# Recognizing the Goals of Uninspectable Agents

**Irina Rabkina** IRABKINA@U.NORTHWESTERN.EDU
Northwestern University, Evanston, IL 60208 USA

**Pavan Kantharaju** PK398@DREXEL.EDU
Drexel University, Philadelphia, PA 19104 USA

**Mark Roberts** MARK.ROBERTS@NRL.NAVY.MIL
Naval Research Laboratory, Washington, DC 20375 USA

**Jason Wilson** JRW@NORTHWESTERN.EDU
**Kenneth Forbus** FORBUS@NORTHWESTERN.EDU
Northwestern University, Evanston, IL 60208 USA

**Laura M. Hiatt** LAURA.HIATT@NRL.NAVY.MIL
Naval Research Laboratory, Washington, DC 20375 USA

## Abstract

Effective interaction between agents requires reasoning about other agents' internal states. In some situations, such as in the case of multiagent systems with a shared policy, agents may have full knowledge of each other's knowledge, preferences, and goals. When interacting with humans or independent artificial agents, however, such direct inspection is not available. Instead, agents must model others' internal states through observation. In humans, such reasoning is called theory of mind (ToM). It has been argued that ToM reasoning can improve performance for artificial agents in team scenarios, as well. Here, we compare the performance of a model of ToM with that of a state-of-the-art goal recognition system on goal recognition tasks of increasingly uninspectable agents. We show that ToM reasoning is beneficial for agents when inspection is unavailable.

## 1. Introduction

Successfully collaborating with other agents requires knowing their objective(s). Sometimes, this information is readily available, such as when multiagent systems share a policy (e.g., Velagapudi et al., 2007), or when agents are capable of communication (e.g., Morgan & Pollack, 1990). However, when communication about internal states is imperfect or unavailable, agents must infer their compatriots' objectives by observing external actions. This task is referred to, among other terms, as goal recognition (E-Martin, R-Moreno & Smith, 2015). The motivation of our work is creating an online software assistant that recognizes the goal(s) of another agent and recommends actions or provides information to assist the agent in completing its goals more effectively.

While other goal recognition systems exist, state-of-the-art goal recognition systems make strong assumptions about the kind of information that is available during goal recognition. They typically receive an observation trace of an agent's activities as a sequence of action-state pairs, including the action's parameters, and reconcile these actions with a set of known or learned possible plans to infer the plan that the agent is performing, and thereby its top-level goal (Ramírez & Geffner, 2009). Alternatively, hierarchical plan recognition (Geib & Goldman, 2011; Holler et al., 2018) reconciles the observation trace using decomposition methods that aggregate the primitive actions into high-level tasks.

Since these recognition approaches access the same information about the observed agent's actions that the agent receives (i.e., the recognition algorithm observes the action-state pairs sent to the agent, including all parameters), the observation trace actually contains information about the internal state of the observed agent that cannot be gleaned from external observations alone. This type of internal information is not available when the agent is, for example, a human. Instead, observations of humans and other unknown agents consist only of external observation information, which is both noisy (i.e., imperfect) and incomplete (i.e., lacking internal information, such as action parameters).

Although prior research has examined the impact of noisy observations on goal recognition (Sohrabi et al., 2016; Vattam & Aha, 2015), few have examined the benefit of internal knowledge for recognition vs. the less informative value provided by external information. We view this as a major limitation of prior work, given that such tight and synchronous communication cannot always be assumed for multi-agent teams, especially those involving humans.

Theory of Mind (ToM) reasoning uses only external information to infer the mental (i.e., internal) states of another agent, including its goals, and has been studied extensively in humans (e.g., Premack & Woodruff, 1978). Computational models of ToM (e.g., Baker et al., 2011; Hiatt & Trafton, 2010; Rabkina et al., 2017) have shown promise in modeling human judgments, but have not yet been applied to complex goal recognition tasks.

To address this gap, we examine the extent to which incorporating internal knowledge, in addition to external knowledge, impacts goal recognition for two models: (1) a computational ToM model called Analogical ToM (AToM) by Rabkina et al. (2017) and (2) the state-of-the-art goal recognition system PANDA-REC by Holler et al. (2018). We demonstrate AToM's goal recognition capabilities on a widely accepted ToM task called stag-hunt (Skyrms, 2004). We then extend to a series of more complex tasks in the open-world domain of Minecraft, at three levels of information: (1) an observation trace with full internal information, directly from the observed agent's planner; (2) an observation trace with partial internal information, from the agent's execution of its plans; and (3) an observation trace with external knowledge only, adapted from the agent's plan execution. When both systems have perfect internal knowledge, AToM is slightly worse than PANDA-REC at recognizing an agent's goals. However, as knowledge is reduced, PANDA-REC performance drops while AToM maintains accuracy.

The contributions of this paper include: (1) establishing AToM as a validated model for goal recognition by demonstrating that it performs comparably to a Bayesian model of ToM when recognizing goals on a standard AI ToM task (2) developing a new benchmark for goal recognition tasks based on Minecraft; (3) demonstrating that removing internal knowledge causes challenges

for goal recognition approaches that do not have ToM capabilities, but not AToM, which continues to perform near ceiling.

## 2. Analogical Theory of Mind

The Analogical Theory of Mind (AToM; Rabkina et al., 2017) is a computational cognitive model of ToM reasoning and development. Its main claim is that ToM reasoning occurs via analogical processes, which have been well-established as a central component of human higher order cognition (see Gentner & Maravilla, 2018). AToM has successfully modeled children's improvement on ToM reasoning from hearing structured stories (Rabkina et al., 2017) and from learning a new grammatical construction (Rabkina et al., 2018). Crucially, the model architecture was not modified between these experiments—only the training data (i.e., the examples used to teach the children) differed. Because AToM has shown generality in modeling human ToM, here we propose using it for reasoning about artificial agents.

At a high level, AToM learns about mental states, such as goals or beliefs, through observation. It uses analogical generalization to build models from observed scenarios, which are applied during reasoning through analogical retrieval and inference. More specifically, during training, AToM takes in observations in the form of structured predicate calculus cases one at a time and compares each to its memory contents via analogical retrieval (Forbus et al., 1995). If a substantially similar case[1] is retrieved, a generalization (McLure et al., 2015) is formed. Otherwise, the case is added to memory as an individual example. During testing, AToM again takes in observations as predicate calculus cases and retrieves the most structurally similar learned case or generalization. The retrieved case is used for further reasoning, such as for answering questions or making predictions. Implementation details can be found in section 5.2 below.

In the next section, we demonstrate AToM's performance in the stag-hunt game (Skyrms, 2004), a ToM task that lies at the intersection of cognitive modeling of ToM reasoning and goal recognition. Then, we test AToM on a more complex goal recognition task.

## 3. AToM on Stag-hunt

Stag-hunt (Skyrms, 2004) is a prisoner's dilemma-style game that has recently been used to test ToM models' ability to recognize cooperation between agents (e.g., Shum et al., 2019; Xiong et al., 2018). During gameplay, players can choose to pursue a high reward (i.e., a stag) cooperatively or a low reward (i.e., a hare) individually. ToM models are then tasked with recognizing whether other agents intend to cooperate. Two formulations of this task exist: (1) the ToM model is a player in the game (e.g., Microsoft's Malmo Collaborative AI Challenge[2]) and (2) the ToM model is an observer, making judgments about other players (e.g., Shum et al., 2019). For direct comparison with a different computational cognitive model of ToM (i.e., Bayesian ToM, BToM; Shum et al., 2019), we take the latter formulation.

The stag-hunt task is similar to goal recognition, to the extent that observed agents have an underlying goal to cooperate (or not). However, the observations typically consist only of movements on a small grid, rather than more complex actions. Furthermore, the goals in typical goal recognition tasks are more complex.

---

[1] A combination of analogical similarity and feedback (either from the experimenter or via traditional supervised learning) is used to determine sufficiency.

[2] https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/

### 3.1 Stag-hunt Task Description

We use the stag-hunt dataset described by Shum et al. (2019). Recall that the goal of this task is to recognize cooperation between observed agents. Agents can cooperate to catch a high-value target (i.e., a stag) or work individually to catch a low-value target (i.e., a hare).

The dataset from Shum et al. (2019) consists of nine examples of the stag-hunt game, each on a partially traversable 7x5 grid map (Figure 1). Each example contains three hunters, two stags, and two hares. Stags can be captured via cooperation by two or three hunters for a high number of points; hares may be captured by a single hunter for a lower number of points. At each timestep, each hunter can move one square up, down, left, or right. Stags can also move one square to escape capture. Three timesteps are simulated per example. Predictions about cooperation goals are made after each timestep.

### 3.2 Stag-hunt Experiment and Results

AToM's accuracy in recognizing intended cooperation between agents in the stag-hunt game is shown in in Figure 2[3]. Cooperation predictions, for each pair of hunters, were made at the end of each timestep. We also report the accuracy of BToM and humans (both from Shum et al., 2019) for comparison. BToM made probabilistic inferences over a model of Composable Team Hierarchies (see Shum et al., 2019), while AToM learned to identify cooperation from observations (see Rabkina & Forbus, 2019). Specifically, AToM was trained and tested using structured representations of the stag-hunt scenes, using leave-one-out cross validation.

Note that at all timesteps, the two models and humans have no statistically significant differences (all $p>0.05$). This suggests that both AToM and BToM successfully model human judgments on this task and are competitive with each other in terms of accuracy. Thus, we have demonstrated that AToM can perform goal recognition in a ToM-specific domain. Next, we expand to a more complex goal recognition task.

## 4. Agent Simulation in Minecraft

In addition to the stag-hunt task, we define a problem space in the open-world computer game Minecraft[4]. Minecraft presents a challenge for both AI planning and goal recognition because the

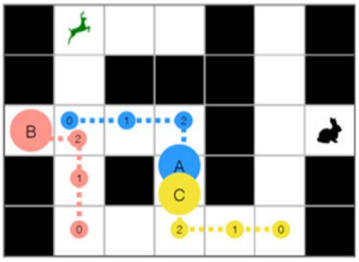


*Figure 1*. An example stag-hunt scenario. Agents A and C have cooperated to capture a stag, while agent B has acted alone to capture a hare. Figure adapted from Shum et al. (2019).

[3] Rabkina & Forbus (2019) provide a full discussion of the stag-hunt experiments.

[4] See Roberts et al. (2016) for a description of the game and the supporting framework we leverage, and Johnson et al. (2016) for information on Minecraft's Malmo platform for AI experimentation.

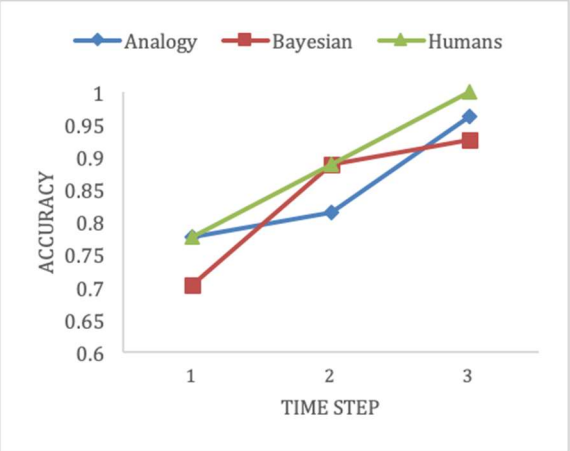


*Figure 2*. A comparison of AToM, BToM, and human accuracy on cooperation recognition in stag-hunt, per time step. Figure adapted from Rabkina & Forbus (2019).

set of possible plans to generate or recognize is open ended. In our task, an agent, Alex, is placed in a flat Minecraft world with a small farm in the middle and items randomly distributed around the perimeter. These include crop seeds, bone meal, chickens, cows, buckets of milk, eggs, and sugar. These items can be used to *craft* (i.e., make or obtain) food items, which provide Alex with varying numbers of food points. After a period of exploring, Alex chooses a goal to craft one food item with the highest possible food points, given the items it has observed. The *Top-Level Tasks* column of Table 1 is the set of possible goals that Alex can accomplish. Point values from Minecraft's internal food points system, which are used to weight goals, are also shown.

Many of Minecraft's crafting tasks have natural hierarchical structures. For example, crafting bread requires three wheat, and wheat is grown and harvested using wheat seeds. Growth can additionally be sped up using an item called bone meal. Due to these natural hierarchies, we define Alex's planning process using Hierarchical Task Networks.

### 4.1 Hierarchical Task Networks

Hierarchical Task Networks (HTNs; Erol, Hendler, & Nau, 1994) define a hierarchical planning framework that describes how to decompose complex tasks into simpler tasks until a sequence of actions that is executable in a given domain is found. Specifically, HTNs are made up of *complex* and *primitive tasks.* Both types of tasks are defined as first order terms with objects and variables from the domain as parameters. For example, *GrowAndHarvest(potato)* in Figure 3 is a task with parameter *potato*.

Complex tasks are activities that must be refined in order to be executed, while primitive tasks are basic objectives. We denote the set of complex tasks as $C$ and primitive tasks as $A$. Figure 3 provides an example of a decomposition of the complex task (in purple, italicized) *ObtainPotato* into primitive tasks (in blue, bold) for Minecraft. *ObtainPotato* is refined into *GrowAndHarvest*, which is further decomposed into *GrowWithBoneMeal*, ***Harvest***, and ***Gather***. Finally, *GrowWithBoneMeal* is decomposed into a sequence of six primitive tasks.

Complex tasks are decomposed with a set of *methods* $M$. A method is defined as ($name, c$, *prec, tn*), where $name$ is the name of the method, $c \in C$ is a complex task, *prec* is a set of preconditions, and *tn* is a *task network*. Task networks are defined as $(T, \alpha, \prec)$, where $T$ is a set of task identifiers,

*Table 1*. Minecraft Model for Planning with SHOP2. In the model definition, Top-Level and Helper Tasks make up the set of complex tasks $C$, while $A$ is the set of Primitive Tasks.

| ***Top-Level Tasks (Food Point Values)*** | ***Helper Task Categories*** | ***Primitive Task Categories*** |
|---|---|---|
| Obtain Chicken (2) | Crafting Items | Movement |
| Obtain Beef (3) | Gathering Items | Look |
| Obtain Pumpkin Pie (8) | Growing Crops | Item Selection |
| Obtain Cake (14) | Using Inventory Item | Item Crafting |
| Obtain Carrot (3) | | Item Gathering |
| Obtain Potato (1) | | |
| Obtain Bread (5) | | |

$\alpha : T \rightarrow C \cup A$ is a function that converts a task identifier into a task name, and $\prec \subseteq T \times T$ defines a partial ordering over tasks in $T$. Below are examples of the primitive task ***Gather*** and the method *m-GrowAndHarvest* that aligns with the decomposition in Figure 3.

Primitive Task ***Gather***(*?crop*)
  :preconditions ()
  :add ( `(inInventory ?crop)` )
  :delete ()

Method m-*GrowAndHarvest*(*?loc*)
  :task (*GrowAndHarvest)*
  :preconditions ()
  :task-network ({$t_1$, $t_2$, $t_3$},
    {$t_1$→*GrowWithBonemeal,*
    $t_2$→***Harvest***,
    $t_3$→***Gather***},
    {$t_1 > t_2 > t_3$})

Variables ?crop and ?loc represent some crop to harvest and a location in the Minecraft world. The method *m-GrowAndHarvest* decomposes the complex task *GrowAndHarvest* into the complex task *GrowWithBoneMeal* and primitive tasks ***Harvest*** and ***Gather***. Once the primitive tasks for accomplishing *GrowWithBoneMeal* and ***Harvest*** are executed and transition the state of the world, ***Gather*** further transitions the state by adding the crop to inventory.

Primitive tasks (i.e., actions) effect the state of the world, they are augmented with the tuple (*prec, add, delete*). A state is a set of first order predicates (Figure 4, Top). *Prec* is a set of preconditions that must be met in the state for the task to be applicable. *Add* and *delete* are sets of predicates that are added and deleted from the state, respectively, during task execution.

An HTN planning problem is a tuple $(D, s_0, tn_I)$, where $D = (A, C, M)$ is an HTN planning model, $s_0$ is an initial state, and $tn_I$ is an initial task network. In this work, we assume that a single task $X$ is being pursued at a given time. Therefore, $tn_I = (\{t\},\ \{t \rightarrow X\},\ \emptyset)$. A solution to the HTN planning problem is a plan $t_1 \ldots t_n$ extracted from a task network that satisfies $tn_I$. Figure 4 (top) provides an example of an HTN planning problem and its corresponding solution for the task *ObtainPotato* (bottom).

### 4.2 HTN Planning and Execution in Minecraft

We now describe the Minecraft HTN planning model and the application of HTN planning to Minecraft. Table 1 summarizes the task categories. Complex tasks are either top-level tasks or

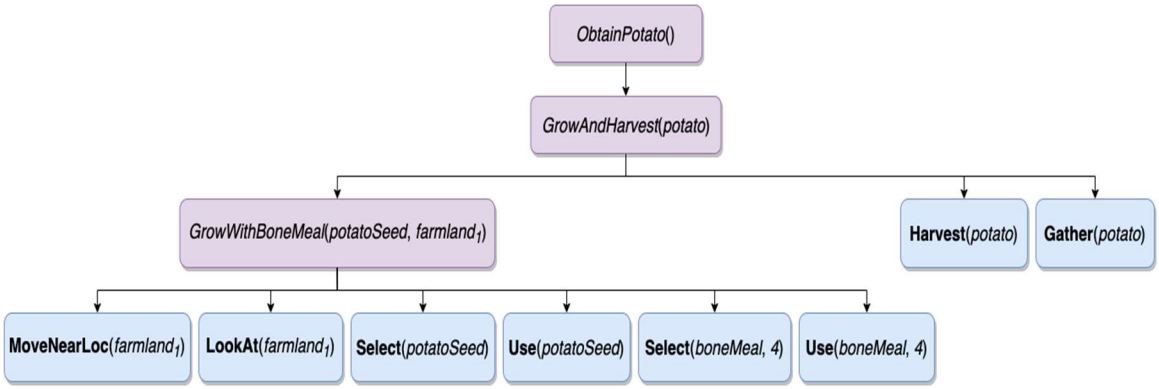


*Figure 3.* Decomposition of the *ObtainPotato* task. Complex tasks are purple, primitive tasks are blue.

helper tasks. Top-level tasks are objectives that the agent may choose to pursue (such as making pumpkin pie and cake). Helper tasks complete top-level tasks and include: crafting items, gathering items, growing crops, and inventory usage. Primitive tasks include: movement, looking, item selection, item crafting, and item gathering.

We use the HTN planner JSHOP2 (Nau et al., 2003; Ilghami & Nau, 2003) along with the above model to generate plans for an agent to execute in the Minecraft environment. The Minecraft states used by SHOP2 contain information such as the inventory of the agent, the entities and locations it has observed, and information about the agent itself, such as its current location. As described above, these facts are modified by the primitive tasks performed by the agent. For example, a gathering primitive task will add items to the agent's inventory. However, these tasks only change the SHOP2 state; they do not directly change the Minecraft game.

To perform these primitive tasks directly in the Minecraft game, plans generated by the SHOP2 planner are used by an agent to construct executable plans in Minecraft. This executable plan is then run in the Minecraft environment to completion. If items required for crafting-related tasks are observed in the environment, but are not in the agent's inventory, the agent constructs a plan to retrieve them before attempting the task. Once all items have been retrieved, the agent then replans to get a new plan for crafting the item. We only replan after all items are retrieved to prevent the agent from constantly replanning and not completing any objectives. Replanning makes sense here because the agent may observe items for more important objectives while retrieving items for crafting. In this case, the agent should execute the more important objective.

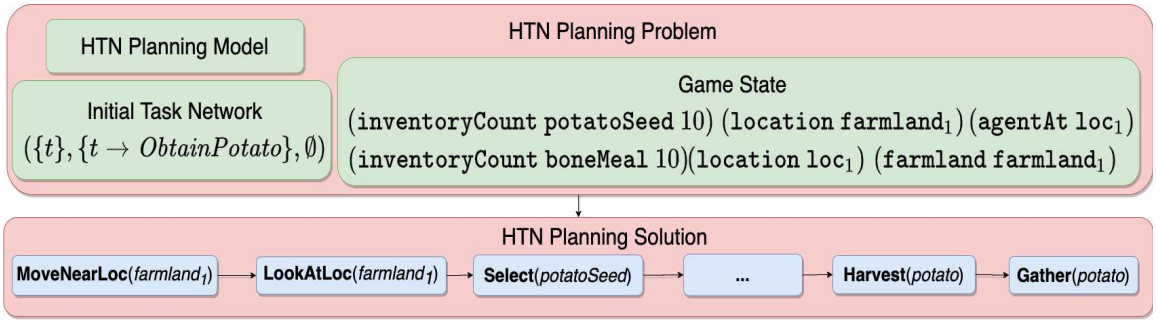


*Figure 4.* Example of Minecraft SHOP2 planning problem (Top) and solution (Bottom).

## 5. Goal Recognition

We next turn to describing how a different agent might recognize Alex's goal in order to infer which food item Alex is working to craft. We formally define the goal recognition (GR) problem as $(D_{GR}, s_0, \vec{o}, G)$, where $D_{GR}$ is a model for the GR problem, $s_0$ is some initial state of the world, $\vec{o}$ is a sequence of observed actions, and $G$ is a set of goals to recognize. A solution GR is a goal $g \in G$ being pursued via the execution of $\vec{o}$. With respect to our Minecraft domain, $G$ is the set of Top-Level Tasks in Table 1 and $g$ is one of those tasks, such as *ObtainChicken*.

We compare the goal recognition accuracy of a model of human ToM reasoning (AToM; Rabkina et al., 2017) with the performance of a goal recognition system (Holler et al., 2018) to show the strengths and weaknesses of each when reasoning about other agents. We describe these two systems next.

### 5.1 Goal Recognition as Planning

Several prior approaches have viewed the problem of GR as a planning task, where techniques from classical planning (Ramirez & Geffner, 2009; Ramirez & Gefner, 2010) and HTN planning (Holler et al., 2018) have been used to solve GR. At a high level, this is done by converting the GR problem into a planning problem and solving it with a planner. A solution to the planning problem is a solution to the GR problem.

Using planning for GR also allows us to leverage state-of-the-art planning techniques. Our study utilizes the Planning and Acting in a Network Decomposition Architecture (PANDA) planning algorithm[5] for GR. PANDA is a hybrid planning algorithm that combines HTN planning concepts with partial-order causal link planning. We use PANDA for GR over SHOP2 in Minecraft. PANDA has been previously used for GR (Holler et al., 2018) and the code for it was readily available. We refer to goal recognition using PANDA as PANDA-REC.

Figure 5 provides a diagram of PANDA-REC. PANDA-REC takes a GR problem $(\mathrm{D_{GR}}, \mathrm{s_0}, \vec{\mathrm{o}}, \mathrm{G})$, where $D_{GR}$ is an HTN model and transforms the problem into an HTN planning problem $(D, s_0, tn_I)$. Here, $D$ is a modification of the HTN model $D_{GR}$ that contains new methods and tasks that are pertinent to $\vec{o}$ and G, and $tn_I$ is an initial task network. This converted problem is then passed into PANDA and the recognized goal $g \in \mathrm{G}$ is extracted from its output.

The problem transformation process works as follows. First, a new complex task $t_I$ is constructed and added to the initial task network $tn_I$. This new task is similar to the *TopLevelAct* in Kautz and Allen (1986) and represents an abstract task that is more general than any top-level task in Minecraft. Next, a set of new methods are constructed that decompose $t_I$ into each of the goals $g \in \mathrm{G}$ .This implies that finding a plan for $tn_I$ would require at least one of the goals in G**,** therefore

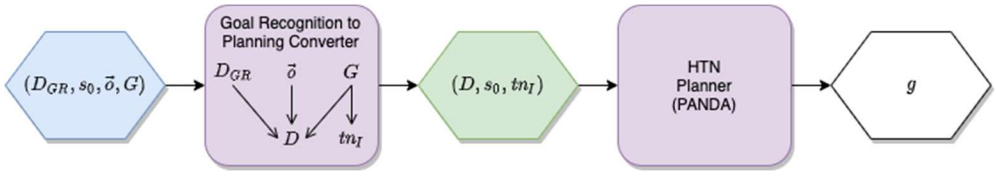


*Figure 5.* Diagram of goal recognition via planning PANDA-REC.

[5] https://www.uni-ulm.de/en/in/ki/research/software/panda

providing us the recognized goal(s). Finally, a new set of methods and tasks are added to $D$ to enforce the restriction that any HTN solution found for $tn_I$ must start with $\vec{o}$.

### 5.2 Goal Recognition as Theory of Mind

We next describe how AToM can be applied to the task of goal recognition. Recall that the central claim of AToM is that ToM occurs through analogical processes. It is implemented using the analogy models in the Companion cognitive architecture (Forbus & Hinrichs, 2017). AToM learns a through experience and does not require an HTN to recognize goals.

We treat goal recognition as a classification problem for AToM. Using the Sequential Analogical Generalization Engine (SAGE; McLure et al., 2015), a model (called a *generalization pool*) is learned for every potential goal type from previously observed traces. A trace can be of arbitrary form, including the output of the SHOP2 planner, a report of the agent's actual actions, or sensor-like observations of those actions. It is passed to AToM as a predicate calculus case.

During training, cases of different goals, $g \in G$, are passed to SAGE one at a time. The most similar previously observed case (if one exists) is retrieved via an analogical retrieval algorithm (MAC/FAC; Forbus et al., 1995). We refer readers to the original paper for specifics of the retrieval algorithm. Importantly, the retrieval algorithm computes a *structural similarity score* between the original case, $o$, and the retrieved case, $r$. At a high level, this score represents the amount and depth of overlapping structure between the two cases (see Forbus et al., 2016 for algorithm and implementation details). If the structural similarity score is above a preset threshold (the default value of 0.8 is used in the present work), the two cases are merged into a generalization, which is added to the model that corresponds to the current case's goal (see McLure et al., 2015).If the similarity score between $o$ and $r$ is not above the threshold needed to form a generalization, $o$ is added to the model as an individual example.

Generalizations contain frequentist probabilities of the facts contained in their underlying cases. For example, if a generalization derives from a case with the facts `{(movesTo cow123), (swingsAt cow123)}` and another with the facts `{(movesTo cow456), (throws cow456)}`, the generalization would contain the fact that a cow is being moved to with a probability of 1.0 and that it is being swung at and thrown each with probability 0.5. As more cases are merged with the generalization, the probabilities are updated. Eventually, facts with probabilities below a preset threshold (the default value of 0.2 is used in the present experiments) fall out of the generalization. Thus, a generalization can be treated as a schema for a given type of case.

The same similarity-based retrieval process is also used during testing. However, retrieval occurs across all learned models (i.e., all learned goals, $G$). The goal, $g$, corresponding to the model of the retrieved case, is returned.

## 6. Experiments and Results

The objective of our experiments is to compare different ways to infer an agent's goals given different types of observed sequences of actions, with varying degree of agent inspectability. To that end, we compare the performance of AToM with PANDA-REC on the Minecraft GR task with these different levels data: agent planner outputs (i.e., full internal knowledge), agent execution traces (i.e., partial internal knowledge), and external observation traces (i.e., external knowledge only). We also test whether AToM and PANDA-REC can transfer goal recognition from a partial internal knowledge model to external observations.

Two random baselines were also considered for each experiment. The first generated its interpretation of the agent's goal by sampling uniformly across goals that appear in the dataset. The second was biased, with each potential goal weighted by its prevalence in the dataset. All results are reported in Table 2. Where available, standard deviations are reported in parentheses (Note that because PANDA's model is pre-determined and deterministic, no standards of deviation are available. Similarly, AToM learns a single model in the Train Exec/Test Ext. condition, so no standard of deviation is computed.). The highest accuracy for each test is bolded.

For our experimental data, we constructed a dataset from Minecraft play session logs. A single play session corresponds to an agent being placed on a map and executing top-level tasks from Table 1 for a predefined amount of time (in this case, 180 seconds). A log, consisting of all planner output and executed actions, is generated for each play session. Planning and execution of multiple top-level tasks are included in each log. For the experiments described in this paper, we extracted a random subset of 100 plan traces (i.e., planner outputs) and 100 execution traces (i.e., sequences of executed actions) from 50 play session logs (10 pseudo-randomly generated maps, 5 times each.). Note that the execution trace dataset had one fewer possible goal, as `obtain_carrot` did not appear in the randomly extracted dataset.

### 6.1 How well do PANDA and AToM perform on plan traces (full internal information)?

The first experiment focuses on recognition of goals when the list of actions slated for execution by Alex is directly observable. We note that these plans are direct internal information about an agent, as plans are constructed by the agent internally, converted into actions that can be executed, and then executed in an environment. In this experiment, PANDA-REC was also provided the HTN model used by Alex to develop its plan while AToM learned from SHOP2 outputs via 10-fold cross-validation (i.e., 10 folds, each consisting of 90 plan traces for training and 10 plan traces for testing).

PANDA-REC was 100% accurate in recognizing goals based on the planned sequence of Alex's actions. This fit our intuition, as PANDA-REC is also given Alex's model of the domain and actions. AToM performed significantly worse, with 92% accuracy (one sample non-parametric median test; $p<.05$). Both systems performed substantially better than the uniform and biased baselines. These results can be found in Table 2, row 1 (*Plan Traces*).

### 6.2 How well do PANDA and AToM perform on execution traces (partial internal information)?

Planner traces provide perfect information about the plans being executed. However, the execution of a plan is rarely perfect, and may not correspond exactly to a plan. In this experiment, PANDA-REC and AToM were tested on goal recognition using the agent's report of its executed actions. We note that some of these actions (i.e., move and look at) contain internal information about the agent, such as the specific object that it is moving toward. PANDA-REC was also provided an HTN model corresponding to the executed actions, while AToM once again learned the model through training. As before, AToM was trained via 10-fold cross-validation.

PANDA-REC's accuracy dropped substantially when working from Alex's actions but remained above both alternate baselines. It performed at 63% accuracy. AToM's performance did not change significantly from Alex's planned sequence of action, maintaining 90% accuracy. A one sample non-parametric median test showed that AToM performed significantly better than PANDA-REC ($p <.05$) in this condition. These results can be found in Table 2, row 2 (*Execution Traces*).

*Table 2*. Results for Goal Recognition Experiments

| | *PANDA-REC* | *AToM* | *Uniform Baseline* | *Biased Baseline* |
|---|---|---|---|---|
| *Plan Traces* | **1.0** | 0.92 (0.075) | 0.14 | 0.226 |
| *Execution Traces* | 0.63 | **0.90** (0.077) | 0.167 | 0.237 |
| *Train Exec. / Test Ext.* | 0.30 | **0.90 ( --- )** | 0.167 | 0.237 |
| *Train & Test Ext.* | 0.63 | **0.88** (0.098) | 0.167 | 0.237 |

### 6.3 How sensitive are PANDA and AToM to external knowledge traces?

In many multi-agent scenarios, communication is limited or impossible. Instead, agents must reason based only on their own observations of compatriots' behavior without internal state. In this experiment, we removed information about the parameters of actions. Thus, traces consisted only of what could be observed externally (e.g., that the agent is moving in a certain direction) and lacked internal state (e.g., where the agent was specifically hoping to go to).

We tested PANDA-REC and AToM's sensitivity to external knowledge traces under two conditions: (1) with a model based on Alex's actual executed actions and (2) with a model based only on external observations of those actions. For (1), PANDA-REC was given the HTN model used in the previous experiment. AToM learned a model using the whole execution trace dataset. For (2), PANDA was given a modified version of the HTN model, which did not contain internal information (i.e., information that would not be available to an external observer—such as what object Alex is moving toward—was removed). As in previous experiments, AToM was trained using 10-fold cross-validation.

When tested on the external knowledge-only traces using the full HTN model of the agent's executed actions, PANDA-REC's performance dipped further to 30% accuracy (Table 2, row 3, *Train Exec/Test Ext*). This drop in accuracy was a result of recognition failing for several of the goals, particularly *ObtainChicken* and *ObtainBeef*. The methods for *ObtainChicken* and *ObtainBeef* in the HTN model were too specific as they require moving and looking at specific types of entities (i.e., chicken or cow). However, the move and look actions in the external knowledge traces were applied to general locations, as an external observer would not know which entity, if any, the agent was moving toward. Thus, recognition failed because the HTN model was not general enough to handle less information. However, when tested using the modified model, which was more general, it performed as well as it had when trained and tested on execution traces (i.e., 63% accuracy; Table 2, row 3, *Train Ext/Test Ext*). AToM performed equivalently across tasks: 90% accuracy when trained on execution traces and 88% accuracy when trained on external knowledge-only traces. This was significantly better than PANDA-REC ($p<0.05$; single sample non-parametric t-test) in both conditions.

## 7. Discussion and Future Work

For these Minecraft recognition tests, AToM outperformed PANDA-REC on goal recognition conditions when given partial internal information or external information only. This is a hallmark of human ToM reasoning, which AToM models. Thus, our results suggest that ToM reasoning via AToM can help agents reason about others.

The chief claim of AToM as a cognitive model is that ToM reasoning and development occur via analogical processes. Here, those same processes allow AToM to robustly reason about the internal states of agents, without direct knowledge of those states. Specifically, analogy allows AToM to make inferences based on its previous observations. For example, if it has learned that agents walk up to cows before slaughtering them (e.g., from agent action traces), it can infer that the object the agent was walking toward before slaughtering it (e.g., in an anonymized agent action trace) was also a cow. Furthermore, analogy's focus on structure makes retrieval with complete object uncertainty possible. That is, if all objects were removed from a trace, AToM would guess that throwing something at the ground and later harvesting something else is a planting task—perhaps mistaking ObtainPotato for ObtainCarrot, but not ObtainBeef. It remains to be seen whether other ToM models can do similar reasoning.

From a practical standpoint, one disadvantage of AToM, as compared to PANDA-REC, is its need to be trained—Recall that AToM was trained via 10-fold cross validation on datasets of 100 total traces in the present experiments. When recognizing from planner output, PANDA-REC was able to use the planner. While the model did need to be modified further for the other conditions, training data was never necessary. On the other hand, PANDA-REC has the disadvantage of requiring a hand-crafted model.

Interestingly, the generalizations learned by AToM were often similar to the individual plans in PANDA-REC's model. This suggests that the models used by PANDA-REC, when converted to cases of a format similar to observation trace outputs, may be sufficient to populate AToM's case library. That is, explicit training may not be necessary. Alternatively, the AToM model might provide insights into learning, rather than hand-crafting, the PANDA model. We will explore these possibilities in future work.

More generally, we would like to give agents the ability to not only recognize compatriots' goals, but also to change their own behavior accordingly. This requires online goal recognition that is accurate while reasoning from partial data (i.e., before the compatriot finishes its task). PANDA-REC can be configured to make a recognition decision prior to seeing a complete plan trace (Holler et al., 2018). However, the computations for this can become too slow for online recognition. On the other hand, analogical retrieval allows AToM to be relatively fast. It remains to be seen whether AToM can maintain accuracy with partial traces. It is likely that other components of ToM reasoning (e.g., about knowledge and desire states) will need to be integrated to increase robustness of AToM's predictions from partial traces. We will explore applications of PANDA-REC and AToM to online goal recognition in future work.

## 8. Related Work

Goal Recognition is the problem of inferring the top-level goal of a partial plan executed by an agent (E-Martin, R-Moreno, & Smith, 2015) and has been extensively applied to games. For example, Gold (2010) uses an Input-Output Hidden Markov Models (Bengio & Fransconi, 1994) to recognize player goals from low-level actions in a top-down action adventure game. Ha et al. (2011) uses a Markov Logic Network (Richardson & Domingos, 2006) to recognize goals in the

educational game Crystal Island. Min et al. (2014) and Min et al. (2016) use deep learning techniques (i.e., stacked denoising autoencoders, Vincent et al., 2010; and Long Short-Term Memory, Hochreiter and Schmidhuber 1997) to also recognize goals in Crystal Island. In contrast, we apply goal recognition to Minecraft. Goals in Crystal Island are tied to the narrative. However, Minecraft does not have a narrative and has an undefined number of possible goals.

Plan recognition (Schmidt, Sridharan, & Goodson, 1978), the sibling problem to goal recognition, entails finding the set of plans and goals an agent is believed to be pursuing given some observed sequence of actions. One way to view plan recognition is presented in the seminal theoretical work by Kautz and Allen (1986). In particular, they viewed plan recognition as a form of McCarthy's circumscription (1980) and represented the plan library in the form of a plan hierarchy/graph. Other work viewed plan recognition as a form of parsing using a formal grammar that defines a set of possible plans that can be executed by an agent. Such grammars include Context-Free Grammars (CFGs; Villain, 1990), Probabilistic Context-Free Grammars (Pynadath & Wellman, 2000), plan tree grammars (Geib & Goldman, 2009), Plan Frontier Fragment Grammars (Geib, Maraist, & Goldman, 2008; Geib & Goldman, 2010), and Combinatory Categorial Grammars (Geib, 2009; Geib & Goldman, 2011). There has also been work on using case-based reasoning for plan recognition, where the plan library is a case base (Cox & Kerkez, 2006; Fagan & Cunningham, 2003).

Other techniques viewed plan recognition as planning. To the best of our knowledge, the first work to do this was by Ramirez and Geffner (2009). Specifically, this approach used off-the-shelf classical planners to solve the plan recognition problem. The main advantage of this approach is that it only requires a model of the domain's actions. Other works that follow this view include Ramirez and Geffner (2010), Ramirez & Geffner (2011), and Sohrabi, Riabov, and Udrea (2016).

Our work focuses on applying the work by Holler et al., (2018) to goal recognition in Minecraft. Their work outlines a technique that uses off-the-shelf Hierarchical Task Network (HTN; Erol, Hendler & Nao, 1994) planning to recognize plans and goals. Unlike prior plan recognition as planning approaches, this does require a plan library.

## 9. Conclusion

Our findings suggest that ToM capabilities are key to robust goal recognition as availability of internal information changes. While there are many different approaches to plan and goal recognition in the literature, this work aimed to provide an initial study of AToM—a computation model of ToM reasoning—on goal recognition tasks by comparison to one state-of-the-art system. We found that, while the state-of-the-art goal recognition system (Holler et al., 2018) performs at 100% accuracy when outputs from the observed agent's planner (i.e., perfect internal information) are available, its performance decreases significantly when only agent actions or observations (i.e., external information) are available. On the other hand, the system that models human theory of mind reasoning (Rabkina et al., 2017), maintains accuracy at approximately 90% as availability of internal information changes. These findings suggest that incorporating theory of mind when reasoning about other agents' internal states can lead to better understanding of others' actions, which may lead to better interactions between agents.

## Acknowledgements

We would like to thank anonymous reviewers of the Plan, Activity, and Intent Recognition workshop for their feedback on an earlier version of this paper. This work was supported in part by

the Air Force Office of Scientific Research to KF and IR, and the Office of Naval Research to LH and MR. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the US Navy.

## References

Baker, C., Saxe, R., & Tenenbaum, J. (2011). Bayesian Theory of Mind: Modeling Joint Belief-desire Attribution. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Bengio, Y., & Frasconi, P. (1995). An input output HMM architecture. In *Advances in neural information processing systems,* 427-434.

Bercher, P., Keen, S., & Biundo, S. (2014). Hybrid planning heuristics based on task decomposition graphs. In *Seventh Annual Symposium on Combinatorial Search*, 35-43.

Cox, M. T., & Kerkez, B. (2006). Case-based plan recognition with novel input. In *International Journal of Control and Intelligent Systems*, 34(2), 96-104.

E-Martin, Y., R-Moreno, M. D., & Smith, D. E. (2015) A fast goal recognition technique based on interaction estimates. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 761-768.

Erol, K., Hendler, J., & Nau, D. S. (1994). HTN planning: Complexity and expressivity. In *Proceedings of the 8th AAAI Conference on Artificial Intelligence*, 1123-1128.

Fagan, M., & Cunningham, P. (2003). Case-based plan recognition in computer games. In *International Conference on Case-Based Reasoning*, 161-170.

Forbus, K. D., Ferguson, R. W., Lovett, A., and Gentner, D. (2016). Extending SME to Handle Large-scale Cognitive Modeling. *Cognitive Science*, 1-50.

Forbus, K., Gentner, D., and Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19, 141-205.

Forbus, K. D., & Hinrich, T. (2017). Analogy and relational representations in the companion cognitive architecture. *AI Magazine*, *38*(4), 34-42.

Geib, C. W., & Goldman, R. P. (2005). Partial observability and probabilistic plan/goal recognition. In *Proceedings of the International Workshop on Modeling Other Agents from Observations,* 1-6.

Geib, C. W., & Goldman, R. P. (2009). A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, *173*(11), 1101-1132.

Geib, C., & Goldman, R. (2010). Handling looping and optional actions in YAPPR. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 17-22.

Geib, C., & Goldman, R. (2011). Recognizing plans with loops represented in a lexicalized grammar. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 91-98.

Geib, C. W., Maraist, J., & Goldman, R. P. (2008). A New Probabilistic Plan Recognition Algorithm Based on String Rewriting. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling*, 91–98.

Geib, C. (2009). Delaying commitment in plan recognition using combinatory categorial grammars. In *Twenty-First International Joint Conference on Artificial Intelligence*, 1702-1707.

Gentner, D. & Maravilla, F. (2018). Analogical reasoning. L. J. Ball & V. A. Thompson (eds.) *International Handbook of Thinking & Reasoning* (pp. 186-203). NY, NY: Psychology Press.

Gold, K. (2010). Training goal recognition online from low-level inputs in an action-adventure game. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*, 21-26.

Ha, E. Y., Rowe, J. P., Mott, B. W., & Lester, J. C. (2011). Goal recognition with Markov logic networks for player-adaptive games. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 32-39.

Hiatt, L. M., & Trafton, J. G. (2010). A Cognitive Model of Theory of Mind. In *Proceedings of the 10th International Conference on Cognitive Modeling,* 91-96. Philadelphia, PA: Drexel University.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Höller, D., Behnke, G., Bercher, P., & Biundo, S. (2018). Plan and goal recognition as HTN planning. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence*, 466-473. IEEE.

Johnson, M., Hofmann, K., Hutton, T., & Bignell, D. (2016). The Malmo Platform for Artificial Intelligence Experimentation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 4246-4247.

Kandaswamy, S., Forbus, K., & Gentner, D. (2014). Modeling learning via progressive alignment using interim generalizations. In *Proceedings of the Annual Meeting of the Cognitive Science Society* (Vol. 36, No. 36).Kautz, H. A., & Allen, J. F. (1986). Generalized Plan Recognition. In *Proceedings of the 5th AAAI Conference on Artificial Intelligence*, 32-37.

McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning. *Artificial intelligence*, *13*(1-2), 27-39.

McLure, M.D., Friedman S.E. and Forbus, K.D. (2015). Extending Analogical Generalization with Near-Misses. *In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas

Min, W., Ha, E. Y., Rowe, J., Mott, B., & Lester, J. (2014). Deep learning-based goal recognition in open-ended digital games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 37-43.

Min, W., Mott, B. W., Rowe, J. P., Liu, B., & Lester, J. C. (2016). Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory Networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2590-2596.

Morgan, P. R. C. J. L., & Pollack, M. E. (1990). *Intentions in communication*. MIT press.

Nau, D. S., Au, T. C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of artificial intelligence research*, *20*, 379-404.

Ilghami O., & Nau, D. S. (2003). *A General Approach to Synthesize Problem-Specific Planners* (Report # CS-TR-4597). Retrieved from University of Maryland website: https://www.umiacs.umd.edu/publications/general-approach-synthesize-problem-specific-planners

Premack, D., & Woodruff, G. (1978). Does the Chimpanzee have a Theory of Mind? *Behavioral and Brain Sciences*, *1*(4), 515-526.

Pynadath, D. V., & Wellman, M. P. (2000). Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 507-514. Morgan Kaufmann Publishers Inc.

Rabkina, I. & Forbus, K. D. (2019). Analogical Reasoning for Intent Recognition and Action Prediction in Multi-Agent Systems. In *Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems*. Cambridge, MA.

Rabkina, I., McFate, C. J., & Forbus, K. D. (2018). Bootstrapping from language in the Analogical Theory of Mind model. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society*.

Rabkina, I., McFate, C., Forbus, K. D., & Hoyos, C. (2017). Towards a Computational Analogical Theory of Mind. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society,* 2949-2954.

Ramírez, M., & Geffner, H. (2009). Plan recognition as planning. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Ramírez, M., & Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 1121-1126.

Ramírez, M. and Geffner, H. (2011). Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine learning*, *62*(1-2), 107-136.

Roberts, M., Shivashankar, V., Alford, R., Leece, M., Gupta, S., & Aha, D. W. (2016). Goal reasoning, planning, and acting with ActorSim, the actor simulator. In *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems.*

Schmidt, C. F., Sridharan, N. S., & Goodson, J. L. (1978). The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, *11*(1-2), 45-83.

Shum, M., Kleiman-Weiner, M., Littman, M. L., & Tenenbaum, J. B. (2019). Theory of Minds: Understanding Behavior in Groups Through Inverse Planning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

Skyrms, B. (2004). The Stag Hunt and the Evolution of Social Structure. Cambridge University Press.

Sohrabi, S., Riabov, A. V., & Udrea, O. (2016). Plan Recognition as Planning Revisited. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 3258–3264

Vattam, S. S., & Aha, D. W. (2015). Case-based plan recognition under imperfect observability. In *International Conference on Case-Based Reasoning*, 381-395. Springer, Cham.

Velagapudi, P., Prokopyev, O., Sycara, K., & Scerri, P. (2007). Maintaining shared belief in a large multiagent team. In *Proceedings of the 10th International Conference on Information Fusion*, 1-8. IEEE.

Vilain, M. B. (1990). Getting Serious About Parsing Plans: A Grammatical Analysis of Plan Recognition. In *Proceedings of the 8th AAAI Conference on Artificial Intelligence*, 190-197.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*, 3371-3408.

Xiong, Y., Chen, H., Zhao, M., & An, B. (2018). HogRider: Champion Agent of Microsoft Malmo Collaborative AI Challenge. In *Thirty-Second AAAI Conference on Artificial Intelligence*.