Show Me How to Win: A Robot that Uses Dialog Management to Learn from Demonstrations

Maryam Zare*
muz50@psu.edu
Dept. Computer Science and Engineering
The Pennsylvania State University

Ali Ayub*
aja5755@psu.edu
Dept. of Electrical Engineering
The Pennsylvania State University

Alan R. Wagner*
azw78@psu.edu
Dept. of Aerospace Engineering
The Pennsylvania State University

Rebecca J. Passonneau*
rjp49@psu.edu
Dept. Computer Science and Engineering
The Pennsylvania State University

ABSTRACT

We present an approach for robot learning from demonstration and communication applied to simple board games like Connect Four. In such games, a visual representation of a winning condition on the board can be converted to an extensive form representation that can then support computation of a winning strategy. We present a robot that can learn simple games from responses to visual questions based on synthesized images, or to verbal questions. We illustrate how reliance on both modalities leads to more efficient learning.

CCS CONCEPTS

• **Computing methodologies** → **Discourse, dialogue and pragmatics**; **Reinforcement learning**; **Learning from demonstrations**.

1 INTRODUCTION

We present a novel method of interactive learning from demonstration applied to the problem of teaching a robot to play games. Researchers have begun to rely on interaction and demonstration to teach robots new activities [5, 12]. Learning from demonstration (LfD) can offer a fast, intuitive method for robots to learn new things, where the research effort occurs prior to the actual learning. Some benefits of robots learning from demonstration are to avoid dependence on offline machine learning approaches with very large datasets, and on time intensive learning that requires exploration of factors whose effect on decision making in particular states is unknown, and possibly irrelevant. Our work is inspired by the belief that robots can learn from demonstration more effectively if they can communicate more freely while learning. Robots that can learn about games through multi-modal interaction could foster more widespread use of agents that can play games with humans, and support research into learning other interactive strategic behavior.

Our work develops a robot agent with the ability to interact in a more fundamental way than in previous work. In contrast to prior work, we focus on game-learning, which is a strategic activity. Prior work on robots learning through interaction often focuses on a single task, and selects queries from a repository of query types, as in [11]. Our robot can generate novel queries from general knowledge about a family of games (e.g., Connect Four, Quarto), and from the specific knowledge it acquires about a game during an interaction. Further, it can formulate visual or symbolic questions and compare the knowledge gained from different questions.

We first present related work on learning from demonstration and through communicative interaction. We then present our baseline robot that learns from visual demonstrations, through reliance on game theory. The following section presents a novel use of dialog management combined with general knowledge about a family of games to support the generation of verbal or visual questions. Finally, we present an extended example of an interaction with a perfectly cooperative and knowledgeable simulated user to illustrate how questions are generated and beliefs are updated.

2 RELATED WORK

The field of artificial intelligence has made significant progress in developing systems capable of mastering games such as Chess, poker, and even Go [15, 16]. Deep reinforcement learning has recently been used to train autonomous agents to play a variety of Atari and other games [6]. Although the agent does learn how to play the game with considerable accuracy, the process requires large amounts of data, time, and accurate perception. In contrast to this prior work, our approach seeks to develop a computational architecture and algorithms that allow a robot to learn from a limited number of verbal or visual questions.

Learning from demonstration (LfD) offers a way to reduce the time and effort to teach robots new skills. LfD has been used to learn a variety of tasks like table tennis [10] and drawer opening [12], but there has been less work focused on using LfD to teach interactive games. Some researchers have employed active learning in which a robot uses a question-answer session with a human to learn a new goal-oriented activity [4, 7, 11, 18]. Although using just question-answer sessions reduces the time and data required to learn a skill, it nevertheless is highly context dependent and may not generalize across different tasks.

Another emerging area of research is *communication task learning* [5]. Learning tasks through communication presupposes an ability to communicate, which in turn presupposes the ability to ground language in a particular world (semantic grounding), and

FDG '19, August 26–30, 2019, San Luis Obispo, CA, USA

ACM ISBN 978-1-4503-7217-6/19/08…$15.00
https://doi.org/10.1145/3337722.3341866

to ground the communicative interaction in an evolving representation of the common ground of the discourse (interaction grounding) [5]. Previous work extracts procedural knowledge about new tasks from dialogue for small action sequences focusing on semantic grounding [13][9], without investing the agent with knowledge about communication management. Mohan and Laird [9] rely on the SOAR cognitive architecture to construct an agent that learns to achieve simple action sequences, e.g., to transfer an object from one location to another, from verbal interaction with a human. Scheutz et al. [13] use one-shot learning to extend this type of approach to allow for a novel vocabulary of objects, but still restricts learning to short sequences of actions. Our work addresses the problem of a robot learning to play a game from visual and verbal communication, which goes beyond procedural knowledge to include strategic knowledge. We assume that complete knowledge of the game is achieved from multiple interactions, thus learning can be resumed later, including during actual game play. After an initial instruction session, however, the robot must be able to interact strategically with an unpredictable human partner, rather than execute a predictable action sequence (procedure). Another thread of work on interactive learning between a human and an agent addresses how to decide what communicative action to take based on the agent's estimation of its own learning progress given a human teacher [11] or, its estimation of a human student's current state [8]. In [8], the agent presents a human student with examples and observes the student's response. The goal is to get the student to guess the mean of a given distribution. The agent expects the other party's response to a query to be a real number and compares this to the underlying distribution to choose the next action. The agent's communication at time t_{i+1} is dependent on what it learns from the other party at t_i. The long term goal is for the student to respond with the correct mean. In contrast, in [11], the agent is learning from a human both by observation and by queries. Here the agent uses active learning to capture the sequence of actions necessary to complete a task. The agent's developing model is used to decide which verbal queries to ask the human. The agent expects the human's response to be a boolean, and uses the response to update its posterior distribution over actions in the sequence it is trying to learn. The goal is for the agent to reduce the entropy of its beliefs about an action sequence, and what the agent communicates at t_{i+1} depends on how confident it is that it has learned an action sequence by time t_i.

In both approaches discussed above, there is a fixed range of queries, and the selected query is based on an estimate of how far the agent is from some relatively simple goal such as a specific value

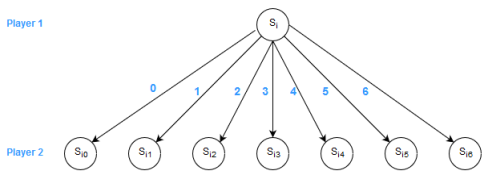


Figure 1: Extensive-form game representation for one stage of the Connect Four game is depicted above. The lower nodes represent the game state after one of the seven actions (0-6) is chosen by player 2, the upper node depicts the current game state when player 1 chooses an action.

(atomic) or a specific sequence (composition of atoms). The queries are also simple, being a request for a real number or a request for the rate of an action. In our work, the range of possible queries is a function of general communication act types (e.g., requests for information) and the agent's current knowledge state (e.g., that the observed game board is a certain size). The agent's goal is to learn general knowledge to carry out strategic game actions for a specific type of game. The learning interaction endpoint is not defined in terms of a specific sequence of actions to be learned. Instead, communication will terminate based on a trade-off between how close the agent is to the goal of learning the game and how much time has been spent asking questions, where the interaction itself has a cost.

3 USING GAME THEORY TO MANAGE ROBOT GAME PLAY

Game theory offers computational representations that have been used to formally represent and reason about a number of interactive games [3, 15, 16]. The normal-form game and the extensive-form game are computational representations from game theory that serve as building blocks to represent a complete interactive game. Sequential stages of an interactive game are represented in extensive-form as a tree. A player's potential actions are denoted by the branches of the tree and nodes of the tree indicate which player makes a decision at each particular stage of the game. Payoffs for selecting particular series of actions are depicted at the stage in which the payoffs are received. Represented in this manner, the challenge of learning a new interactive game is reduced to learning the structure and underlying components of the game theoretic representation.

In our previous work [2], we developed a game-theoretic LfD approach in which a Baxter robot learnt the win conditions of Connect Four by asking questions about demonstrations of win conditions. The game Connect Four can be represented computationally as a perfect information extensive-form game (Figure 1). At each stage of the game both players have complete information about the state of the game, the actions taken by the other player and the actions available to the other player in the next stage. At each turn, a player selects a column to place their colored chip. Figure 1 shows an extensive-form game representation for a portion of the Connect Four game. The nodes represent the game state after a player selects an action from seven actions denoted 0-6 for the column in which the chip is placed. Images of the Connect Four game (Figure 2 left) can be directly translated into an intermediate matrix format (Figure 2 middle) indicating which player has pieces occupying specific positions in the matrix. This matrix can then be used to generate possible extensive-form games (Figure 2 right) that can be checked against the game's win conditions. More importantly, the extensive-form game can be translated back into matrices and used to predict what different game states should look like or, as described later, presented to a person as potential win conditions for verification. This translation process allows the robot to generate and communicate visual examples of game states to a person and ask questions to the person about the game state (e.g. "Is this a win for yellow?").

Learning by demonstration from a human can be considered an inference problem in which the goal is to watch the actions taken

by the human to accomplish the task. In the context of playing the game Connect Four, the person demonstrates a win condition for the game to the robot. Using the approach described above, the win condition is represented as an extensive-form game and the robot attempts to surmise a general rule underlying the win condition by presenting the person with internally generated example board states and asking them whether the board depicts a winning game. Intuitively, the robot's behavior resembles the action of a person trying to learn a new game by generating fictional situations and asking whether these situations would result in a win. To create the extended-form game structure, the robot asks a series of questions, beginning with basic questions. The robot first asks two questions: "How many players can play this game?" And "Is this a type of game in which players take alternative turns?" Once these questions are answered by the person, the robot knows how the player's actions will iterate and a largely empty extended-form game structure can be created. The robot uses pre-programmed information about the basic components of the game, such as what the board looks like, the game pieces and their associated colors, and how to physically perform the actions of placing a game piece (a chip) in a column of the game board. We used open source software for the Connect Four game which includes tools for creating the requisite robot behavior and identifying the Connect Four game pieces [1]. In the future we hope to have the robot learn these items as well.

After the first set-up questions, the robot learns the game's win conditions. First, the robot asks the human for a demonstration of a win condition (e.g. Figure 2 left). The robot waits for the person to state, "I am done" to know that the person has demonstrated the win condition on the board. The robot converts the visual information obtained (image of the static board) into an extended-form game (Figure 2 right). Figure 2 depicts the process of a column win. Clearly there are many other arrangements of the game pieces that will lead to other columns wins. Even though the robot does not have access to all of the game situations it can leverage the human to

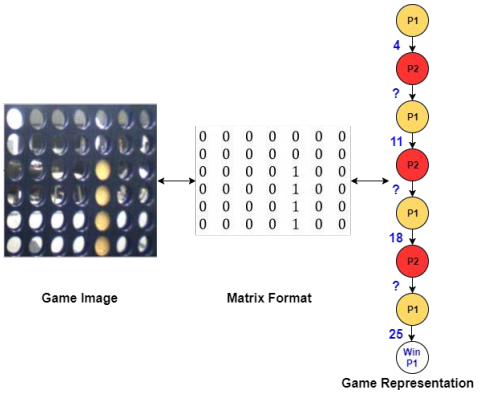


Figure 2: A column win condition for the Connect Four game seen from the robot's perspective is shown above (left). The associated extensive-form representation is shown on the right. Only the actions taken by the robot are depicted without representing all actions available at each stage. The numbers along with the arrows show the action number chosen by the robot (4, 11, 18, 25) and the human (?). (Best viewed in color.)

generate rules for winning. The image of the board obtained from the camera on the robot is altered to reflect possible extended-form games representing example game situations. Along with the image of the game situation a simple yes/no question is asked to confirm if that game situation will be a win condition or not. Through a series of these yes/no (visual) questions the robot learns the rules or constraints related to the demonstrated win condition. In this paper, we show how a dialogue manager can be combined with this game-theoretic learning approach to ask verbal or visual questions about demonstrated win conditions of Connect Four and other similar board games like Quarto.

4 DIALOG MANAGEMENT

Our agent relies on a dialog management module whose purpose is to choose communicative actions strategically. This contrasts with the systems in [9][13][8][11]. This paper illustrate how the dialog manager decides what communicative action to take through a *semi-random dialog policy*, constrained by the simple strategy of selecting communicative actions that will improve the agent's knowledge of the game. In separate work, we show policy learning for three board games: Connect 4, Gobblet and Quarto ([20], under review). At the end of each dialog, the agent has learned how to interact in a game, which is strategic rather than procedural knowledge. In separate work, we build on this framework to learn dialog policies acquired through reinforcement learning. Our approach is inspired by previous work on task-oriented dialog management [14] [19], where the agent has a limited set of a priori actions and is trying to achieve a fixed goal, such as booking a hotel, getting directions to a destination, and so on. The main differences between our design and previous work [14] [19] are: 1) the dialog goal is not based on a pre-defined template, 2) communicative actions are generated from a general representation of game knowledge rather than specified a priori, and 3) dialog length is a tradeoff between how close the agent is to the knowledge goal and how much time has been spent in the interaction.

In the rest of this section we first define the action space for the agent to construct communicative actions. Then we present how the belief space is updated at the end of each turn exchange. Finally, we provide a sample interaction.

4.1 Communicative Action Space

The communication action space allows the agent to question a human partner about how to win prior to actual play, building on the visual questions presented in section 3. It assumes the game is in the family of board games like Connect Four or Quarto. Following the framework presented in section 3, all communicative actions are questions about win conditions. Tables 1-4 present the building blocks for a formal representation of game knowledge, based on a modified first order logic. We first define in Table 1 the variables that are used to specify a representation of a board. Tables 2 - 4 define functions from the known input board representation to an output board representation to be asked about. The values returned by the function in Table 2 are properties of a set of one or more game pieces on a board representation. The functions in Table 3 return new board representations that apply a transformation on coordinates of disks in the input representation. The functions in Table 4 change the number of pieces in the input board representation. Table 5

shows how the agent can construct communicative actions from the building blocks given in Tables 2 - 4. The communicative actions are general, and can apply to other kinds of games. The feature set

Function Name	Range
Feature	pattern, quantity, shape, height, color
Angle	0, 45, 90, 125, 180
Coordinate	$\{(x, y)\|x \in$ #BoardRows, $y \in$ #BoardColumns$\}$

Table 1: Variables used by functions defined in Tables 2 - 4.

Function Name	Meaning
WinProperty(feature = f_i)	A question about f_i

Table 2: Feature function: Enables the robot to ask about the features of disks.

is defined in Table 1, but only "pattern" and "quantity" are applicable for Connect Four. The remaining features are applicable to more complicated games like Quarto, where game pieces come in different shapes, heights, etc. The function shown in Table 2 enables the robot to confirm which features of the disks on a board representation contribute to making it a win condition. That is, the agent can ask whether it is the **quantity** or **pattern** of disks that make a given configuration a win condition.

Table 3 shows functions that allow the robot to ask questions about different ways to reconfigure a given board representation. We have an action named **Permute**, so the robot can ask about the sequence of the actions. **Translate** checks whether the position of the disks on the board are important. For example, the robot can check if four disks on *any* row is a win. The range of offset is defined in Table 1. **Rotate** applies a valid rotation to a configuration of disks by θ degree defined in Table 1, where a valid rotation is one that is possible from a given location. For example, a vertical win condition in the rightmost column can be rotated to the anti-diagonal, but not the diagonal.

Function Name	Meaning
Permute(feature = f_i)	Permute the disks in a board representation
Translate(offset = (x, y))	Translate all of the disks by the offset
Rotate(angle = θ)	Rotate all of the disks by the angle

Table 3: Shift Board Functions: Generate a board representation that re-arranges the disks

The third set of actions shown in Table 4 are for questions where the current board representation is modified by changing the number of disks. For example the robot can ask if the result of adding a disk in a given cell is a win condition.

Function Name	Meaning
AddPiece(position = (x, y), feature = f_i)	Add a disk with feature f_i at position (x, y)
RemovePiece(position = (x, y), feature = f_i)	Remove a disk with feature f_i from position (x, y)

Table 4: ChangeDisk Functions that alter the disks in the board representation.

The first part of Table 5 shows the communicative action types, five of which apply to Connect Four. Parameter ID specifies which player the robot is talking about. If **WinValue=True**, the question is a yes/no question. If it is **None** the question asks whether the new board configuration results in winning or losing the game. A complete question is generated by selecting a communicative action type and instantiating it with relevant game state functions from Tables 2 - 4. Note that this includes complex questions that use both a ChangeDisk function and a ShiftBoard function. The robot can also ask the open-ended question, **RequestNewWinCondition()**.

The sixth communicative action shown in Table 5, **RequestOtherPlayer()** is to question whether the other player can undo a possible win condition after it has been reached, which is relevant for games like chess. If the answer is yes, then the agent can ask for specifics about how the other player can undo a win condition.

The second part of Table 5 shows the communicative responses the simulated user can make. For the purpose of this paper, we assume the simulated user is completely knowledgeable, understands all the questions, and always responds cooperatively, e.g., with the correct answer to a yes/no question.

4.2 Belief State

An interaction consists of a sequence of turn exchanges in which the agent asks a question and the simulated user responds. The belief state at each turn exchange of the dialog shows the knowledge the robot has acquired up to that point in time. We define the belief state to be a concatenation of vectors that represent beliefs about each function defined in Tables 2 and 3, plus a vector representing the coordinates of game positions, plus a vector for the other player. At the beginning of the dialog all the probabilities are accumulated over "None" which means the robot has no information. Formally,

Communicative Actions of Robot	
Function Name	**Meaning**
RequestInfo(ID, ChangeDisks, ShiftBoard, WinValue)	Add/remove a disk to/from the board, then reconfigure the board and ask a question
RequestInfo(ID, ChangeDisks, WinValue)	Add/remove a disk to/from the board and ask a question
RequestInfo(ID, ShiftBoard, WinValue)	Reconfigure the board and ask a question
RequestInfo(ID, WinProperty, WinValue)	Ask a question about one of the features
RequestNewWinCondition ()	Ask for a new win rule
RequestOtherPlayers()	Ask about the effect of other players actions
Start()	Start the interaction
Finish()	End the interaction
Communicative Actions of Interlocutor	
Inform()	Indicate if the configuration is a win/loss
Affirm()	Positive answer to a yes/no question
Negate()	Negative answer to a yes/no question
Start()	Start the interaction
Finish()	End the interaction

Table 5: This table shows the communicative actions of robot(top) and interlocutor (bottom).

the belief vector B_t is defined as:

$$B_t = P_{Property_t} \oplus P_{Permute_t} \oplus P_{Translate_t} \oplus P_{Rotate_t} \oplus P_{OtherPlayer_t} \oplus P_{Coordinate_t} \quad (1)$$

The belief vector gets updated through two generic formulas proposed in [17]. When an agent takes a turn and gets a response at time t, the component belief vector $vect_{t_i}$ gets updated if the turn exchange is a question and answer about a function or a property. When the robot asks for a new win condition in the middle of an interaction, a new belief vector to represent a new configuration of disks is initialized, and subsequent turn exchanges are assumed to be about this belief vector. (Designing a more sophisticated belief and dialog state tracking to check for consistency between new and old information is part of our future work.) When the response from the user is positive or contains a new win condition, the corresponding belief vectors get updated according to equation (2), where P_{u_t} represents the confidence over the user utterance. We set this variable to be always 1.0.

$$P_{vect_t} = 1 - \left(1 - P_{vect_{t-1}}\right)\left(1 - P_{u_t}\right) \quad (2)$$

When the user response is negative the relevant sub-belief vectors are updated according to equation (3).

$$P_{vect_t} = \left(1 - P_{vect_{t-1}}\right)\left(1 - P_{u_t}\right) \quad (3)$$

Due to the simplicity of Connect Four, it is possible to represent complete knowledge of all ways to win, the ultimate learning goal.

4.3 Example

We show a sample dialog in which the robot can learn the constraints of Connect Four from a simulated user, using a *semi-random dialog policy* (i.e., the next dialog act type is randomly selected). The simulated user is deterministic, and answers every question completely. For purposes of illustration, we forced a semi-random series of questions that would show a belief update to every component of the belief state. The example shows each turn exchange, with a formal representation of the utterances. We also produce a natural language response using a template. Due to space limitations, we show the belief state only at the beginning of the interaction (Figure 3a), and at the end (Figures 3b-3c). This example shows the gains in both dialog efficiency and effectiveness provided by the dialog manager. Before, the robot could ask a maximum of 11 visual questions ("is this a win for yellow?"), to learn a single win rule out of four possible win rules: a four-disc sequence on the horizontal, vertical, diagonal, or anti-diagonal. With the dialog manager, the robot can acquire more diverse knowledge from a shorter interaction: here it learns two specific ways and one general way to win from only 7 questions.

At the beginning of the interaction, the robot has no beliefs about how to play the game, thus in Figure 3a, the disc coordinates vector is empty, and all the other vectors have all the probability mass associated with the "None" position of the vector. From turn exchange 1, it acquires the knowledge that four in a row horizontally starting in the second column is a win condition.[1] The updated belief vector for disc coordinates appears in Figure 3b. From 2, it learns that the discs can be placed in any order (see permute). From 3, it generalizes this to any row and offset (see translate). From 4, it learns that the other player cannot undo any of its learned win conditions (see other player). From 5 it learns that four in a vertical row in the last column is a win, as shown in the final disc-coordinates vector in Figure 3c. From 6, it learns the previous pattern can be rotated to the anti-diagonal, the only feasible rotation for this column position (see rotate in Figure 3c). From 7 it learns that exactly four discs in these patterns are required (quantity property).

- Turn Exchange 0.
  - Agent (Formal Rep): Start()
    NL: Hello! Can you teach me how to play Connect Four?
  - Sim User (Formal Rep): Start()
    NL: Hi! Yes!

  (Initial belief shown in Figure 3a).
- Turn Exchange 1.
  - Agent (Formal Rep): RequestNewWinCondition()
    NL: Can you show me a win condition?
  - Sim User (Formal Rep): Inform().
    NL: Sim user demonstrates a win condition by placing pieces on the board (cf. Figure 2 for a similar exchange.))

  (Belief over disk coordinates gets updated at the end of this turn exchange.)
- Turn Exchange 2.
  - Agent (Formal Rep): RequestInfo(1, Permute(), WinRule= None)
    NL: Does it matter in which sequence I put my disks on the board?
  - Sim User (Formal Rep): Inform()
    NL: No, any sequence ends up in a win.

  (Belief over permute gets updated at the end of this turn exchange.)
- Turn Exchange 3.
  - Agent (Formal Rep): RequestInfo(1, translate(All), WinRule= None)
    NL: Is it a win for any column and row?
  - Sim User (Formal Rep): Inform()
    NL: Yes!

  (Belief over translate gets updated at the end of this turn exchange.)
- Turn Exchange 4.
  - Agent (Formal Rep): RequestOtherPlayer()
    NL: How can the other player affect undo a win?
  - Sim User (Formal Rep): Inform()
    NL: No. They have no effect.

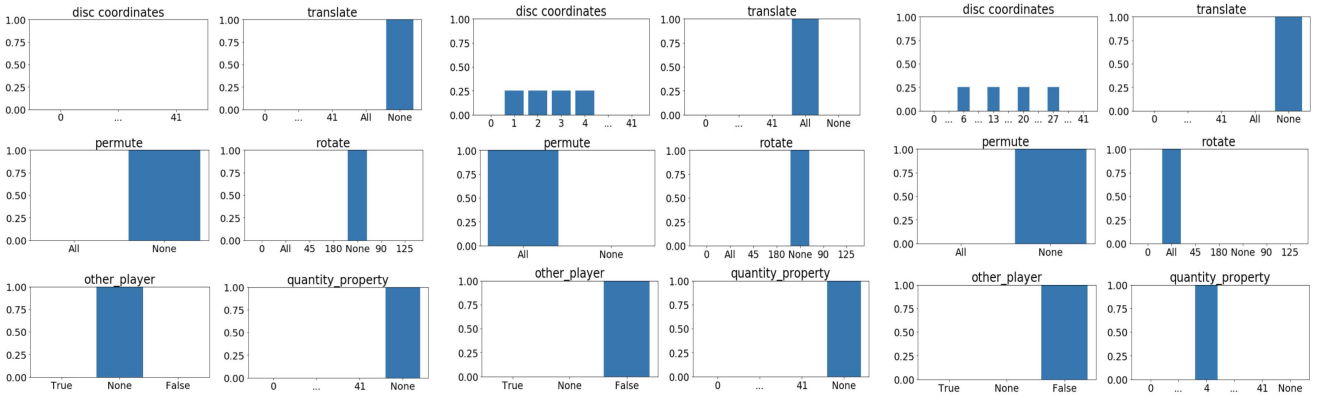
  (Belief over other player gets updated.)
- Turn Exchange 5.
  - Agent (Formal Rep): RequestNewCondition()
    NL: Can you show me a new win condition?
  - Sim User (Formal Rep): Inform()
    NL: Sim user demonstrates a win condition by placing pieces on the board. cf. Figure 2 for a similar exchange.))

  (A new belief vector is initialized and belief over coordinates gets updated at the end of this turn exchange.)
- Turn Exchange 6.
  - Agent (Formal Rep): RequestInfo(1, Rotate(All), WinRule= True)
    NL: If I rotate my disk by all valid degrees, I still win, right?
  - Sim User (Formal Rep): Affirm()
    NL: Yes!

  (Belief over rotate gets updated at the end of this turn exchange.)
- Turn Exchange 7.
  - Agent (Formal Form): RequestInfo(1, Property(quantity), WinRule= None)
    NL: Is it a win because of the number of disks on the board?
  - Sim User (Formal Rep): Affirm()
    NL: yes.

  (Belief over quantity gets updated at the end of this turn exchange.)

[1] The win condition given by the user is reflected in the belief state update.



(a) Initial belief state before asking any question **(b) First learned win condition at the end of dialog** **(c) Second learned win condition at the end of dialog**

Figure 3: Agent beliefs at the beginning and end of the example interaction with the simulated user.

- Turn Exchange 8.
  - Agent (Formal Rep): Finish()
    NL: I think I know enough to play. Thanks! GoodBye!
  - Sim User (Formal Rep): Finish()
    NL: Bye!

  (Final belief is shown in Figure 3b-3c).

The final belief state shown in Figures 3b-3c is converted to a generalization over the type of extensive form tree shown in Figure 2. Instead of an enumeration of the winning paths learned from the interaction, the generalized extensive form tree shows any specific winning paths that have not been generalized over, and any abstract path sets in which some of the nodes and arcs are generalizations over specific actions. Figure 4 shows the learned abstracted extensive form tree, along with colored ellipses for ease of reference. A specific win path in the pink ellipse shows actions that place disks in the bottom four cells of the rightmost column, corresponding to the disc coordinates belief vector in Figure 3c. A specific win path is shown in the green ellipse for the anti-diagonal pattern, that combines the disc coordinates and rotate belief vectors

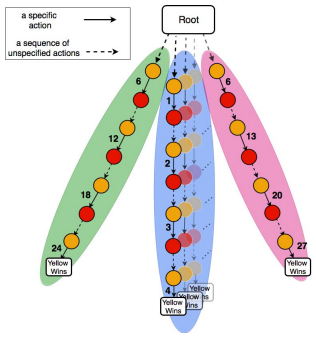


Figure 4: Robot knowledge at the end of the example dialog, as a generalization of the extensive form representation. From left to right the branches show the knowledge about a specific anti-diagonal win pattern (green ellipse), a specific vertical win (pink ellipse), and the set of all horizontal win patterns (blue ellipse).

from Figure 3c. The blue ellipse encloses an abstracted path, shown here as a set of all paths that constitute a horizontal pattern of four discs in a row, which corresponds to the belief state in Figure 3b.

5 CONCLUSION

Our robot that can ask symbolic questions about game demonstrations can now learn more about the game of Connect Four in a shorter interaction than in its previous incarnation, where it could only ask visual questions. Our robot can also now learn about similar games, such as Quarto and Gobblet, using much the same machinery. The specific contributions that make this possible are a framework for generating novel symbolic questions about how to play a board game, a division of labor in the game state representation between general communicative knowledge and knowledge to represent states in specific kinds of games, and a belief update process for the robot to update its beliefs about how much of the game it knows so far. After one interaction, the robot's belief state is saved into a file that can be reloaded into the robot so that it can continue to learn more about the same game in a subsequent interaction.

Our immediate next steps are for the robot to learn a dialog management policy for Connect Four or Quarto through simulated interaction, to compare the policies for the two games, and to apply the learned policies in interaction with humans. In each interaction, a counter will track how much time the robot has already spent interacting with a given partner, and balance the cost of continuing the session against the benefit of additional knowledge that might be gained. In the longer term, we plan to add natural language generation so the robot can generate alternative verbalizations of its symbolic questions. We also plan to extend the work to cover richer games where it is either inconvenient or impossible to enumerate all the win conditions.

6 ACKNOWLEDGEMENTS

This work was funded in part by Penn State's Teaching and Learning with Technology (TLT) Fellowship, and an award from Penn State's Institute for CyberScience.

REFERENCES

[1] 2013. Connect Four Demo: Rethink Robotics. (2013). http://sdk.rethinkrobotics.com/wiki/Connect_Four_Demo.

[2] Ali Ayub and Alan R Wagner. 2018. Learning to Win Games in a Few Examples: Using Game-Theory and Demonstrations to Learn the Win Conditions of a Connect Four Game. In *International Conference on Social Robotics*. Springer, 349–358.

[3] Conway H. J. Berlekamp, E. and R. Guy. 1982. Winning Ways for your Mathematical Plays: Games in general. *Academic Press* (1982).

[4] M. Cakmak and A. L. Thomaz. 2012. Designing robot learners that ask good questions. Proceedings of the seventh annual ACM/IEEE International conference on Human-Robot Interaction.

[5] Joyce Y. Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. 2018. Language to Action: Towards Interactive Task Learning with Physical Agents. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 2–9. https://doi.org/10.24963/ijcai.2018/1

[6] Borghoff U. M. Dobrovsky, A. and M. Hofmann. 2016. An approach to interactive deep reinforcement learning for serious games. 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom).

[7] T. R. Hinrichs and K. D. Forbus. 2014. X Goes First : Teaching Simple Games through Multimodal Interaction. *Advances in Cognitive Systems* 3 (2014), 31–46.

[8] Francisco S. Melo, Carla Guerra, and Manuel Lopes. 2018. Interactive Optimal Teaching with Unknown Learners. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 2567–2573. https://doi.org/10.24963/ijcai.2018/356

[9] Shiwali Mohan and John E. Laird. 2014. Learning Goal-oriented Hierarchical Tasks from Situated Interactive Instruction. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14)*. AAAI Press, 387–394. http://dl.acm.org/citation.cfm?id=2893873.2893934

[10] Kober J. Kroemer O. Mulling, K. and J. Peters. 2013. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research (IJRR)* (2013).

[11] Mattia Racca and Ville Kyrki. 2018. Active robot learning for temporal task models. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 123–131.

[12] Mukadam M. Ahmadzadeh S. R. Chernova S. Rana, M. A. and B. Boots. 2017. Towards robust skill generalization: Unifying learning from demonstration and motion planning. Conference on Robot Learning(CoRL).

[13] Matthias Scheutz, Evan Krause, Brad Oosterveld, Tyler Frasca, and Robert Platt. 2017. Spoken Instruction-Based One-Shot Object and Action Learning in a Cognitive Robotic Architecture. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1378–1386. http://dl.acm.org/citation.cfm?id=3091282.3091315

[14] Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management. (2016).

[15] Huang A. Maddison C. J. Guez A. Sifre L. Driessche G. v. d. Schrittwieser J. Antonoglou I. Panneershelvam V. Lanctot M. Dieleman S. Grewe D. Nham J. Kalchbrenner N. Silver, D. and I. Sutskever. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489.

[16] Hubert T. Schrittwieser J. Antonoglou I. Lai M. Guez A. Lanctot M. L. Sifre Kumaran D. Graepel T. Lillicrap T. Simonyan K. Silver, D. and D. Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv Reprint arXiv:1712.01815* (2017).

[17] Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*. 423–432.

[18] Rosen E. MacGlashan J. Wong L. L. Whitney, D. and S. Tellex. 2017. Reducing Errors in Object-Fetching Interactions through Social Feedback. IEEE International Conference on Robotics and Automation (ICRA).

[19] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proc. IEEE* 101, 5 (2013), 1160–1179.

[20] Maryam Zare, Ali Ayub, Alan Wagner, and Rebecca Passonneau. 2019. In Review, Learning Board Games. (2019).