Recognizing the Goals of Uninspectable Agents

Irina Rabkina[1], Pavan Kathnaraju[2], Mark Roberts[3], Jason Wilson[4], Kenneth Forbus[5], Laura Hiatt[6]

Northwestern University[1,4,5], Drexel University[2], Naval Research Laboratory[3,6]

irabkina@u.northwestern.edu[1], pk398@drexel.edu[2], mark.roberts@nrl.navy.mil[3], jrw@northwestern.edu[4], forbus@northwester.edu[5], laura.hiatt@nrl.navy.mil[6]

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 the internal states of their compatriots through observations of their behaviors in the world. 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 (Analogical Theory of Mind; Rabkina et al., 2017) with that of a state-of-the-art goal recognition system (Holler et al., 2018) on goal recognition tasks of increasingly uninspectable agents. We show that ToM reasoning is beneficial for agents when direct access to the internal states of their compatriots is not available.

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 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 quickly.

While other goal recognition systems[1] 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

On the other hand, the use of only external information to infer the mental (i.e., internal) states of another agent, including its goals, has been studied extensively in humans (e.g., Premack & Woodruff, 1978) and is called Theory of

[1] We use the term goal recognition to refer to goal, plan, and task recognition systems, since all of the above are capable of recognizing the objective of an observed trace (i.e., the goal).

Mind (ToM) reasoning. 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 task recognition system PANDA by Holler et al. (2018). We perform an initial demonstration of AToM's ToM reasoning capabilities on a widely accepted ToM task, 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) observation trace with full internal information, directly from the observed agent's planner, (2) observation trace with partial internal information, from the agent's execution of its plans, and (3) 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 at recognizing an agent's goals. However, as knowledge is reduced, PANDA's performance drops while AToM degrades gracefully.

The contributions of this paper include: (1) demonstrating that AToM performs well at recognizing goals on a standard ToM task, thus establishing it as a fair baseline for state-of the-art ToM techniques; (2) developing a new benchmark for goal recognition tasks based on Minecraft; (3) demonstrating that removing internal knowledge causes a drop in goal recognition accuracy in PANDA, but not AToM, which continues to perform near ceiling.

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 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 computational cognitive model of ToM (i.e., Bayesian ToM, BToM; Shum et al., 2019), we take the latter formulation.

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

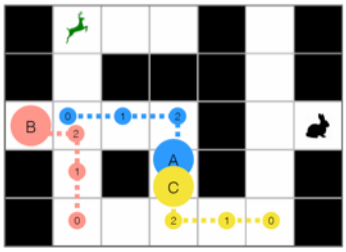

Figure 1. An example stag-hunt scenario. Agents A and C have cooperated to capture a stag, while agent B has acted alone to capture a hare. Figure adapted from Shum et al. (2019).

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. Furthermore, cooperation is assumed to be a shared goal between two agents.

By showing that AToM is as accurate as both humans and BToM on this task, we aim to show that it is competitive on tasks that are specifically designed to test ToM reasoning. A full discussion of this work can be found in Rabkina & Forbus, 2019.

Stag-hunt Task Description

For direct comparison, we use the stag-hunt dataset described by Shum et al. (2019). Recall that the goal of this formulation of the stag-hunt 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 must be captured by a single hunter for a lower number of points. Three timesteps are simulated per example. Cooperation predictions are made at the end of each timestep.

Stag-hunt Results

AToM's accuracy in recognizing intended cooperation between agents in the stag-hunt game can be found in Figure 2. We also report the accuracy of BToM and humans (both Shum et al., 2019) for comparison. BToM made decisions over a model of team hierarchies, while AToM learned to identify cooperation via leave-one-out cross-validation (see Rabkina & Forbus, 2019 and Approach, below).

Note that at all timesteps, the two models and humans do not differ (all $p>0.05$). This suggests that both AToM and

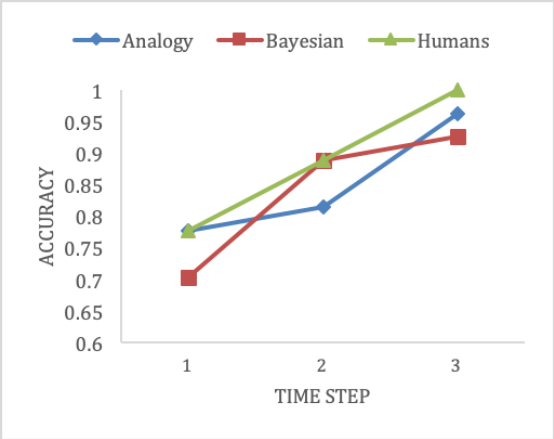

Figure 2. A comparison of AToM, BToM, and human accuracy on cooperation recognition in stag-hunt, per time step.

BToM successfully model human judgments on this task and are competitive with each other in terms of accuracy.

Agent Simulation in Minecraft

We define a problem space in the open-world game Minecraft (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). 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. After a period of exploring, Alex chooses a goal that will maximize its food points, given the items it has observed. We use Minecraft's internal food points[3] system for value calculations, shown in the left column of Table 1. The goal recognition systems must recognize which goal Alex is pursuing. We assume that it pursues one goal at a time (i.e., no interleaved goals).

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, the agent's behaviors are defined using Hierarchical Task Networks.

Hierarchical Task Networks

Hierarchical Task Networks (HTNs; Erol, Hendler, & Nau, 1994) provide a way to encode hierarchical plan knowledge. We define an *HTN planning model* as a tuple $D = (M, T)$, where M is a finite set of *methods* and T is a finite set of *primitive* and *compound tasks*. Primitive tasks are executable actions with preconditions and effects. Compound tasks are more abstract and are composed of compound or primitive tasks. Methods are used to decompose compound tasks into a sequence of other tasks.

An *HTN planning problem* $P = (D, s_0, \mathbb{T})$ is a tuple, where D is an HTN planning model, s_0 is an initial state, and $\mathbb{T}$ is a list of non-primitive tasks, where $T \subseteq \mathbb{T}$. A solution to P, $\pi = < a_0, a_1, \ldots, a_n >$, is a plan where each action a_i corresponds to a primitive task, π completes all tasks in $\mathbb{T}$, and there exist no compound tasks in $\mathbb{T}$. An HTN planner constructs this plan using the methods defined in the domain to decompose tasks in $\mathbb{T}$ into a sequence of actions. In this work $|\mathbb{T}| = 1$, since the agent pursues only one goal at a time.

HTN Planning and Execution in Minecraft

We used the HTN planner SHOP2 (Nau et al., 2003) to generate plans for an agent to execute in the Minecraft environment. Planning with SHOP2 requires two components of knowledge: *state model* and *HTN planning model*. The state model used by SHOP2 defines the Minecraft game state as a set of first-order predicates. These predicates can have both numerical and symbolic arguments. More specifically, the state model contains information about the inventory of the agent, such as items in inventory and location of items in the hotbar. This model also contains information about entities and locations the agent has observed, and information about the agent itself, such as its current location and view location. An example of the state model is `{(entity_at cow123 loc123), (inventory_count wheat_seeds 10)}`.

The HTN planning model contains primitive and compound tasks that the agent can do in the environment. Specifically, we categorize primitive tasks in the HTN domain model into the following: *movement*, *look*, *item selection*, *item crafting*, and *item gathering*. We also categorize compound tasks into *top-level tasks* and *helper tasks*. Top-level tasks are objectives that the agent directly wants to pursue (such as making pumpkin pie and cake). Helper tasks are

Table 1. Minecraft Model for Planning with SHOP2

Top-Level Tasks (Food Point Values)	***Helper Task Categories***	***Action 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)		

[3] https://minecraft.gamepedia.com/Food#Foods

those that the agent does in order to complete top-level tasks. Helper tasks can be categorized into *crafting items*, *gathering items*, *growing crops*, and *consuming/using items in inventory*. These categorizations are summarized in Table 1. An example of a plan generated by the SHOP2 planner for obtaining beef would be: `{(move-near-entity cow123), (look-at-entity cow123) (select iron-sword) (attack cow123) (gather beef)}`.

Plans generated by the SHOP2 planner are used by an agent to construct executable actions in Minecraft. At a high level, a plan is parsed and converted into an executable sequence of actions (i.e., an executable plan). This executable plan is then executed in the Minecraft environment to completion. However, the behavior can be different for crafting-related tasks if the agent does not have the necessary items in inventory to craft. Specifically, if the items have been observed in the environment, but are not in the agent's inventory, it constructs a plan to retrieve them. If items have not been observed in the environment, the entire plan is ignored. Once all items have been retrieved, the agent then replans to get a new plan for crafting the item. Replanning makes sense here because the agent may observe items for more important objectives while retrieving items for crafting. In this case, during replanning, the agent should execute the more important objective.

Approach

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 systems here.

Planning and Acting in a Network Decomposition Architecture (PANDA)

Similar to the work by Holler et al. (2018), we use the Planning and Acting in a Network Decomposition Architecture (PANDA; Bercher, Keen, & Biundo, 2014) as an off-the-, shelf HTN planner for planning as goal recognition. PANDA is a hybrid planning algorithm, which combines HTN planning concepts with partial-order causal link planning. We refer to HTN planning for plan and goal recognition using PANDA, as PANDA-REC.

At a high level, PANDA-REC takes as input a sequence of actions and an HTN planning model and converts the planning model into a model for goal recognition. This model is then passed into PANDA, and the recognized goal is extracted (i.e. a single top-level task). We chose to use PANDA for the goal recognition task, rather than SHOP2, because code for the conversion process was readily available. Note that that this approach (i.e., using PANDA for goal recognition) is currently state-of-the-art (Holler et al., 2018).

Analogical Theory of Mind (AToM)

We use the Analogical Theory of Mind model (AToM; Rabkina et al., 2017) as a model of theory of mind (ToM) and apply it to the task of goal recognition. The central claim of AToM is that ToM occurs through analogical processes. It is implemented using the analogy stack in the Companion cognitive architecture (Forbus & Hinrichs, 2017). AToM is trainable 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 *generalization pool* is learned for every potential goal type. Intuitively, a generalization pool is a model, learned from previous observed traces, of the goal it represents. It consists of individual and generalized cases; here, a case consists of the trace of the agent accomplishing the given goal. Depending on the experimental condition (see Experiments, below) this may be the output of the SHOP2 planner, a report of the agent's actual actions, or sensor-like observations of those actions.

During training, cases are passed to SAGE one at a time. If one exists, the most similar previously observed case is retrieved via MAC/FAC (Forbus, Gentner & Law, 1995). We refer readers to the original paper for specifics of the retrieval algorithm. Importantly, however, MAC/FAC 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 `o` and `r` (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*, `g`, which is assimilated back into the generalization pool (see McLure et al., 2015).

Generalizations contain frequentist probabilities of the facts contained in their constituent cases. For example, if a generalization contains 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 added to 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.

If the similarity score between `o` and `r` is not above the threshold needed to form a generalization, `o` is added to the

Table 2. Results for Goal Recognition Experiments

	PANDA-REC	*AToM*	*Uniform Baseline*	*Biased Baseline*
SHOP 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** (0.98)	0.167	0.237

generalization pool as an individual example. Note that generalizations are treated as cases for the purposes of retrieval and `r` may itself be a generalization.

During testing, MAC/FAC is once again used for retrieval. However, retrieval occurs across all generalization pools. The pool from which `r` is actually retrieved is assumed to represent the agent's goal in `o`.

Experiments

The objective of our experiments is to compare different ways to infer an agent's goals given different types of observed sequences of actions. Specifically, we extract sequences of actions (i.e., traces) from both planner output and agent action executions in Minecraft. We then compare AToM with PANDA-REC described in the Approach section above.

All traces used in our experiments were extracted from Minecraft play sessions logs. A 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. From this session, a single log is generated and consists of all planner output and executed actions of an agent throughout the session.

We constructed a dataset of play session logs by having an agent play 10 pseudo-randomly generated maps 5 times, with simulations running for 180 seconds each. Once the log dataset was constructed, we randomly extracted datasets of 100 plan traces (i.e. planner output) and 100 action execution traces. Note that *obtain_carrot* did not appear in the action execution trace dataset.

Two random baselines were computed 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. The highest accuracy for each test is bolded.

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 executable actions generated by the planner (in this case, SHOP2) 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 provided the HTN model used by SHOP2 while AToM learned from SHOP2 outputs via 10-fold cross-validation.

PANDA-REC was 100% accurate in recognizing goals based on the SHOP2 planner's output. This fit our intuition, as PANDA-REC is given the HTN planning model used by the SHOP2 planner. AToM performed worse, with 92% accuracy. Both systems performed substantially better than the uniform and biased baselines.

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

Planner traces provide perfect information about the plans being attempted. 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 actions. We note that some of these actions (i.e. move and look at) contain internal information about the agent. PANDA-REC was provided an HTN model corresponding to the executed actions (HTN-EXEC), 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 agent actions but remained above both baselines. It performed at 63% accuracy. AToM's performance did not change significantly from the planner traces outputs. It maintained 90% accuracy.

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, but not where it is going) and lacked internal state (e.g., the target of the movement).

We tested PANDA-REC and AToM's sensitivity to external knowledge traces under two conditions: (1) with a model that includes internal information from agent traces and (2) with a model that only includes external information. For (1), PANDA-REC was given the action execution HTN model (i.e., HTN-EXEC) used in the previous experiment. AToM learned a model using the whole execution trace dataset. For (2), PANDA was given a modified action execution HTN model (HTN-TOM), which did not contain internal information, and AToM was trained using 10-fold cross-validation, as in previous experiments.

When tested on the external knowledge-only traces using the HTN-EXEC model, PANDA-REC's performance dipped again to 30% accuracy. However, when tested using the HTN-TOM model, it performed as well as it had when using HTN-EXEC on execution traces (i.e., 63% accuracy). AToM performed equivalently across tasks: 90% accuracy when trained on execution traces and 88% accuracy when trained on external knowledge-only traces.

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 the hallmark of human ToM reasoning, which AToM models. Thus, our results suggest that ToM reasoning in general, and ToM reasoning via AToM in particular, 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 `obtain_potato` for `obtain_carrot`, but not `obtain_beef`. 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. 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 in order 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.

Related Work

Goal Recognition is the process 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 con-

trast, 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).

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), 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.

While there are many different approaches to plan and goal recognition in the literature, this work aimed to provide an initial study comparing AToM to goal recognition. There is currently another study being done on applying Combinatory Categorial Grammar plan recognition to the traces presented in the experiments. As such, we aim to look at the remaining plan and goal recognition algorithms presented in this section for future work.

Conclusion

In this work, we have analyzed the performance of two systems on goal recognition as availability of internal information changes. 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, which may lead to better interactions between agents.

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.

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.

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.

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.

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.

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., 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.

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*.