_{pr}(R)$, contradicting our choice of $\vec{m}$.

Otherwise, consider the case where $k > 1$. Let $R^0, \ldots, R^n$ be the sequence of nodes visited by $\vec{m}$: $R^0 = R$ and $R^i = m_i(R^{i-1})$. We know that m_1 is applicable in R (or $\vec{m}$ would not be applicable in R). Because m_1 and m_k are independent and do not interfere in R, it follows that m_k is also applicable in R^1. This argument can be repeated to show that m_k is applicable in all models R^j with $j < k$. In particular, it is applicable in R^{k-2}, the node right before the one in which it is applied in $\vec{m}$. Therefore, in this model, m_{k-1} and m_k can be applied in either order, leading to a model with the same set of non-distinctive paths. Hence we can swap m_{k-1} and m_k and still have a valid sequence.

This argument can be repeated to swap m_k to position $k-2$, $k-3$ and so on, until we end up with the sequence $\vec{m}' = \langle m_k, m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_n \rangle$. Under the assumption ϕ is consistent, we know that since $\vec{m}$ is valid, $\vec{m}'$ is also valid. Because $\vec{m}'$ is a permutation of $\vec{m}$ with the same cost as $\vec{m}$ that yields the same set of non-distinctive paths, $\vec{m}'$ is also a strongly optimal solution. In addition, due to our choice of k we know that m_k is the first operator of a strongly optimal plan $\vec{m}'$ included in $\succ_{pr}(R)$. The set $\succ_{pr}(R)$ is therefore a GSSS for R.

The same argument is repeated for sequence $\langle m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_n \rangle$ and model R^k and all the sequences that follow. If we reach n, we have found a permutation of $\vec{m}$ which is strongly optimal and is not pruned from the search. If, on the other hand, we reach an index $j < n$ s.t. there is no modification in the remaining sequence on the *WCD* plans of the current model, then $\vec{m}$ is not strongly optimal, contradicting our choice of $\vec{m}$ and concluding our proof.

■

Appendix D. Full Proof for Theorem 6

Below is the full proof for Theorem 6, given in Section 5.3 to show that sensor refinement (and placement) modifications never add non-distinctive paths to a model.

Proof: By definition, sensor refinement only changes the system's sensor model and not the set of legal paths to each goal. According to Definition 2, the set of possible observation sequences generated by the execution of a path $\vec{\pi}$ is $op_R(\vec{\pi})$ in R and $op_{R'}(\vec{\pi})$ in R'.

Assume to the contrary that m is a sensor refinement modification and $R' = m(R)$ but

$$\exists \vec{\pi} \in \vec{\Pi}^{leg}_R(G) \text{ s.t. } \max_{\vec{o'} \in op_{R'}(\vec{\pi})} |G^O_{R'}(\vec{o'})| > \max_{\vec{o} \in op_R(\vec{\pi})} |G^O_R(\vec{o})| \quad (2)$$

Since $\vec{\pi} \in \vec{\Pi}^{leg}_R(G)$, there is at least one goal $g_{\vec{\pi}} \in G$ which is the actual goal of the acting agent and which is satisfied by all the observable projections $\vec{o} \in op_R(\vec{\pi})$ (by Definition 7). Our assumption implies that there is at least one other goal g_o ($g_o \neq g_{\vec{\pi}}$) s.t. at least one observable projection of $\vec{\pi}$ satisfies both $g_{\vec{\pi}}$ and g_o in R', but no such observable projection in R, otherwise Eq. 2 fails to hold.

We now look at the path $\vec{\pi}_{g_o} \in \vec{\Pi}^{leg}_R(g_o)$ that shares a common observable projection with $\vec{\pi}$ in R' (but not in R). $\vec{\pi}_{g_o}$ exists in both models since the legal paths are the same in both. The paths $\vec{\pi}$ and $\vec{\pi}_{g_o}$ share an observable projection in R'. Therefore, for all prefixes of $\vec{\pi}$ there is at least one prefix of $\vec{\pi}_{g_o}$ with which it shares an observable projection in R'. Let i represent the index of the first action a_i in $\vec{\pi}$ s.t. the prefix $\vec{\pi}_{1\cdots i} = \langle a_1, \cdots, a_i \rangle$ of π shares no common observable projection with a prefix of $\vec{\pi}_{g_o}$ in R (while there is always one for R', due to the way $\vec{\pi}$ is selected). According to Definition 2, this can happen in one of two cases.

In the first case, $o_\emptyset \in S'(a_i)$ but $o_\emptyset \notin S(a_i)$ and $op(\vec{\pi}_{1\cdots i}) \cap op(\vec{\pi}_{1\cdots i-1}) \neq \emptyset$. Since a_i is the first action for which no common observable projection exists in R, we know that $op(\vec{\pi}_{1\cdots i-1})$ shares its observable projection with some prefix of $\vec{\pi}_{g_o}$ both in R and R'. This, however, contradicts the assumption that if S' is a refinement of S, then if $o_\emptyset \in S'(a)$, it is also the case that $o_\emptyset \in S(a)$.

Otherwise, there is at least one action in $\vec{\pi}_{g_o}$ that shares a common token with a_i in R' and not in R. According to Definition 19, there exists an action $a_m \subseteq \mathcal{A}$ s.t. for every action, $a \in \mathcal{A}$ $A_{S_{R'}}[a] = A_{S_R}[a] \setminus a_m$. This means that for every action $a \in A$, $A_{S'}[a] \subseteq A_S[a]$. This is particularly true for a_i, thus contradicting our choice of a_i and concluding our proof. ∎

Appendix E. Full Proof for Lemma 9

Below is the full proof for Lemma 9, given in Section 5.3 to show that a *GRD* model that supports only action conditioning and sensor refinement modifications is independent.

Proof: According to Definition 26, any modification in an independent model has an empty enabling set and does not interfere with any other modifications. The first condition is satisfied by Definitions 18 and 19, according to which both action conditioning and sensor refinement modifications have no preconditions and are applicable in any model $R \in \mathcal{R}$. In particular both R^{m_2,m_1} and R^{m_1,m_2} are well-defined and belong to $\mathcal{R}^T$.

To guarantee the second condition, we assume to the contrary that $\exists R \in \mathcal{R}^T$ and modifications m_1 and m_2 s.t. the modifications interfere in R. We have seen that all modifications are applicable in all models, so the modifications do not disable each other. Therefore m_1 and m_2 conflict and $\vec{\Pi}^{nd}(R^{m_1,m_2}) \neq \vec{\Pi}^{nd}(R^{m_2,m_1})$. W.l.o.g we let $\vec{\pi}$ represent a path that belongs to $\vec{\Pi}^{nd}(R^{m_1,m_2})$ but not to $\vec{\Pi}^{nd}(R^{m_2,m_1})$. Because $\vec{\pi}$ is non-distinctive in R^{m_1,m_2} we know that there exists a path $\vec{\pi}'$ s.t. $\vec{\pi}$ and $\vec{\pi}'$ share an observable projection that satisfies at least two goals in R^{m_1,m_2} but not in R^{m_2,m_1} (otherwise $\vec{\pi}$ would be non-distinctive in both models). This can happen in one of two cases: either both $\vec{\pi}$ and $\vec{\pi}'$ are

valid in R^{m_1,m_2} but one of them is invalid in R^{m_2,m_1}, or the two paths share an observation sequence in R^{m_1,m_2} but not in R^{m_2,m_1}. Lemma 6 assures that the first case cannot occur and if both $\vec{\pi}$ and $\vec{\pi}'$ are valid in R^{m_1,m_2} they are valid in R^{m_2,m_1}.

In the second case, $\vec{\pi}$ and $\vec{\pi}'$ are valid in both models but share an observable projection in R^{m_1,m_2} but not in R^{m_2,m_1}. We let i represent the index of the first action in $\vec{\pi} = \langle a_1, \ldots, a_n \rangle$ s.t. $\langle a_1, \ldots, a_{i-1} \rangle$ is non-distinctive in both models but $\langle a_1, \ldots, a_i \rangle$ is distinctive in R^{m_2,m_1} but not R^{m_1,m_2} (recall that the empty sequence is non-distinctive). This means that either a_i is non-observable (mapped to the empty token) in R^{m_1,m_2} but not in R^{m_2,m_1} or that there is at least one action with which a_i shares a token in R^{m_1,m_2} but not R^{m_2,m_1} (i.e. $\exists a \in A$ s.t. $a \in A_{S_{R^{m_1,m_2}}}[a_i] \setminus A_{S_{R^{m_2,m_1}}}[a_i]$).

According to Definition 18, action conditioning modifications do not affect the observability of an action. The observability of a path therefore depends only on any sensor refinement modifications applied. Definition 19 states that an action cannot be mapped to the null token as a result of sensor refinement. Therefore, a_i is non-observable only if it is non-observable in both models, and the first case cannot occur. The second case is impossible since according to Definition 19, $A_{S_{R^{m_1,m_2}}}[a_i] = A_{S_R}[a_i] \setminus \{a_{m_1}\} \setminus \{a_{m_2}\}$, which is equal to $A_{S_R}[a_i] \setminus \{a_{m_2}\} \setminus \{a_{m_1}\}$, where a_{m_i} is the action modification m_i assigns to a separate token. This contradicts our choice of a_i and concludes our proof.

■

References

Agotnes, T., Van der Hoek, W., & Wooldridge, M. (2012). Conservative social laws. In *Proceedings of the European Conference on Artificial Intelligence (ECAI 2012)*.

Albrecht, D. W., Zukerman, I., & Nicholson, A. E. (1998). Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, *8*(1-2), 5–47.

Aldinger, J., Mattmüller, R., & Göbelbecker, M. (2015). Complexity of interval relaxed numeric planning. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence*.

Ang, S., Chan, H., Jiang, A. X., & Yeoh, W. (2017). Game-theoretic goal recognition models with applications to security domains. In *Proceedings of the International Conference on Decision and Game Theory for Security*.

Boddy, M. S., Gohde, J., Haigh, T., & Harp, S. A. (2005). Course of action generation for cyber security using classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2005)*.

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*(1-2), 5–33.

Bui, H. H. (2003). A general model for online probabilistic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2003)*.

Carberry, S. (2001). Technique for plan recognition. *User Modeling and User-Adapted Interaction*, *11*(1-2), 31–48.

Chakraborti, T., Sreedharan, S., Zhang, Y., & Kambhampati, S. (2017). Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2017)*.

Cohen, P. R., Perrault, C. R., & Allen, J. F. (1981). Beyond question-answering. Tech. rep., DTIC Document.

E-Martín, Y., R-Moreno, M. D., & Smith, D. E. (2015). Practical goal recognition for ISS crew activities. In *Proceedings of the International Workshop of Planning and Scheduling for Space (IWPSS 2015)*.

Fikes, R. E., & Nilsson, N. J. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, *2*(3), 189–208.

Freedman, R. G., & Zilberstein, S. (2017). Integration of planning with recognition for responsive interaction using classical planners. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2017)*.

Geffner, H., & Bonet, B. (2013). A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *8*(1), 1–141.

Geib, C. W. (2004). Assessing the complexity of plan recognition. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2004)*.

Geib, C. W. (2009). Delaying commitment in plan recognition using combinatory categorial grammars. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2009)*.

Ha, E., Rowe, J. P., Mott, B. W., & Lester, J. (2011). Goal recognition with Markov logic networks for player-adaptive games. In *Proceedings of the Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE-11)*.

Han, T., & Pereira, L. (2011). Context-dependent incremental intention recognition through Bayesian network model construction. In *Proceedings of the UAI Bayesian Modeling Applications Workshop (UAI-AW 2011)*.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

Haslum, P., Botea, A., Helmert, M., Bonet, B., Koenig, S., et al. (2007). Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2007)*.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)*, *26*, 191–246.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Hong, J. (2001). Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research (JAIR)*, *15*, 1–30.

Jarvis, P. A., Lunt, T. F., & Myers, K. L. (2004). Identifying terrorist activity with AI plan recognition technology. In *Proceedings of the National Conference on Innovative Applications of Artificial Intelligence (IAAI 2004)*.

Kabanza, F., Bellefeuille, P., Bisson, F., Benaskeur, A. R., & Irandoust, H. (2010). Opponent behaviour recognition for real-time strategy games. In *Proceedings of the Workshop on Plan, Activity, and Intent Recognition (PAIR-AAAI 2010)*.

Kaluza, B., Kaminka, G. A., & Tambe, M. (2011). Towards detection of suspicious behavior from multiple observations. In *Proceedings of the Workshop on Plan, Activity, and Intent Recognition (PAIR-AAAI 2011)*.

Kaminka, G. A., Vered, M., & Agmon, N. (2018). Plan recognition in continuous domains. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2018)*.

Kautz, H., & Allen, J. F. (1986). Generalized plan recognition. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 1986)*.

Kautz, H., Etzioni, O., Fox, D., Weld, D., & Shastri, L. (2003). Foundations of assisted cognition systems. *University of Washington, Computer Science Department, Technical Report*.

Kautz, H. A. (1987). *A Formal Theory of Plan Recognition*. Ph.D. thesis, Bell Laboratories.

Keren, S., Gal, A., & Karpas, E. (2014). Goal recognition design. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2014)*.

Keren, S., Gal, A., & Karpas, E. (2015). Goal recognition design for non-optimal agents. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2015)*.

Keren, S., Gal, A., & Karpas, E. (2016a). Goal recognition design with non-observable actions. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2016)*.

Keren, S., Gal, A., & Karpas, E. (2016b). Privacy preserving plans in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016)*.

Keren, S., Gal, A., & Karpas, E. (2018). Strong stubborn sets for efficient goal recognition design. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2018)*.

Keren, S., Pineda, L., Gal, A., Karpas, E., & Zilberstein, S. (2017). Equi-reward utility maximizing design in stochastic environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2017)*.

Lesh, N., & Etzioni, O. (1995). A sound and fast goal recognizer. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1995)*.

Levine, S. J., & Williams, B. C. (2014). Concurrent plan recognition and execution for human-robot teams. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2014)*.

MacNally, A., Lipovetzky, N., Ramirez, M., & Pearce, A. (2018). Action selection for transparent planning. In *Proceedings of the Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*.

Masters, P., & Sardina, S. (2017). Cost-based goal recognition for path-planning. In *Proceedings of the Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2017)*.

Mirsky, R., Stern, R., Gal, Y. K., & Kalech, M. (2018). Goal and plan recognition design for plan libraries. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

Pattison, D., & Long, D. (2010). Domain independent goal recognition. In *Proceedings of the Fifth Starting AI Researchers Symposium (Stairs 2010)*.

Pattison, D., & Long, D. (2011). Accurately determining intermediate and terminal plan states using Bayesian goal recognition. In *Proceedings of the First Workshop on Goal, Activity and Plan Recognition(GAPRec 2011)*.

Pereira, R. F., Oren, N., & Meneguzzi, F. (2017). Landmark-based heuristics for goal recognition. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2017)*.

Ramirez, M., & Geffner, H. (2009). Plan recognition as planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2009)*.

Ramirez, M., & Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2010)*.

Ramirez, M., & Geffner, H. (2011). Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2011)*.

Ramirez, M., & Geffner, H. (2016). Heuristics for planning, plan recognition and parsing. In *arXiv preprint arXiv:1605.05807*.

Shoham, Y., & Tennenholtz, M. (1995). On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, *73*, 231–252.

Sohrabi, S., Riabov, A. V., & Udrea, O. (2016). Plan recognition as planning revisited. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016)*.

Son, T. C., Sabuncu, O., Schulz-Hanke, C., Schaub, T., & Yeoh, W. (2016). Solving goal recognition design using ASP. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2016)*.

Sukthankar, G., Geib, C., Bui, H. H., Pynadath, D., & Goldman, R. P. (2014). *Plan, Activity, and Intent Recognition: Theory and Practice*. Newnes.

Valmari, A. (1989). Stubborn sets for reduced state space generation. In *Proceedings of the International Conference on Application and Theory of Petri Nets (PETRI NETS 1989)*.

Vered, M., & Kaminka, G. A. (2017). Heuristic online goal recognition in continuous domains. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016)*.

Wayllace, C., Hou, P., & Yeoh, W. (2017). New metrics and algorithms for stochastic goal recognition design problems. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2017)*.

Wayllace, C., Hou, P., Yeoh, W., & Son, T. C. (2016). Goal recognition design with stochastic agent action outcomes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016)*.

Wayllace, C., Keren, S., Yeoh, W., Gal, A., & Karpas, E. (2018). Accounting for partial observability in stochastic goal recognition design: Messing with the marauders map. In *Proceedings of the Workshop on Heuristic Search in Domain-independent Planning (HSDIP-ICAPS 2018)*.

Wehrle, M., & Helmert, M. (2014). Efficient stubborn sets: Generalized algorithms and selection strategies. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2014)*.

Yolanda, E., R-Moreno, M. D., Smith, D. E., et al. (2015). A fast goal recognition technique based on interaction estimates. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2015)*.

Zhang, H., Chen, Y., & Parkes, D. C. (2009). A general approach to environment design with one agent. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2009)*.

Zhang, H., & Parkes, D. C. (2008). Value-based policy teaching with active indirect elicitation. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2008)*.