The Pennsylvania State University
The Graduate School

AN AGENT LEARNING DIALOGUE POLICIES FOR SENSING PERCEIVING AND LEARNING THROUGH MULTI-MODAL COMMUNICATION

A Dissertation in
Computer Science and Engineering
by
Maryam Zare

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2021

The dissertation of Maryam Zare was reviewed and approved by the following:

Rebecca J. Passonneau
Professor of Computer Science and Engineering
Dissertation Advisor
Chair of Committee

Robert T. Collins
Professor of Computer Science and Engineering

Rui Zhang
Professor of Computer Science and Engineering

Alan R. Wagner
Professor of Aerospace Engineering

Chitaranjan Das
Department Head of Computer Science and Engineering

Abstract

Language communication is an important part of human life and a natural and intuitive way of learning new things. It is easy to imagine intelligent agents that can learn through communication to for example, help us in rescue scenarios, surgery, or even agriculture. As natural as learning through language is to humans, developing such agents has numerous challenges: Language is ambiguous, and humans convey their intentions in different ways using different words. Tasks have different learning goals and some are more complex than others. Additionally, humans differ in their communicative skills, particularly, in how much information they share or know. Thus, the agent must be able to learn from a wide range of humans and to adapt to different knowledge goals.

This work proposes **SPACe**, a novel dialogue policy that supports **S**ensing, **P**erceiving, and **A**cquiring knowledge through **C**ommunication. SPACe communicates using natural language, which is translated to an unambiguous meaning representation language (MRL). The MRL supports formulating novel, context-dependent questions (e.g. *"wh-"* questions). SPACe is a single adaptive policy for learning **different tasks** from **humans who differ in informativeness**. Policies are modeled as a Partially Observable Markov Decision Process (POMDP) and are trained using reinforcement learning. Adaptation to humans and to different learning goals arises from a rich state representation that goes beyond dialogue state tracking, to allow the agent to constantly sense the joint information behavior of itself and its partner and adjust accordingly, a novel reward function that is defined to encourage efficient questioning across all tasks and humans, and a general-purpose and extensible MRL. As the cost of training POMDP policies with humans is too high to be practical, SPACe is trained using a simulator. Experiments with human subjects show that the policies transfer well to online dialogues with humans.

We use games as a testbed, and store the knowledge in a game tree. Games are similar to real-world tasks: families of related games vary in complexity as do related real-world tasks, and present a problem-solving task where the state changes unpredictably due to the actions of others. Game trees are a well-studied abstraction for representing game knowledge, reasoning over knowledge, and for acting on that knowledge during play. We have tested our agent on several board games, but the methodology applies to a wide range of other games. The agent's learning ability is tested in a single dialogue and across a sequence of two dialogues. The latter is particularly important for learning goals that are too complex to master in one dialogue. Tests of the agent to learn games not seen in training show the generality of its communication abilities. Human subjects found the agent easy to communicate with, and provided positive feedback, remarking favorably on its ability to learn across dialogues *"to pull in old information as if it has a memory"*.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank the many people that have helped, mentored, and encouraged me over the past few years of graduate school. I am extremely grateful to my supervisor, Rebecca Passonneau, for her invaluable advice, support, and patience during my PhD study. I would like to thank all the members in my PhD committee. Their immense knowledge and plentiful experience have encouraged me in all the time of my academic research. Finally, I would like to express my gratitude to my family and friends. Without their tremendous understanding and continuous support in the past few years, it would be impossible for me to complete my study.

Chapter 1 Introduction

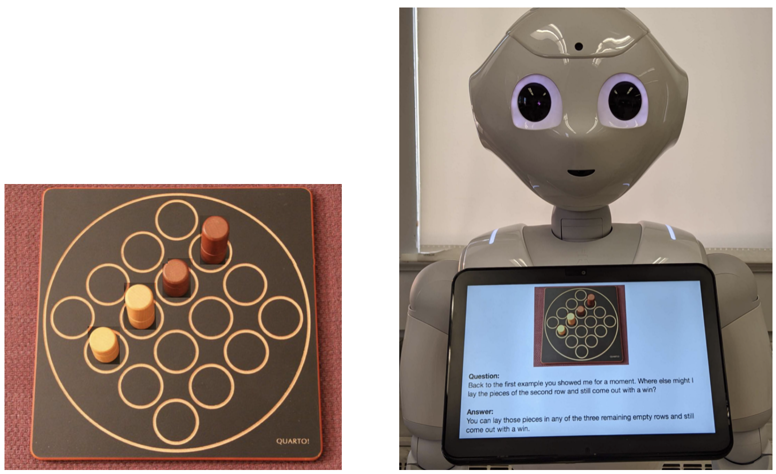
Artificial agents that learn tasks through communication can collaborate with people more effectively on real-world or virtual-world activities. Nowadays, artificial agents are used in a wide range of tasks, from smart vehicles on the road, to agriculture and even robotic surgery. These agents could be more flexible, potentially better partners, and truly collaborative if they can learn while collaborating on a task by asking humans questions to learn more about the current goals and how to pursue them. Language perhaps is the fastest, most compelling, and natural way of teaching new things as it is the way humans learn: by communicating with others through language while interacting with the world. Despite the obvious benefits, there is very limited work in this domain due to characteristics of this problem that make it very challenging.

1.1 Challenges in Learning Through Communication

Humans use language as a vehicle to communicate their intentions. We use words to request for information, that is, to carry out communicative actions, and to pursue communicative goals (e.g. to warn someone, to describe an action or object, and so on.) However, all human languages are highly ambiguous in word meaning, sentence structure, and communicative function. If humans are to guide agents and teach them new things through language communication, agents must have the capability to understand the context specific meanings of the words, to relate and connect them to the physical world (e.g. actions, objects, and so on), and to ground an internal representation of utterance meaning for reasoning about the world. From this perspective, a dialogue is an interaction between two agents who choose and execute communicative actions in sequence in a complex alignment.

Developing task-based dialogue agents is challenging. The agent has to have a way to measure how far it is from a specific learning goal, and be able to make online decisions to choose communicative actions that allow it to progress towards the goal. This requires the agent to not only have a way of formulating communicative actions, but also a strategy to pick an action at each step, and to take into account that the responses of the dialogue partner are unpredictable. If a question receives an answer, the agent might follow-up with a related question. If the response to a question from the agent is a non-answer, the agent might need to switch to an entirely new context.

Another challenge is that humans differ in what they know and in their communication skills, particularly in how much information they share in response to questions about a particular domain. This can depend on their knowledge, personal disposition, how much engaged they are in a conversation, and so on. Human-human communication relies on many kinds of online adjustments that govern how we choose to express ourselves to



((a)) D_1 Board ((b)) Pepper

Question: Back to the first example you showed me for a moment. Where else might I lay the pieces of the second row and still come out with a win?

- Informative Answer: You can lay those pieces in any of the three remaining empty rows and still come out with a win.

- Uninformative Answer: It'll be a win if they are put down on the first row.

Figure 1.1: The type of multi-modal dialogue SPACe agents engage in is illustrated here. The agent is trying to learn the Quarto board game. We see a demonstration of a win condition for the Quarto board game, along with an example question the Pepper robot can generate on-the-fly about this board, followed by an informative and uninformative answer to the same question, taken from a corpus we gathered to train our adaptive POMDP dialogue policies. The first answer provides all the other board locations for four pieces in a row, while the other answer provides only one.

particular interlocutors who differ in the way they communicate, in a given context, in pursuit of a given communicative goal. Similarly, given a task to learn, the agent will have better communication skills if it can accommodate to differences across interlocutors. The agent should have a way to sense differences in information return and adjust its questioning behavior to accommodate. In other words, given a specific task and a specific dialogue partner, the agent should be strategic in asking questions, that work best based on the learning goal and the dialogue partner (i.e., should pose questions that have an expectation of high information gain). For example in Figure 1.1, the Pepper robot is trying to learn the Quarto board game. Pepper is presented with a demonstration of how to place four pieces in a row on a Quarto game board that would count as a win. The agent asks for other locations of a row win, followed by two answers from a dialogue corpus described later in this thesis. The first informative answer indicates that any of the empty rows is a possible location, while the second uninformative answer refers only to the first row. As this thesis will show, depending on the types of answers the agent receives (informative or uninformative), the agent's strategy for asking the next questions should be different, to maximize the information the agent acquires (e.g., It might be better to ask "yes/no" questions from an uninformative partner that is not volunteering a lot of information when is prompted with open-ended questions).

Finally, different tasks have different learning goals. While some tasks might be easy to learn in span of a short dialogue, others might be too complex to learn in one dialogue. Given that a dialogue partner might not tolerate too many questions, the agent should be able to continue to learn across dialogues, where each dialogue might be with a different dialogue partner.

1.2 Our Approach

This thesis takes a step towards general agents that can learn a task or strategic activity through communication by addressing the above challenges. There are different ways tasks can be defined or formulated. Some tasks are procedural, with strict and pre-specified steps to follow to achieve the goal, like opening a drawer or making a strawberry smoothie. Other tasks are more strategic like how to learn a game, where the agent has to pursue goals and make online decisions in an uncertain environment that includes unpredictable behavior of other agents. This thesis focuses on the latter learning goal for the communicating agent. We use games as a testbed. In all their variety, games model fundamental characteristics that apply to all human activities: pursuit of goals in consideration of potentially unpredictable states of the environment, and in consideration of potentially unpredictable actions of other agents in the game. The agent presented in this work develops dialogue policies to learn certain two-player zero-sum board games, but we believe our methodology generalizes to a wide range of board games.

We propose a dialogue agent that can learn through communication. The agent uses a meaning representation language (MRL) which facilitates unambiguous communication as it represents the underlying intentions of the questions and the responses. The MRL is compositional and has general operators to generate questioning actions like *Requesting information*, that can be can be used to formulate specific questions in a given context (e.g. *Requesting information about different ways a game board currently under discussion can be reconfigured.*). The MRL allows the agent to ground the things it learns in two specific ways. First, it provides a way for the agent to relate the words to a representation of the physical world (game pieces, game actions, and so on). Second, it allows the agent to ground the things it learns during a dialogue into an abstract game knowledge representation. To represent the game knowledge, we rely on extensive-form game trees. We rely on game trees for representing the knowledge the agent acquires because with a game tree, the agent can act on the knowledge it learns during play. This is a good evaluation metric as learning success can be measured as the average wins, losses and draws during play. Additionally, as we will show later, a game tree supports various kinds of reasoning, and can be utilized as a long-term memory which the agent can use to further its knowledge acquisition across multiple dialogues.

We cast dialogue as a Partially Observable Markov Decision Process (POMDP) where a dialog policy can be learned using reinforcement learning in which an agent's choice of communicative action, given the dialog goal, depends on the current state. POMDPs have been widely used for task-oriented dialogue. But to our knowledge they have not been used for learning through communication. Our dialogue policy learning relies on offline interactions with a simulated partner using the MRL. A distinctive difference between our work and previous work in learning through communication is that we do not assume that the humans the agent is going to learn from would have the same communicative behavior. Depending on many things including how much a person knows about a specific domain, how much they pay attention to a conversation, their ability to articulate their intentions, and so on, humans can differ in their informativeness. Therefore, we design an agent that takes these differences into account. We investigate differences among policies that are trained with simulated dialogue partners that have different informativeness. Through several experiments, we demonstrate first, that humans do vary in their informativeness, and second, these differences affect the policies in the way the agent asks questions, and in how much the agent learns. For example, we show that it is better to ask more *"yes/no"* questions from a partner who tends to withhold information and more open-ended questions from partners who are generous with information.

After gathering evidence that humans' informativeness has an impact on the policies the agent learns, we develop an agent that can adapt to the differences in amount of information humans provide. This is achieved using a novel belief state component, *information dynamics*, that tracks the specificity of the agent's questions and the amount of information in people's answers. This allows the agent to ask questions tailored to the types of responses it has been receiving. The agent's ability to track information dynamics from turn-to-turn also supports offline learning of POMDP policies in simulation, that transfer directly to online dialogues with people. This addresses a fundamental limitation of POMDP policies: dependence on massive numbers of learning trials that people would be intolerant of. Furthermore, using the game tree as a long-term memory, we demonstrate that the agent can continue learning across a succession of dialogues. This is particularly important for an agent whose goal is to learn something, as some tasks are too complex to be learned in one dialogue and people might not tolerate long dialogues.

We evaluate the agent in dialogues with simulators, where we can control the informativeness, and with people, where we can measure their informativeness post-hoc. Experiments show that our agent learns as much as oracle baselines which are specifically designed to perform well for a dialogue partner with informativeness x, in single dialogues. Further, in a sequence of dialogues, the agent learns well independent of whether the second partner's informativeness is higher, lower, or the same as in the first dialogue. The policies transfer easily to dialogues with people. Additionally, the agent exhibits two unanticipated and beneficial behaviors resulting from information dynamics. First, in single dialogues where the simulator is only moderately informative, the adaptive policy outperforms the oracle, due to greater diversity in its questions. Because information dynamics tracks things like which properties of game pieces the agent has asked about in the past, it learns to ask about new properties. Second, if the simulator in a second dialogue is more informative than in a first one the agent discovers it can *retry* a question that failed previously, and finally get an answer.

1.3 Contributions

Here are the contributions of the work presented in this thesis:

- We present a novel meaning representation language (MRL) for commumication about games that is compositional, to provide unambiguous representation of the communicative intentions of the dialogue utterances. Because the MRL is compositional, in contrast to previous work where queries are limited and often pre-defined, our MRL allows the agent to formulate general communicative actions (e.g. *Request information*) on-the-fly for specific contexts and thus **generalizes to many games**.

- We introduce a novel application of POMDP policy learning for dialogue systems. Although, POMDP dialogue polices have a long history, this is the first work that we know of that uses a POMDP dialogue policy for developing agents that **learn** new things. We propose a hierarchical dialogue policy that is trained offline using reinforcement learning, enables the agent to easily talk about previous contexts, and is learned through a novel reward function that encourages the agent to learn as much as possible through efficient questions.

- We believe this is the first work that takes into account the difference between people in terms of how much information they tend to share, and how it can affect the polices the agent learns, and introduce an agent that adapts to these differences. We achieve this by introducing a novel component of belief state in dialogue systems which we refer to as **information dynamics**, that allows the agent to continuously sense the changes in the questions it asks and the answer it receives and adjust accordingly. In the previous work that we know of, it is assumed that for the system to work people must be informative.

- Grounded task learning through language usually addresses grounding objects or actions (e.g. color red is grounded as `Red`). We take a step forward and in addition to providing a way for the agent to relate language to the physical world, we enable the agent to ground the game knowledge it learns directly into a game tree representation. We believe this is a step forward to developing agents with a more general learning capability (i.e. agents that can learn new things **unseen** during training).

- One of the unique features of our agent is its persistent knowledge store that is an extensive-form game tree. Because of the game tree the agent has a knowledge store that can be used to play with people, and to continue to learn complex tasks from other partners in successive dialogues. Because the MRL is grounded in both the game tree abstraction and in properties of game boards, the agent was able to produce an unexpected and extremely beneficial behavior as a by-product of the dialogue policy design. In a new dialogue, if the agent encounters a more informative person than in a previous dialogue, the agent can **retry** a question that failed with the earlier dialogue partner, thus opportunistically filling in its knowledge gap.

1.4 Thesis Road Map

Chapter 2 discusses the relevant work. Our work draws on distinct lines of research. The distinction between perception and cognition aligns with work in cognitive architectures [1, 2], which are used for learning through communication. Our proposed information dynamics is informed by studies of how and why people synchronize their behavior in dialogue [3,4], and how we can build artificial agent that can adapt by equipping the agent to sense the changes in its environment [5], which in a dialogue we cast as the change in the amount of information exchange between the agent and the partner. Our use of MDP policies draws from a long tradition in task-oriented spoken dialogue systems [6, 7].

We introduce a meaning representation language (MRL) in Chapter 3. We explain how the MRL allows an unambiguous communication and how different communicative actions are defined to enable the agent to ask context-specific questions. We also explain how we use game trees as a long-term knowledge store, and how the MRL enables the agent to ground the game knowledge it acquires during a dialogue in a game tree.

Next in Chapter 4, we show how through reinforcement learning, we can train dialogue policies that use the MRL to communicate with a dialogue partner and learn a game. We provide background information on how we cast the dialogue policy as a Markov Decision Process (MDP). Because of differences in communicative skills and behavior and in general the change in attention and engagement in a dialogue, humans will differ, in how much information they offer. Therefore, we investigate differences among polices that are trained with simulated dialogue partners who have different informativeness. Our experiments with human subjects provide evidence that they vary in their informativeness and that these differences impact the polices in two different ways: in how much the agent learns and the questioning behavior of the agent. We also tests the agent in playing games with subjects to evaluate the game knowledge it learns in a dialogue.

Motivated by the results in Chapter 4, Chapter 5 introduces an agent that senses the differences in its partners in terms of how generous they are with information and learns a policy that is adaptive to these variations. We explain how we achieve this using a novel component of dialogue belief state, *information dynamics*, that monitors the information requests and information return between agent and interlocutor. We explain how this ability arises from the range of communicative actions defined in Chapter 3, and a novel reward structure for reinforcement learning of the policy that encourages the agent to interleave two learning goals: to ground what it told or shown in a specific dialogue, and to further its game knowledge acquisition.

Chapter 6 presents several experiments with the adaptive agent introduced in Chapter 5. We first demonstrates how an agent with a single adaptive policy can learn any of the games it was trained for, under varying conditions of simulator informativeness. We further demonstrate that the adaptive policy is a step towards general communication skills by its ability to learn about situations never seen during training, and to continue learning in a sequence of dialogues with different humans. Finally, through a study with human subjects, we show that the agent can communicate effectively with people, and can keep learning a complicated game in a succession of dialogues, that people find the agent easy to interact with, and are impressed with the agent's ability to recall previous dialogues. We conclude the thesis by discussing the limitations of the current work and future work in Chapter 7.

Chapter 2 Related Work

In this chapter we review related work, starting with research on different approaches for learning through communication in section 2.1. Then we review works that are focused on language grounding in communication in section 2.2. We next talk about relevant work that addresses the notion of adaptation for developing artificial agents in section 2.3. We conclude this chapter by reviewing the relevant work to POMDP dialogue systems trained with reinforcement learning.

2.1 Learning Through Communication

Learning from demonstration (LfD) offers a way to teach robots new skills without the cost of applying machine learning to large datasets. LfD has been used to learn a variety of simple tasks like hitting a ping pong ball [8] and opening a drawer [9]. A logic-based approach was applied to learn games like Connect Four and Tic-Tac-Toe from demonstration videos, using pre-defined relations for game states [10], as in [11]. Some researchers have employed active learning in which a robot asks clarification questions of a human to respond to fetching requests [12], selects pre-defined queries to learn simple task sequences [13], or selects a specific question type to learn a given skill [14].

Interactive Task Learning has also been applied to learn games like Tic-Tac-Toe and Tower of Hanoi [15], but without developing a general approach to learning through communication. In [16], the agent can resolve the meanings of polysemous words by means of a general formulation of the learning task in terms of game states, legal actions, and a compositional representation of task elements that generalizes across tasks, partly through the vehicle of a cognitive architecture [1]. In one approach, the SOAR cognitive architecture is used to construct an agent that learns procedures, such as how to change an object's location, from verbal interaction with a human [17]. One-shot learning extends this type of approach to allow for a novel vocabulary of objects [18]. Our work also relies on a compositional meaning representation. Instead of using a cognitive architecture that models human sensing and cognition, we extend POMDP policy learning for dialogue management to incorporate an explicit mechanism to sense the behavior of the dialogue partner.

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 [13] or, its estimation of a human student's current state [19]. In [19], 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 [13], 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 (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. Additionally, dialogue policies are learned through reinforcement learning that allow the robot to vary its questions during an interaction, depending on the current context. Each turn has a cost, which enforces efficient questions.

2.2 Language Grounding in Task Learning

Previous work on learning through communication has addressed concept grounding or task learning, rather than learning how to act when the state changes due to other agents' actions. In [20], machine-learned classifiers ground words and phrases provided by a human in an agent's perception of the world. Language can also be grounded more directly in perception, by machine learning the relevant perceptual categories from data, rather than pre-specifying them in a formal semantics [21].

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 to ground the communicative interaction in an evolving representation of the common ground of the discourse (interaction grounding) [22]. Previous work extracts procedural knowledge about new tasks from dialogue for small action sequences focusing on seman- tic grounding [17, 23], without investing the agent with knowledge about communication management. Scheutz et al. [23] 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 an agent 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. The agent must be able to interact strategically with an unpredictable human partner, rather than execute a predictable action sequence (procedure).

In [24], an agent learns cloth folding through rich verbal communication, based on AND-OR graphs. It can understand utterances with context dependencies common to human language but challenging for machines (e.g., descriptions of objects that evolve over several utterances). Language interaction via semantic parsing combined with deep reasoning is used in agents that explain their actions [25, 26], using existing NLP tools for parsing into a logical form [27], and a rule-based, broad-coverage toolkit for generating English from structured input [28].

Other work that relies on rich, situated reasoning through multi-modal communication is based on an architecture for collaborative problem-solving [29], with plan-based dialogue management [30]. These works either do not have distinct dialogue management modules, or rely on manually-engineered dialogue management rather than machine-learning. Our work presents machine-learned POMDP policies using a method that generalizes across different games, and across differences in dialogue partners' informativeness.

2.3 Adaptive Artificial Agents

The notion of adaptation has been used in several works on POMDP dialogue policy learning. In [31], authors address generalization to new tasks through domain adaptation. In [32], authors propose an agent that learns from a tutor, managing the trade-off between accuracy of learned meanings and the cost of tutoring through a dialogue policy that adapts its confidence in the learned meanings.

In [33], the authors design a chef robot that can operate in two modes. The robot can teach experts or novices how to make a desert and can be set up to teach by giving detailed explanations of cooking tools or teach in a high level mode. Through their experiments they show how quality of communication decreases when a novice is taught by a robot that provides concise explanations, or when an expert is given too much detail. A limitation of the work reported in [33] is that it requires prior knowledge about a person's cooking expertise.

Adaptation from simulation to real world to fill the gap between learning in simulation and applying a learned policy in the real world has been addressed in a many ways. In [34], a policy to simulate action (e.g., Cartpole) is learned offline from demonstration videos and balanced with online policy learning using temporal difference analysis. In task completion spoken dialogue systems, the Dyna-Q reinforcement learning approach to jointly learn a world model and a policy [35] has been extended to Deep Dyna-Q for large state and action spaces [36, 37].

In robotics of physical movement, the discrepancy in performance of policies that are trained in simulation and tested in the real world is a known problem, sometimes referred to as the *reality gap*. Designing realistic polices to close this gap by sensing the physical environments is investigated in several works [38, 39]. In these approaches, the agent models its current environment and is able to adapt quickly based on sensory input. Some other recent work on robots learning motor skills through simulation and using them in real-world environments learns policies for a large number of simulated environments (domain randomization [40, 41]), or performs online fine-tuning of policies trained offline [42]. Transfer of robotic motor skills to real-world environments with unknown dynamics has been addressed by learning families of related policies and then learning to linearly combine them [43], learning a classifier for environment parameters to choose the correct policy [5], or searching directly for the correct policy using the current accumulated reward to find the best of the available strategies [44].

Our adaptive sensing of information dynamics is inspired by work in the robotics of physical movement, where the POMDP policies adapts to a dynamic dialogue context by continuously sensing the changes in behavior of the interlocutor, as well as the agent's own behavior. The policy design also supports offline training where the agent aggressively explores many communicative actions with a wide range of dialogue partners to learn a policy that transfers well to online dialogue with people as well.

2.4 Reinforcement Learning of Dialogue Policies

Reinforcement learning of dialogue policies has long been used in task-oriented spoken dialogue systems [6, 45], where dialogue is modeled as a Partially Observable Markov Decision Process (POMDP), in which the next dialogue action depends on the current state, and where the uncertainty about the current state can result from errors in speech recognition or natural language understanding. Although POMDP polices provide a great foundation for representing decision making under uncertainy in observation of the world, exact policy optimization for POMDPs is tractable only for relatively small state/action problem cases [46]. However, we can present a POMDP with discrete state and observation spaces as an MDP with a continuous state space [46], which enables the MDP algorithms to be used for policy optimization.

In POMDP polices, probabilistic dialogue state tracking, or belief tracking, was introduced as a central component of spoken dialogue systems to better handle noisy speech recognition and other sources of uncertainty in understanding a speaker's intentions [47–49]. Belief state tracking in such systems helps the agent choose dialogue actions that progress towards the task goal (e.g., finding a restaurant, booking a flight ticket and so on), using task-relevant information slots (e.g., desired cuisine, neighborhood, price, and so on). Recent work on dialogue management policies show that the dialogue agent can learn to use a belief tracker's distribution over speaker intents to decide whether to clear up a misunderstanding by requesting clarification from the dialogue partner. Many different belief tracking models have been proposed in the literature, from discriminative statistical models [50] to generative [51] or rule-based methods [52].

Recent dialogue systems use analogous architectures, but with neural models for NLU and NLG [53]. Given enough data, these models can learn which lexical and semantic features are good indicators for a given slot value and can capture different ways humans convey their intentions using different words [54]. The sequence-to-sequence (seq2seq) model that was developed for language modeling and machine translation [55, 56], often combined with various attention mechanisms [57, 58], has achieved huge successes in machine translation, and has also been applied to the NLG task. In [59], the authors proposed a seq2seq neural model with LSTM cells that can generate sentences in natural language as well as dependency trees from input dialogue acts. In [60], a seq2seq base model is introduced with a modification to the RNN gates, to show improvement in a few NLG domains. Previous work has also focused on different training strategies and architectures. For example, in [61], the authors propose a neural natural language generator module for spoken dialogue systems that has a hierarchical decoding method and takes into account the linguistic patterns at different levels of the generated sentence, thereby avoiding overly lengthy or complex outputs.

To address the high cost of online policy learning over many iterations, policy learning can use a simulated dialogue partner [62]. With offline training of dialogue policies, Gaussian process POMDP learning leads to faster convergence [63], which we used in this work. Another approach learns a policy from user interaction while concurrently updating a feedback signal to adjust the probability of each state-action pair, based on user feedback [64]. Also in online training, a reward model and policy can be learned concurrently during real-world interactions, with less supervision of the reward model from user feedback as the policy improves [35]. To exploit data from different dialogue domains for transfer to a new domain, a committee of policies can be used [65], based on Bayesian Committee Machines [66].

Our work builds on a foundation of dialogue management through reinforcement learning policies for task-oriented dialogue, but relies on a more abstract belief state to represent game states and actions. However, we utilize a belief state representation that goes beyond tracking the information exchanged in a dialogue to include the agent's sensing of the ongoing dialogue behavior. The policy learns how to formulate the next question, depending on the current partner, so as to maximize its learning goal. In contrast to task-oriented dialogue systems, our policies also rely on a representation of the knowledge being acquired through communication. This is necessary for formulating the reward so as to guide the policy towards good questions, with respect to a measure of the agent's resulting knowledge gain. Our agent can apply the same policy trained in simulation to dialogues with human partners due to its sensing ability. Another novelty of our the dialogue policies developed here is that the same policy can be used in a succession of dialogues with different people, due to the way the policy utilizes the belief state and a persistent knowledge store. This is beneficial for learning a complex activity that cannot be mastered in a single dialogue. No other work we know of addresses dialogue management for achieving a goal that can be pursued through multiple dialogues.

Chapter 3 Language in Grounded Learning through Communication

This thesis presents agents that learn new strategic activity directly from interaction with people, based on visual demonstrations and text-based, natural language communication. Our agent relies on a novel application of Partially Observable Markov Decision Process (POMDP) dialogue policies: how to learn new things through efficient questioning of people. Each question is a dialogue action conditioned on the current state. POMDP policies have been used widely for task-oriented dialogue systems, but to our knowledge, they have not been applied to agents that learn through communication. The dialogue policies presented here are for learning games. Given that we model dialogue as Markov Decision Process, where the dialogue actions (communicative intents) are conditioned on the current state, a precondition of the work is to design the agent's communicative actions. This chapter deals with the underlying formal representation of communicative actions using the kind of meaning representation language (MRL) that supports natural language processing research on semantic parsing and natural language generation. We also describe the collection of a corpus that pairs MRL expressions with natural language expressions. We then use existing natural language processing techniques for modules that interpret natural language input as MRL expressions, and to generate natural language utterances from the MRL communicative actions. Use of POMDP dialogue policies supports the uncertainty of natural language understanding.

The MRL proposed is novel in several respects. First, it provides a convenient formalism to ground communication in two directions: in a representation of the physical world of game boards and pieces, and in a very abstract representation of the game knowledge the agent acquires through dialogue. Second, where much work in POMDP dialogue systems relies on fixed sets of dialogue acts, the MRL presented here supports generation of context-dependent communicative actions through compositionality of the MRL.

Games are an ideal testbed that can grow increasingly complex along multiple dimensions, from very simple two-player settings (e.g., tic-tac-toe), to highly complex multi-party interactions (e.g., bridge). Also games model fundamental characteristics that apply to all human activities: pursuit of goals in consideration of potentially unpredictable states of the environment, and in consideration of potentially unpredictable actions of other agents in the game. Prior work on learning through communication often focuses on a single task, engineers the dialogue behavior rather than learning a dialogue policy, and does not address generalization of the dialogue behavior to new learning tasks [67]. Use of game trees for knowledge representation provides multiple benefits with respect to reasoning (See Chapter 4), system evaluation, and grounding natural language expressions in a semantic world. The agent can later play a game it has learned, using the algorithms like minimax [68]. Because game trees are a formalization of game knowledge and can represent a wide range of games, we later show how the agent can learn a single dialogue policy that can generalize to different games.

The rest of the chapter is organized as follows. Section 3.1 explains language related challenges for learning through communication. In section 3.2, we give an overview of how we use extensive-form game trees to store game knowledge, and how the agent adds to its knowledge by asking questions of dialogue partner. Next in section 3.4, we describe the meaning representation language (MRL) in detail. As we explain below, the MRL is compositional in the sense that the agent constructs questions using question operators combined with logical forms that reference specific characteristics of the physical artifacts of the game, or actions that can be taken in the game. In this way, the agent can formulate new questions related to aspects of game actions or game states. We explain how we developed a novel data collection method to gather data for natural language modules that convert the MRL to natural language (NL) and vice versa, in section 3.5. We conclude the chapter by describing the natural language understanding (NLU) and natural language generation modules (NLG) that enable the agent to communicate with humans in natural language.

3.1 Challenges in Grounded Language Communication

In human-human interaction, we use words to convey our intentions for many different things: requesting information, describing an action or an object, warning or persuading someone, and so on. When we hear words, we interpret them, understand their underlying meanings, and integrate them in our internal representation. Artificial agents that learn through language communication must also have a way to relate language to the environment, and a way to build a representation of it [69, 70]. In other words, their understanding of the physical world and how it works must be grounded [71]. However, there are challenges for grounded learning through communication. There is a many-to-many connection between the linguistic tokens we use and their underlying meanings as humans communicate their intentions in many different ways [72], and language itself is inherently ambiguous.

Language grounding has a long history in the field of artificial intelligence. From designing hard coded and physical rules [71], to designing machine-learned classifiers to ground words and phrases [73], to end-to-end neural models [21]. This work relies on a meaning representation language (MRL) that allows the agent to learn using an unambiguous language that represents the underlying intention of the questions it asks and the information a dialogue partner provides in response. As well as providing a way for the agent to relate language to objects and action, the MRL allows the agent to ground the game knowledge it gradually acquires in a dialogue, directly to its game tree (See next section.). Our proposed MRL is general, can be used for a family of games, and allows the agent to form new communicative actions on-the-fly given a context by using a logic function from that context to specific actions. Later in section 3.5, we show how the MRL can be used with natural language understating and natural language generation modules to enable the agent to communicate with humans.

3.2 Game Knowledge as an Extensive-form Tree

To represent and store the game knowledge, we use extensive-form game trees. Using game tree as a knowledge store has many benefits: 1) given that we use it as a persistent knowledge store, after the agent has a dialogue, it can use algorithms like minimax [68] to play with a person. This provides a way to test the quality of the knowledge the agent has learned by testing how often it can win a game. 2) As we will show in Chapter 5, it provides an abstract knowledge representation that supports policy learning across different games, including new games not seen during training. 3) The agent can reason about its existing knowledge in the game tree and infer new win conditions similar to the ones it has learned, using a similarity metric we will explain in the next chapter.

Game theory offers computational representations that have been used to formally represent and reason about a number of games [74, 75]. The normal-form game and the extensive-form game are computational representations from game theory that serve as building blocks to represent a complete game. Sequential stages of an interactive game are represented in extensive-form as a tree. Nodes in the tree represents the state of the game and the edges are the actions that can be taken be the players. The leaves of the tree are the terminal states of the game. For the zero-sum games that our agent learns, terminal states can be loss, win, or draw. For example, Figure 3.1 shows an extensive-form game representation for a portion of the Connect Four game. The upper node represent the state after Player 1 took an action and the lower nodes represent the game state after Player2 selects an action from seven actions denoted 0-6 for the column in which the disc is placed. At each step 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.

Our agent has two learning goals: to learn about game moves and to learn about win conditions. By learning game moves, the agent learns what actions (tree arcs) are available at each state of the game (tree nodes) and by learning about win conditions, the agent learns what are the state of the tree leaves. At the start of a dialogue, the agent has generic knowledge about games in the form of a start state G_0, meaning it knows that it is going to learn a zero-sum two player games, where players take turns in taking actions. It also has a place-holder for game states $\gamma \in \Gamma$ that map to game pieces placed on a grid game board, another place-holder for the actions (moves) $\mu \in M$, a procedure to build a `successor` function that defines the successor states γ_{i+1} given a move action μ taken in a state γ_i, and the notion of a terminal state (tree leaf) in which the agent

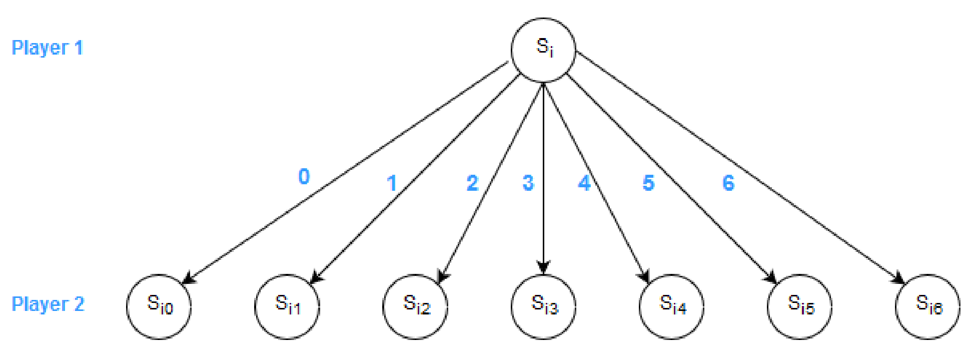


Figure 3.1: Extensive-form game representation of Connect Four, showing the possible actions (0-6) Player 2 can take to make one game move. The upper node is the game state S_i, after Player 1 chose an action. The lower nodes represent the possible game states $S_{i0} - S_{i6}$, when one of the seven actions (0-6) is chosen by Player 2.

wins, loses or the game is a draw. Then, the game tree is incrementally extended based on answers to the agent's questions (See Chapter 5 for details.). Using the MRL, the agent learns about a game by asking visual or verbal questions about game moves or win conditions (a win condition in the tree is a path to a win in which the other player's actions are left unspecified.), and uses the responses to update the game tree.

To summarize, we use game trees for grounding natural language expressions in an abstract representation of game states and actions. The next section presents the MRL and how MRL expressions are grounded in representations of objects and actions for specific games, and in the associated game trees. The communicative actions are questions about what kinds of actions are taken in the specific game under discussion, and questions about arrangements of n pieces that constitute ways to win. Later chapters return to game trees and how they support the design of a reward for reinforcement learning of a dialogue policy, reasoning over the game tree to draw inferences about what to ask, and generalization of a single dialogue policy across different games.

3.3 Board Games

Here we introduce the specific games we focus on in this work. Our agent can learn two-player zero-sum board games where each player tries to place n pieces in a straight line. The board games we focus on here are Connect Four, Gobblet, Gomoku, Quarto, and Tic-Tac-Toe, but the methodology generalizes to a wide range of n-in-a-row board games. In Chapters 4 and 5, we show how we train and test the agent for offline learning of a dialogue policy to learn Connect Four, Gobblet and Quarto from human dialogue partners. Later in Chapter 6, we test the learning-through-communication skills of the agent with respect to games not seen in training, and show how an agent that is trained for Connect Four, Gobblet and Quarto, can learn the unseen games Tic-Tac-Toe and Gomoku.

Here we provide a detailed explanation of each of these five games:

- Connect Four has a vertical game board of seven columns with slots for six game discs per column (a board size of $6 \times 7 = 42$). Since the board is vertical, at each node a player has at most seven possible actions. Once an action is taken, it cannot be taken again. A player wins when four of her discs are adjacent in a column, row, or diagonal. After the first player makes a move, the second player can choose one column out of seven, continuing from the first player's choice. If both players choose the same column six times in total, that column is no longer available for either player. As a result, the branches for possible game actions are reduced by one. Also, when both players make all choices (42 game moves) and there are still no 4 discs in a row, the game ends as a draw in a terminal state (game tree leaf). Finally, if any player makes 4 in a row, the game tree path ends in a terminal win state, and the game ends.

- Gobblet is played on a 4×4 horizontal board, where each player has four sizes of game pieces, three of each size. At each stage there are a maximum of ($4\ rows \times 4\ cols \times 4\ sizes = 64$) possible actions where each action is a combination of the board coordinate and piece size. When the game begins, the first player gets to choose one board position among 16 to place a colored disc of a certain size. Each player has 12 pieces of the same colour but different sizes that can nest on top of one another to create three stacks of four pieces. Therefore, there are ($4\ sizes \times 16\ places = 64$) branches in the game tree at the root node. After the first player makes a move, the second player can choose one out of 15 remaining positions on the board, continuing from the first player's choice of the game tree, or pick a piece that is larger than the one the first player picked (if any), and cover (or gobble) the first player's piece. The game tree continues to grow with a maximum of 64 actions from each state until a win, loss, or draw terminal state is reached.

- Gomoku is played with black and white game pieces on a Go board, with each player choosing a color. It can be played using the 15×15 board or the 19×19 board. Players take turns placing a piece of their color on an empty intersection, with black playing first. The winner is the first player to form an unbroken chain of five pieces horizontally, vertically, or diagonally. Gomoku has a large tree because it is played on a large board. When the game begins, black has 225 available actions, given a 15×15 board, or 361 available actions, given a 19×19 board. The second player can choose one out of the 224 or 360 remaining positrons on the board. Similar to Connect Four, at each state of the game, the branches for action choices are reduced by one, and the game tree terminates when the board is filled (draw), or a win or loss is reached.

- Quarto has a 4×4 board and 16 game pieces, distinguished into two colors, two heights, two shapes, and whether they are solid or hollow. In each turn, the opponent identifies a piece for the current player to place on the board. Four in a row wins if there is a property shared by all four pieces. At the start of the game, the second player picks one out of the 16 distinct pieces for the first player to use. The first player has 16 positions to put the given piece. After the first player makes a move, it picks one out of at most 16 distinct pieces for the second player. The second player chooses one position out of the remaining 15 positions on the board to place the piece. Therefore, at each state of the game there are at most 16 available actions. Similar to the above games, the game ends when a draw, loss, or a win is reached.

- Tic-Tac-Toe is played on a 3×3 grid where two players, X and O, take turns marking the spaces in the grid. The player who succeeds in placing three of their marks in a diagonal, horizontal, or vertical row is the winner. Compared to the other four games tested here, Tic-Tac-Toe has a smaller game tree. At the start of the game, the first player has 9 actions available. Following the first player, the second player has 8 actions available since a marked position cannot be used again. As the game tree grows the number of actions at each state reduces by one, until a draw, win or loss is achieved.

3.4 Meaning Representation Language

This section explains the types of MRL questions that can be generated, and the types of MRL responses the agent can understand. When a dialogue about a game begins, the agent can ask about individual moves or game pieces and about ways to win, where a win condition is a sequence of n pieces in a straight line, possibly with constraints on the shape, height, color, etc. As discussed in the previous section, the agent has generic knowledge about two players games at the start of the dialogue. Using communicative actions that ask about game moves or pieces, the agent learns the specific set of actions available in a particular game, and therefore, how to add actions and states to a specific game tree. Using communicative actions that ask about win conditions, the agent learns paths from the root to a leaf in the game tree. As we see later in the thesis, the agent learns a policy in which it can interleave both kinds of questions.

The dialogues are multi-modal: the agent can ask the interlocutor to demonstrate a game move or a way to win, or ask for verbal responses. They are structured as sequences of sub-dialogues, where each sub-dialogue starts with a visual demonstration of a game board. The communicative action generator takes as input the current context (i.e. current board demonstration) and the communicative action type selected by the dialogue policy, and generates a specific communicative action for the agent in a meaning

Communicative Action Operators	
Action Type	**Meaning**
Global Policy	
Req_GDem()*	Request the demonstration of a new game move.
Req_WDem()	Request the demonstration of a new win condition.
GameP(i)*	Resume discussion of game moves with reference to the i-th demonstration D.
WinC(i)	Resume discussion of win conditions with reference to the i-th demonstration D.
Start(), Finish()	Greetings and Goodbyes.
Local Policy	
Conf(ShiftBoard)	Is demonstration D still a win after ShiftBoard? (ShiftBoard can be Rotate or Translate.)
Conf(Property())	Does Property() cause demonstration D to be a win? (Property can be shape, color, height, hollowness, or size.)
Conf(ChangeDisk())	Is demonstration D still a win after ChangeDisks? (ChangeDisk can be RemoveDisk or AddDisk.)
Req(ShiftBoard())	What ShiftBoard operations on demonstration D will also be a way to win?
Req_Other()	Can the other player undo D?
Req(Location())*	What locations other than demonstration D are legal moves?
Conf(Location())*	Is Location() also valid for the game piece shown in demonstration D?
Req(Piece())*	What other game pieces can be played instead of the one shown in demonstration D?
Conf(Piece())*	Can Piece() be used instead of the game piece used in demonstration D?

Partner's Communicative Actions	
Inform()	Some or all of the requested information has been provided.
Affirm()	Positive answer to a yes/no question.
Negate()	Negative answer to a yes/no question.
Unknown()	Non-answer to a question.
Start(), Finish()	Greetings and Goodbyes.

Table 3.1: Communicative action operators for the agent and the interlocutor. Asterisks mark those used to learn about game moves.

representation language we describe here. The policy action types are divided into global policy action types and local policy action types and shown on Table 3.1. The predicates that are used in some of these communicative actions to build a complete question are shown in Table 3.2. The communicative actions are to mirror the hierarchical structure of the policy. With global actions, the agent decides whether to resume conversation about the current context (i.e. current board demonstration), switch to a previous context i, (e,g, a win condition demonstration (`WinC(i)`), a game move demonstration (`GameP(i)`)), or prompt the dialogue partner for a new demonstration (e.g. a new win condition (`Req_WDem()`), or a new game move demonstration (`Req_GDem()`)). Figure 3.2 shows two turn exchange for the Connect Four that is sampled from a trained policy. We show the question-answer pairs both in MRL and in natural language for clarity. The agent first decides to talk about demonstration D_1 by choosing `WinC(1)` which is translated to *"Let us talk about the win condition you showed a minute ago."* Once the agent chooses a context to discuss, it can generate specific questions about the chosen context with action types introduced under the Local Policy header in Table 3.1. In the first question in Figure 3.2, the agent asks if it can reconfigure the board by moving the disks and still win the game. (The global policy is discussed in detail in Chapter 4.)

The agent can ask *"yes/no"* questions (e.g. `Conf(Property())`), or ask *"wh-"* type questions (e.g. `Req(ShiftBoard())`). There are 11 (local) *"yes/no"* question types (`Conf()`), nine about win conditions and two about grounding. The agent can generate 5 *"wh-"* question types (`Ref()`) (e.g., about locations of game pieces, or their properties, such as shape and color): two about win conditions, and the rest about grounding. The *"wh-"* questions are open-ended questions and prompt the dialogue partners to provide as much information as they can. The *"yes/no"* question types are more numerous because they request confirmation of very specific categories of information. Later in Chapter 4, we will show how agent uses these two types of questions to adapt to partners that differ in how much information they volunteer. We will show that for example, when the

Communicative Action Predicates	
Predicate	**Range**
Game Piece	
Shape*	Round, Square
Height*	Tall, Short
Color*	Red, Blue
Hollowness*	Hollow, Solid
Size$^+$	Small, Medium1, Medium2, Large
Game Board	
Columns	$\{c \mid c \in \#BoardColumns\}$
Rows	$\{r \mid r \in \#BoardRows\}$
Angles	$\{0, 45, 90, 135\}$
Board Position	$\{(x, y) \mid x \in \#BoardRows, y \in \#BoardColumns\}$
	(e.g. 42 places for Connect Four, 16 places for Quarto, etc.)

Table 3.2: The Figure shows the predicates that can be used with the communicative actions presented in Table 3.1, to formulate a complete question. Asterisks mark those used in Quarto. and plus signs mark those used in Gobblet.

((a)) D_1

- **$\mathbf{Q}_1$)** Agent: *WinC(1), Req(translate(), {implicit arg = D_1})*
  (NL: Let us talk about the win condition you showed a minute ago. Where else can these disks be placed to be a win?)
- **$\mathbf{A}_1$)** Dialogue Partner: *Inform([0,2])*
  (NL: One column to the left or right.)

- **$\mathbf{Q}_2$)** Agent: *WinC(1), Conf(translate(3),{implicit arg = D_1})*
  (NL: Can it shift two columns over?)
- **$\mathbf{A}_2$)** Dialogue Partner: *Affirm()*
  (NL: yes.)

Figure 3.2: An example of two turn exchanges for Connect Four extracted from a dialogue between a trained agent and a simulator. We show both the MRL and the natural language for each question and answer. Both turn exchanges are about the win condition shown on the board.

partner is less informative, the agent asks more *"yes/no"* questions than *"wh-"* questions.

The action types can be viewed as functions that return a complete MRL as a value. If no argument is shown, the current board D_i is the implicit argument. `Conf()` and `Req()` can be used to ask questions about actions that can be taken on the current board (e.g. `ShiftBoard`), about properties of the game pieces (`Property`), about the locations game pieces can be put on in a game move (`Location`), or about pieces that can be used in place of the ones shown on the board (`Piece`). `Location` can be realized as specific board positions, and `Piece` can be realized to point to a specific game piece on the board using the game piece properties presented in Table 3.2. The last two questions are about game moves as marked by asterisk in Table 3.1. There are two `ShiftBoard` realizations. `Translate` can be used with `Conf()` to generate *"yes/no"* questions about a shift of the current demonstrated win condition by a valid offset value α, where the offset format is shown as `Board Position` in Table 3.2. It can also be used with `Req()` and no offset value to generate *"wh-"* questions to prompt the user for other locations on the board the demonstrated win condition can be translated to. Both uses of `Translate` are illustrated in Figure 3.2. The second `ShiftBoard` realization, `Rotate`, is for analogous *"yes/no"* or *"wh"* questions about rotations of D_i by a valid angle θ (See Table 3.2). `Property` is for asking whether observed properties of game pieces contribute to a demonstrated win condition, such as color or shape for Quarto. `Property` can be realized as `Color`, `Shape`, `Height`, `Size`, or `Hollowness` to represent different properties a game piece can have. To see the actual values each of these properties can take refer to Table 3.2.

The kinds of answers that the agent currently understands are shown at the bottom of Table 3.1. A *wh-* question elicits an `Inform()` act, and a *yes/no* question elicits a positive (`Affirm()`) or negative (`Negate()`) answer, or (`Unknown()`). Here we assume dialogue partners will be truthful, but may not always know the answers to questions, and may provide incomplete answers.

Figure 3.2 also illustrates a board demonstration D_1 for Connect Four with a horizontal sequence of four disks starting in position 1. The turn exchanges reference the same context. The first question shows the MRL for a *wh-* question asking for other board positions that can be the start position, if the row is moved up, down, right or left (`Translate`). The dialogue partner responds by providing only two other locations on the board where the four pieces in the row can be placed (`Inform[0,2]`), where 0 and 2 are two `Board Positions` (See Table 3.2) Concurrently, the agent also adds this new game knowledge information to its game tree which is not shown here for the purpose of simplification. In the second turn exchange, the policy generates a question to confirm if the game pieces can be moved to a specific location and it still be a win conditions, and the user provides a positive answer (`Affirm()`). At the end of each turn exchange the agent's belief state is updated to include the new acquired information. The belief state has several components that corresponds to different communicative actions of the agent. For example, the information that is relevant to ways a board can get reconfigured by moving the pieces can be learned with communicative actions `Conf(Translate)` or `Req(Translate)`, and update a belief component named (`Translate`) of length 42 where each vector element is a position on the board. If the question is for example about whether it is the color of game pieces that contributes to the win, then a belief vector that has vector elements corresponding to different colors `(Red, Blue)` will get updated. This is explained in detail in the next chapter.

3.5 Dialogue Data Collection

The agent can communicate using the meaning representation language discussed in previous section. To add natural language capability for the agent, we developed a novel data collection method to produce a corpus consisting of <Game-board,MRL,NL> tuples

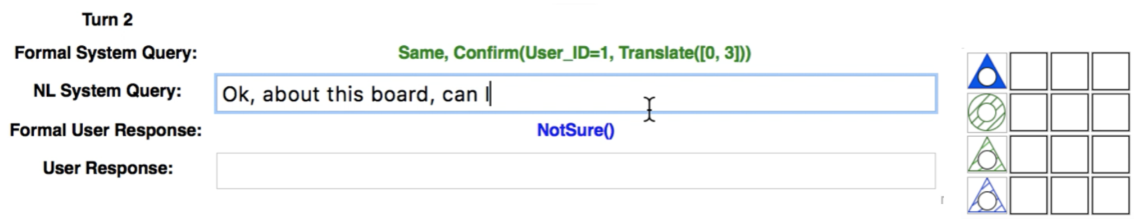


Figure 3.3: Data collection GUI for turn exchange 2 of a dialogue. The annotators are presented with the question-answer pairs in meaning representation and are asked to provide the corresponding natural language. The Figure shows half of the question typed in English for the corresponding MRL.

for each utterance in 960 dialogues between an agent and simulated dialogue partner (the simulator, which uses a database of responses, is described in Chapter 4). The corpus is collected for the game Quarto, as it is one of most complex games our agent learns. To collect the corpus, we first train a dialogue policy using a simulator, where the simulator communicates with the agent using the MRL. Once the dialogue policy is trained, we can generate dialogues, where both agent's utterances and the dialogue partner's responses are in the MRL. We trained annotators to translate all dialogues into English. A sample of a translated dialogue is presented later in the section. As we will explain in the next section, we use this data to train machine learned natural language models to convert MRL to natural language and vice versa.

To our knowledge, this is the first corpus of its kind. Most previous dialogue corpora we know of fall into one of three other categories: human-*Wizard-of-Oz*, human-agent, or human-human. Corpora for human-*Wizard-of-Oz* are used either to inform manually engineered dialogue management or as training data for machine learned dialogue managers. These corpora are collected for the purpose of restaurant reservation [76], finding available vacation accommodations [77], or even open-domain information retrieval systems [78]. Human-agent corpora are often annotated with dialogue acts for applications such as travel booking systems [79]. Human-human corpora are either collected under constrained settings where humans are instructed to follow a series of instructions [80, 81], or are naturally occurring conversations between humans [82–84]. Distinctive characteristics of the Quarto corpus are that every utterance has an MRL and a natural language version where the MRL is a communicative act. The dialogues involve a shared multi-modal context, leading to deictic reference to the game board and with a known structure into sub-dialogues.

To collect our corpus, we developed two graphical user interfaces (GUIs) to display a

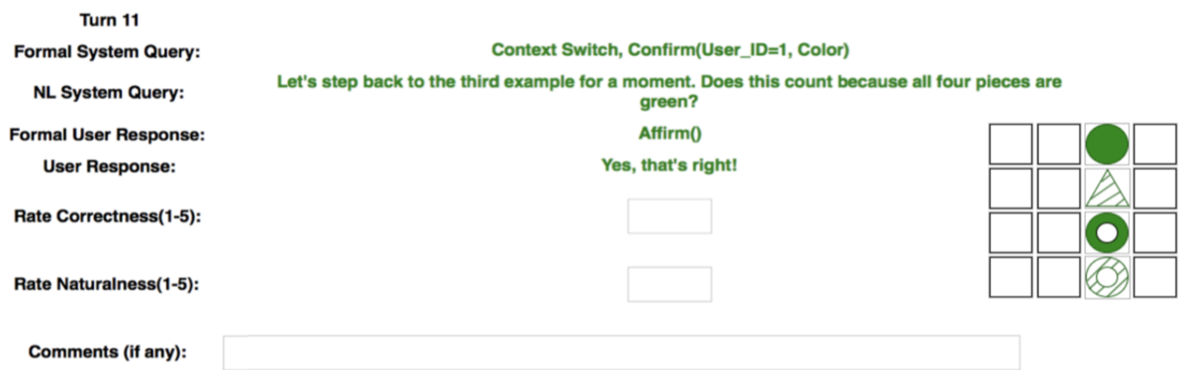


Figure 3.4: Evaluation GUI for turn exchange 11 out of 14 in the whole dialogue. After providing the natural language translations, annotators are asked to rate each others dialogues based on correctness and naturalness. A comment entry at the bottom of each turn, allows the annotators to provide any additional comments they might have.

schematic representation of the current board demonstration (see Figure 3.3), and to allow annotators to page through each turn exchange. One GUI was for the translation task, and a second was to collect ratings on the translations. The example in Figure 3.4 illustrates the evaluation entry boxes for turn exchange 11 of a dialogue, with a Formal System Query followed by a translation to English and a Simulated User Response followed by its translation to English. Note that this turn exchange involves a context switch to the Quarto board shown in the GUI. Each board is depicted in the GUI as a 4×4 grid with four pieces. In Quarto, the 16 game pieces have one of the two values of color, height, shape and hollowness, totalling four pieces that have the same of each property. In the GUI, these are shown as blue vs. green, solid vs. striped, circles vs. triangles, and with an empty center or not.

Thirteen undergraduate students from a course in artificial intelligence participated as part of their course work. Students were first trained in the MRL, including comparisons with the first order logic translations of English that students had learned in class. Their instructions were to translate into colloquial English. Meetings were held where students discussed examples and asked questions. All translations were rated for correctness and naturalness on a five-point scale where 5 was the top. On average, correctness was 4.79 and naturalness was 4.72. Below we show a sample dialogue from our corpus where we present the turn exchanges in meaning representation along with their natural language translations.

- **Q1)** `Start()`
  NL: Hi there! Can you teach me how to play Quarto?

A1) `Start()`

NL: Sure thing, let's get started.

- **Q2)** `Req_WDem()`

  NL: Can you show me an example of a win?

  A2) `Inform(board=`*`visual_representation`*`)`

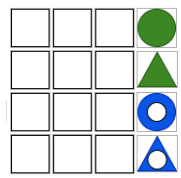


Figure 3.5: First Win Condition (D_1)

- **Q3)** `Req_WDem()`

  NL: Alright, how about another?

  A3) `Inform(board=`*`visual_representation`*`)`

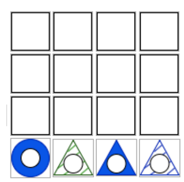


Figure 3.6: Second Win Condition (D_2)

- **Q4)** `WinC(`D_1`), Conf(Translate(col0))`

  NL: In the first example you showed me, would it still count if the pieces were placed down the first column instead of down the fourth?

  A4) `Affirm()`

  NL: Yeah, it would.

- **Q5)** `WinC(`D_1`), Req(Rotate())`

  NL: Say I keep the blue triangle in the bottom-right corner, but try placing the remaining pieces along some other direction upward, to the left, or a combination of both. Could I still get a win?

  A5) `Inform([135, 180]`

  NL: You could get a win so long as you placed the remaining pieces along the anti-diagonal, or leftward along the bottom row.

- **Q6)** `WinC(`D_1`), Conf(Shape)`

  NL: Does this count because of the shape of the pieces?

  A6) `Negate()`

  NL: No, this win is not based on the shapes.

- **Q7)** `WinC(`D_1`), Conf(Quantity)`

  NL: Well, does this count because there are four and only four pieces?

  A7) `Affirm()`

  NL: Yes, it wouldn't count for any other number of pieces.

- **Q8)** `WinC(`D_2`), Conf(RemoveDisk((0,1)`

  NL: Okay, in the second example you showed me, what would happen if I removed the green triangle?

  A8) `Negate()`

  NL: Then you would no longer have a win.

- **Q9)** `Req_WDem()`

  NL: Can you show me another example of a win?

  A9) `Inform(board=`*`visual_representation`*`)`

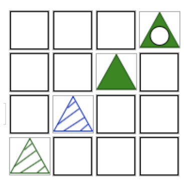


Figure 3.7: Third Win Condition (D_3)

- **Q10)** `Finish()`

  NL: Great, thanks for helping me get started!

  A10) `Finish()`

  NL: I'm happy I could help!

The 960 dialogues contain 12,885 turn exchanges. The English translations contain 229,641 word tokens, and 1,498 word types. The NLG data has 146,055 tokens and 1,102 types. The NLU data is somewhat less rich, with 83,586 tokens and 952 types. As we will discuss in detail in Chapter 4, we have developed a simulator that can be configured to different levels of informativeness ranging from 0% to 100%. Our corpus covers dialogues with different informativeness. The 960 dialogues consist of 535 from a 60% informative simulator, 255 from a 100% informative simulator, and 170 from a 50% simulator.

Because all turn exchanges are tied to a physical board, the corpus is rich in spatial references, similar to the ones shown in Figure 3.2 (i.e. a question about board D_1). The students referred to the pieces by specific attributes (e.g. *next to that green circular piece*), exact location on the board (e.g. *top corner piece*), relation with other pieces (e.g. *to the right of the square piece*), or deictic reference (e.g. *this piece here*). There are also many anaphoric references (e.g. *about that win you showed*, *the second win*).

3.5.1 Natural Language Modules

To enable the agent to have a dialogue with humans in natural language, we train a natural language generation (NLG) model to convert MR question actions to NL. We also train a natural language understanding (NLU) model to translate NL responses to MR. To train the machine learned NLG and NLU models, we used the dialogue corpus discussed in previous section. Although the dialogue corpus is specifically gathered for Quarto, we were able to apply the same corpus to all our games and actions as the MRL is general enough to be used for other games as well. Both natural language models are sequence-to-sequence LSTMs with two hidden layers and Bahdanau attention [57], trained using the Adam optimizer [85] for 15 epochs. We measured NL quality on the 32 human-agent dialogues discussed in Chapter 6. We rated the NLG for correctness and intelligibility on a five-point scale, where 5 is the best. The average NLG score was 4.6±0.1. The NLU accuracy was 82±8.9%.

We assume answers provided to the agent are truthful. To ensure the agent does not receive invalid game knowledge due to inaccurate NLU (e.g. a location on the board where four in a column cannot be placed), we always perform an automated verification on NLU output, information, both in training with the simulator and in testing with people. For *"yes/no"* questions, any invalid NLU prediction is converted to the `Unknown()` answer. In list answers to *"wh-"* questions, we remove any items in the response list that are inconsistent.

3.6 Summary of MRL for Grounded Learning through Communication

This chapter presented a novel MRL for dialogues in which an agent learns games from people, and a novel utilization of extensive form game trees to ground the MRL in game tree knowledge. In the following chapter, we show how the MRL supports the agent's ability to generate context-dependent communicative actions using an MDP dialogue policy. As we discuss in this next chapter, the MDP models how the agent's communicative actions are conditioned on the agent's belief states after each turn, given its interpretations of the answers it receives to its questions. We will see how the belief state structure supports the grounding of questions in game tree knowledge, and how the reward for reinforcement learning of the dialogue policy is defined in terms of the current state of the agent's game tree knowledge. We will also see how the MDP policies differ, depending on the informativeness of the dialogue partner's answers.

Chapter 4
Dialogue Policies for Learning through Communication

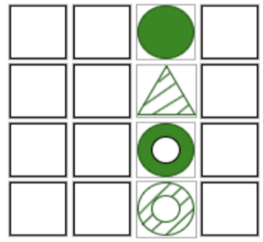
The previous chapter presented an MRL for communicative actions consisting of questions from an agent and responses from an agent or human. We show in this chapter that through reinforcement learning, the agent learns a dialogue policy to generate its communicative actions, conditioned on the current state, so as to maximize its cumulative reward over time. That is, as in much previous work on task-oriented dialogue, we cast dialogue as a Markov Decision Process (MDP). However, our agent's dialogue goals are to learn board games, rather than to complete a task. To our knowledge, this thesis is the first work to train dialogue policies for learning through communication. Although our ultimate goal is to train POMDP policies, which we turn to in Chapter 5, here we restrict our attention to MDP policies. Our specific objectives here are to show how we adapt the framework of MDP dialogue policies to handle learning through communication using a simulated partner, and that this new type of MDP dialogue policy has different behavior, depending on the simulated partner's informativeness during training. Specifically, we show that the agent will learn more if it chooses to ask questions that are sensitive to the information quality of the answers it gets.

Our first objective here is to show how we design and train MDP policies using the MRL, meaning without natural language during training. We test the trained policies with simulated dialogue partners that use MRL in experiments that show how the policies perform best when the policy matches the informative of the simulated partner. To motivate the POMDP policies we present later, we also test the MDP policies with human dialogue partners. In the dialogues with human partners, we use the NLU and NLG modules described in the previous chapter, but only at test time, not during training of the dialogue policies. The MDP policies perform surprisingly well with people, but not as well as the POMDP policies we present later. As the policy goal is to learn a game by asking for board demonstrations and posing verbal questions, the main novelties of our new MDP policies are the reward function used in the reinforcement learning to guide the agent to choose better communicative actions, a hierarchical dialogue policy to support context switching, and the agent's ability to generate context-dependent questions.

The second objective of this chapter draws on insights from the pragmatics of language, meaning a perspective on language as action. In dialogue, each utterance is a communicative action. Early work on language as action posited that utterances should be informative, truthful, relevant, and clear [86]. Here we assume that the agent's partners will give truthful, relevant and clear responses, but make no assumptions about how informative they will be. This is an important criterion for an agent that will learn through communication, because different people have different levels of expertise in the activity the agent seeks to learn, and different levels of expertise in articulating informative responses. Part of the measure of the performance of a learned dialogue policy will depend on how well the policy can be trained to construct good questions, meaning questions that maximize its ability to learn about a new game in one dialogue. We will show in this chapter that what counts as a good question depends on what kinds of answers the agent receives, with respect to whether information is provided at all, and how much information is provided. When we control the simulator's answers to provide different amounts of information, the trained policies differ (different communicative actions are chosen). We show that the agent learns more during a dialogue given a policy that is trained to match the informativeness of the dialogue partner. We also present evidence that humans are not always equally informative.

We investigate reinforcement learning of dialogue policies offline, which makes it easy to produce and compare policies trained with a simulator. Complete and correct answers to the agent's questions can be stored in a database for the simulator to retrieve. The novel aspect of our simulator is that we can configure it to have an informativeness range from 0% to 100%. As an example, Figure 4.1, shows how a simulated dialogue partner provides an uninformative answer (`Unknown()`) to the question. An informative answer here would be "yes" (`Affirm()`) or "no" (`Negate()`. Responses to open-ended questions can be fully informative, uninformative, or somewhere between the two. For example, in a game like Connect Four with seven columns, the agent can ask what other columns the discs could be placed and still win, and the answer could be any subset of the remaining six columns.

In this chapter, as the focus is on the impact of informativeness on dialogue polices, we



((a)) D_3 Board

Question

– **MRL:** `WinC(3), `*`Conf`*`(Color)`
  NL: Let's step back to the third example for a moment. Does this count because all four pieces are green?

Answer

– **MRL:** `Unknown()`
  NL: I'm not sure about that!

Figure 4.1: An example of a turn exchange where the agent refers back to board D_3, by switching the context, and the dialogue partner provides an uninformative answer.

simplify the learning goal and assume that the agent knows the specific game actions, but does not know the win conditions. Therefore, the agent only uses MRL communicative action types that are about win conditions. Further, the agent learns a different dialogue policy for each game it seeks to learn. Later in Chapter 5, we remove this simplifying assumption and introduce a single dialogue policy for learning different games, and for learning from people with different informativeness.

The sections are organized as follows. We provide a detailed explanation of how we represent the dialogue as a Markov Decision Process (MDP) in section 4.1. To engage in a game-learning dialogue, a Markov Decision Process policy π chooses the agent's dialogue actions, meaning an action a_t at time t depends on the current state s_t, which is fully observable. Reinforcement learning finds an optimal policy π to choose communicative actions that will maximize the expected total reward over time. Then we present three sets of experiments in sections 4.2, 4.3, and 4.4. In section 4.2, we compare the policies for different games and different levels of informativeness of simulated dialogue partners. Results show how policies differ across games, and for different dialogue partners. For example, the agent asks more *"wh-"* questions when the dialogue partner is more forthcoming, and more *"yes/no"* questions when the dialogue partner is withholding. As a measure of policy success, we report results as how much of a complete game tree the

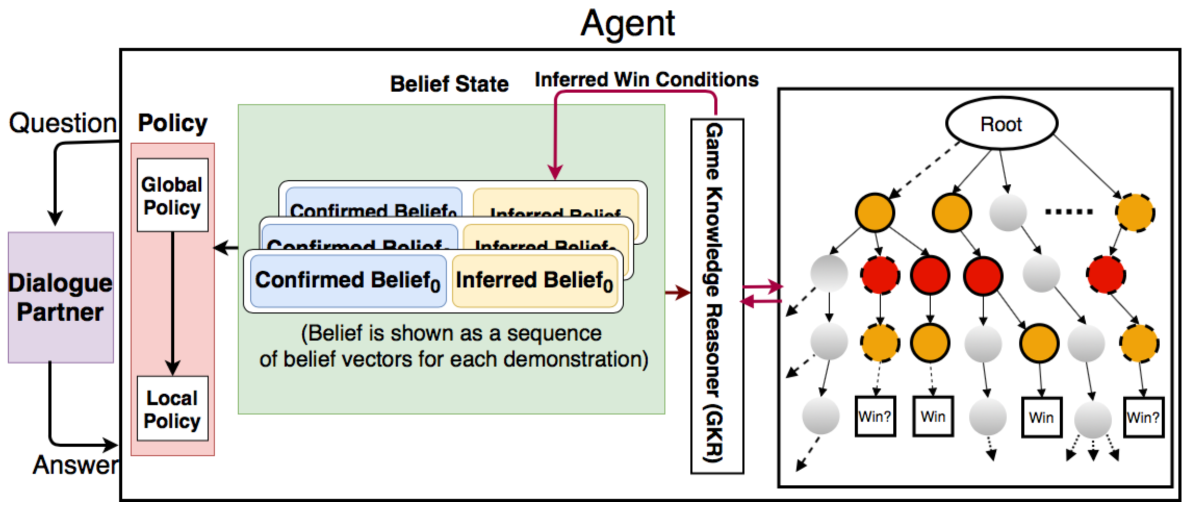


Figure 4.2: System architecture that grounds communicative actions and learning goals in game tree knowledge. The game knowledge module (far right) represents evolving game knowledge as a game tree. A game knowledge reasoner (GKR) applies a metric to assess the strategic value of new beliefs about a game that the user has confirmed (blue), and infers new ways to win that are added to belief vectors as inferred beliefs (yellow). Hierarchical global and local dialogue policies support context switching: when the agent asks the dialogue partner to provide a new demonstration of a winning board configuration, a new local belief vector is initialized to represent this win condition, and new questions can be formulated that reference the local belief vector. After the dialogue partner answers each next question, the belief is updated. Dialogues with simulators use a meaning representation language.

agent learns. The experiment in section 4.3 presents an alternative measure of policy success in which the agent plays Connect Four against humans using the knowledge it acquired from dialogues. In section 4.4, we present an experiment with human dialogue partners to show that the agent can have successful dialogues with people, and that people do vary in their informativeness.

4.1 Dialogue as Markov Decision Process

We model the dialogue as a Markov Decision Process (MDP). MDP formalizes decision making as a tuple $< S, A, T, R, s_0 >$, where S is the set of **states**, A the set of **actions**, T the transition model that specifies a probability distribution over successor states s_{i+1} given an action a_j taken in s_i, a reward function R, and the initial state s_0. An MDP dialogue agent's choice of communicative action is handled by a learned **policy** π that maps dialogue states to actions to maximize the discounted, cumulative reward $R_t = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t \; r_t]$ over time (presented in section 4.1.2). To train the policy, we rely on a **simulator** where we randomize the amount of information provided by simulator responses. We also use the simulator to test the fully trained policy under controlled conditions of informativeness. In the remainder of this subsection, we provide general background on our MDP framework.

The **policy** π is hierarchical, which enables the agent to switch contexts between different board demonstrations. The global policy determines when to ask questions about the current board demonstration, ask to see a new board demonstration, or resume discussion of a previous one. Game board images are stored in a database, so the agent can re-display them to the dialogue partner when needed. For example, in Figure 4.1, the agent refers to the column win, by switching the context to demonstration D_3 (`WinC(3)`). The local policy triggers questions about the win conditions. The policy is trained using reinforcement learning which is discussed later in the chapter.

Belief state is specific to each dialogue, and is updated after each question-answer pair. New information that was presented visually or verbally updates the belief state, so that the agent can track its progress towards its goal of acquiring new win conditions. The belief state has two types of knowledge about win conditions: inferred and confirmed, shown in blue and yellow in Figure 4.2. The confirmed beliefs are the win conditions that are directly communicated to the agent by the dialogue partner. The agent can also infer beliefs about win conditions, where it uses a Game Knowledge Reasoner (GKR), as explained in section 4.1.1. The agent can choose to ask for its inferred beliefs to be confirmed by the partner; if confirmed, they become part of the confirmed belief state. The belief state represents beliefs about what was said or inferred, and is therefore re-initialized for each dialogue. It is not the knowledge store. Both components of the belief state update the agent's persistent knowledge (game tree), as detailed in section 4.1.1.

Actions consist of the communicative actions of the agent, generated on-the-fly as MRL from a set of question operators, which are functions over contexts. The full set of question operators were presented in the previous chapter (cf. Table 3.1). Again, they break down into those that pertain to the global versus local policy. Each communicative action a is a function from an action type, where the argument is the current board demonstration under discussion. For example, the operator `Req(ShiftBoard)` is chosen by the local policy, and can be instantiated as a question about whether an observed or discussed win condition can be shifted up or down (`Translate()`), or rotated on its axis (`Rotate()`), and still be a win.

In Figure 4.2, the **Game Knowledge Reasoner (GKR)** is an interface between the agent's belief state and its persistent knowledge, stored as a game tree as discussed in the previous chapter. The GKR has two roles: First, it converts win conditions learned through dialogues into paths in the game tree. A measure of how complete the game tree has become is part of the reward structure for the reinforcement learning of the dialogue policy. Second, the GKR draws inferences about new ways to win that are added to the agent's belief state as unconfirmed beliefs. The policy can lead the agent to request confirmation of unconfirmed beliefs; if confirmed, these will also be added to the game tree by the GKR.

The **Simulator** has different sets of MRL responses for each of the question categories (see Table 3.1 in Chapter 3). We use the simulator to train a policy π offline with reinforcement learning. We systematically control the simulator to have informativeness $0\% \leq x \leq 100\%$. A 100% informative dialogue responds to all questions completely. For visual questions, we keep a list of all the possible winning conditions sorted by the number of times they have been presented by the dialogue partner in ascending order. When the agent asks for a board demonstration (a `Req_WDem()` question), a simulator with $x\%$ informativeness randomly chooses a win condition to demonstrate from the top $(100 - x)\%$ of the sorted list. For *"yes/no"* (`Conf()`) questions, a $x\%$ informative simulator responds with `Unknown()` to queries with $100 - x\%$ probability, and provides the correct answer (`Affirm()` or `Negate()`) with $x\%$ probability. For *"wh-"* (`Req()`) queries, the simulator provides only $x\%$ of a complete answer.

4.1.1 Belief State

The belief state tracks the state of the dialogue, so that the agent can choose its next dialogue action. The fully trained agent will continue asking questions until it expects no additional reward. Increments of its beliefs about win conditions are positively rewarded, using a metric over the agent's game tree. After each turn, there is an update to the dialogue belief state and to the permanent knowledge store.

The belief state has a hierarchical structure that mirrors the structure of the policy π. The global belief space is a set of belief vectors B that represent beliefs acquired during a dialogue (see Figure 4.2). Each new demonstration D_i instantiates a new local belief vector B_i to represent confirmed information observed in D_i or acquired and inferred from responses to questions about D_i.

The belief state B is the union of the sets of confirmed beliefs (B_C) and inferred beliefs (B_I). Formally, a game belief vector B is defined as the concatenation of all the subvectors that represent distinct types of game knowledge. Each subvector pertains to an observed property of game pieces (e.g., color) or a type of physical rearrangement of a configuration of pieces (e.g., rotate): [1]

$$\begin{aligned}
B_C &= b_{Color_c} \oplus b_{Shape_c} \oplus b_{Hollowness_c} \oplus b_{Height_c} \oplus b_{Size_c} \\
&\oplus b_{Rotate_c} \oplus b_{Translate_c} \oplus b_{OtherPlayer_c} \oplus b_{Board_c} \\
B_I &= b_{Color_i} \oplus b_{Shape_i} \oplus b_{Hollowness_i} \oplus b_{Height_i} \oplus b_{Size_i} \\
&\oplus b_{Rotate_i} \oplus b_{Translate_i} \qquad (4.1) \\
B &= B_C \oplus B_I \qquad (4.2)
\end{aligned}$$

B_C tracks the game knowledge the agent learns directly from the dialogue partner by asking visual or verbal questions. The subvectors correspond to different communicative actions we introduced in Chapter 3. The subvectors $b_{Size_c}, b_{Color_c}, b_{Shape_c}, b_{Hollowness_c}$, and b_{Height_c} track whether one these properties of game pieces on the board contributes to a win condition, which corresponds to the communicative action `Conf(Property())` where `Property()` can be realized as `Size(), Color(), Shape(), Hollowness()` or `Height()`. These subvectors are length three to represent the binary values of each property (e.g. the player's or opponent's colors for `Color (Blue or Red)`, `Short` and `Tall` for `Height`, etc.). Table 3.2 in Chapter 3 shows the complete list of predicates for game piece properties. Additional to the properties, each subvector has a vector element dedicated to None, to represent the lack of knowledge about the contribution of the game piece property to the win condition. The subvectors b_{Rotate_c} and $b_{Translate_c}$ correspond to the ways a board can be reconfigured and still be a win (e.g. moving pieces up or down, or rotating the pieces by an angle θ). These two components are updated when `Conf(ShiftBoard)` or `Req(ShiftBoard)` are used where `ShiftBoard` is realized as `Translate` or `Rotate`. The length of $b_{Translate_c}$ is equal to the number of board positions (i.e. 42 for Connect Four, 16 for Quarto, etc.) plus a vector position for None. The length of b_{Rotate_c} is equal to 4 to represent three different angles the pieces can be rotated on the board (i.e. 45, 90, 135), along with None. The subvector $b_{OtherPlayer_c}$ is length two, and is updated in response to the communicative action `Req_Other()` to ask whether the other player can undo a win condition. The subvector b_{Board_c} represents the board the dialogue partner demonstrates, which corresponds to the communicative

[1] The belief formula presented here shows the complete list of belief subvectors. Depending on the game, some of these components might not be relevant (e.g. Quarto is the only game where pieces have different heights).

action `Req_WDem()`.

Each win condition the agent learns at the end of a turn exchange is added to B_C. The same win conditions are also given as inputs to the GKR, which converts them to paths in the game tree knowledge store. GKR first identifies whether they have already been added to the tree, given that there might be overlaps between the responses to questions the agent asks. The GKR adds each new win condition to the tree as a set of under-specified paths of the form $W_i = \{a_{i1}, a_{i2}, ..., a_i\}, i \in n!$, representing an ordered sequence of n player actions from the start node to a leaf in which the agent wins, and where the opponent's actions are left unspecified.

The belief B_I has the same subvectors as B_C, less the subvectors for the board and the other player. Beliefs about the board are updated only by what the agent sees (virtually), not by what it infers. In principle, the agent could draw inferences about the other player, but our agent draws inferences only about its own actions. For the inferred beliefs B_I, we rely on the game tree reasoner (GKR) to infer new win paths, which it does after it has updated the game tree with confirmed beliefs.

For each new win path W_i, GKR tries to infer new win conditions, which are then added to B_I. The intuition is that paths consisting of a sequence of actions that are siblings of actions in a win path W_i might represent alternative ways to win. That is, the GKR infers new win paths W_j where each next action $a_{jn} \in W_j$ is a sibling of its corresponding action $a_{in} \in W_i$. Formally, for each action a_{in} in W_i, the GKR uses a sibling function to retrieve its siblings:

$$Sibling(a_{in}) = \{(a'_1, 1), (a'_2, 2), ..., (a'_k, k)\} \tag{4.3}$$

where the sibling list consists of 2-tuples of the sibling action and its distance d from a_{in}. Using this function, GKR infers new sibling sequences W_j^d:

$$W_j^d = \{a_{j1}, a_{j2}, ..., a_{jn}\} \tag{4.4}$$

where each action in W_j^d is d steps away from the corresponding actions in $W_i, d \in \{1, 2, 3, .., k\}$, where k is the number of total possible actions less those that are already in the tree.[2]

Figure 4.3(a) illustrates a board demonstration D_1 for Connect Four with a hori-

[2]Thus k decreases as the tree grows. We do not threshold k for any of the games reported in the thesis. However, for very complex games with large tree size, k could be set to less than the total number of unused actions to control for the time complexity of the inference procedure.

zontal sequence of four disks starting in position (0,1), meaning row 0, column 1. Two subvectors of B_C and one of B_I are shown before and after each turn exchange. After the demonstration D_1, the agent has confirmed beliefs about what it saw (B_{board_c}), no confirmed beliefs based on questions about D_1, and inferred beliefs derived by the GKR. The B_C subvector for the board b_{Board_c} is length 42 (7×6), to represent all 42 positions a disc can be placed; the y-axis represents a belief value in [0,1]. For purposes of illustration, we restrict our attention to the subvectors in B_C and B_I about t translations of this win condition, respectively $b_{Translate_c}$ and $b_{Translate_i}$. For $b_{Translate_c}$, the belief range on the x-axis includes an additional position **None**, to represent the lack of confirmed knowledge about translations of D_1. The $b_{Translate_i}$ represents stronger and weaker inferences derived by the GKR from the D_1 knowledge about where it might be possible to translate the row and still have a win: moderate belief in shifting left or right by one, weaker belief in shifting right by two.

To update B_C, we rely on the baseline belief tracking method proposed in [52]. Given a response to a request for a new demonstration or to a verbal question, the subvector $vect_t$ of the belief state B_C gets updated after the agent interprets the response. If the turn exchange is a question and answer about a function (e.g. `Translate()`) or a property (e.g. `Shape()` of game pieces), and the response from the dialogue partner is positive or contains new information, the corresponding subvectors get updated according to equation (4.5). When the dialogue partner response is negative, the relevant subvector is updated according to equation (4.6).

$$P_{vect_t} = 1 - \Big(1 - P_{vect_{t-1}}\Big)\Big(1 - P_{u_t}\Big) \tag{4.5}$$

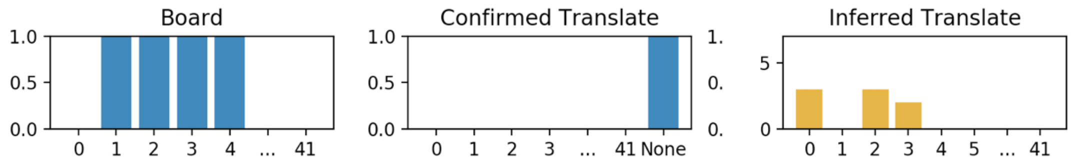
$$P_{vect_t} = \Big(1 - P_{vect_{t-1}}\Big)\Big(1 - P_{u_t}\Big) \tag{4.6}$$

where P_{u_t} is a probability representing the agent's belief in its interpretation of the answer to its question. In this chapter, where we work with MDPs rather than POMDPs, P_{u_t} is always 1.0. For the POMDP we use in Chapter 5, P_{u_t} comes from the NLU module, which assigns probabilities to possible MRL interpretations of natural language answers.

4.1.2 Policy Training

As shown in Figure 4.2, the policy has a hierarchical structure. The global policy π_g chooses communicative actions that determine whether to continue the current sub-dialogue context, or initiate a new one, while the local policy π_l generates questions about the board context established by the global policy.

((a)) D_1

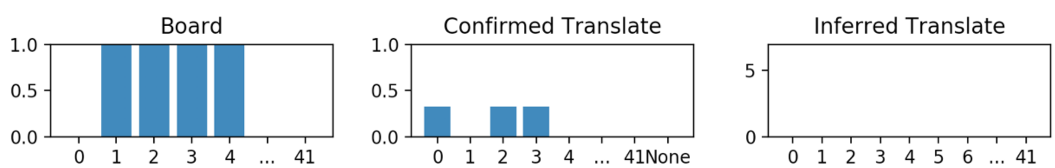


((b)) Belief state after seeing win condition D_1

- **Q$_1$)** Agent: *WinC(1), Req(translate(), {implicit arg* $=$ *D$_1$})*
  (NL: Let us talk about the win condition you showed a minute ago. Where else can these disks be placed to be a win?)
- **A$_1$)** Dialogue Partner: *Inform([0,2])*
  (NL: One column to the left or right.)

((c)) Confirmed and inferred beliefs are updated

- **Q$_2$)** Agent: *WinC(1), Conf(translate(3),{implicit arg* $=$ *D$_1$})*
  (NL: Can it shift two columns over?)
- **A$_2$)** Dialogue Partner: *Affirm()*
  (NL: yes.)



((d)) Confirmed and inferred beliefs are updated

Figure 4.3: An example of two turn exchanges. Questions and answers are presented in both MRL and natural language for clarity. Three components of the belief state that gets updated for these two turn exchanges are shown as well.

Through simulation, we can control the informativeness of the dialogue partner's responses, and thus investigate the impact of informativeness on policy learning. We train multiple policies, setting the dialogue partner informativeness to a value between 0% and 100%. For training, we adopt the Gaussian process, Q-learning approach used in [87]. The authors achieved good results with less training time. The model has relatively few hyper-parameters and converges quickly to a local optimum ($< 20k$ epochs). We adopted their model and trained dialogue polices for $10k$ epochs. The policy gets updated at the end of each dialogue.

4.1.3 Reward

Reinforcement learning depends on the cumulative reward over time as the learning agent explores and exploits its knowledge, to converge on a policy [88]. Given that the agent's goal is to learn win conditions of a game, the reward function includes a measure of how much each next turn adds to the agent's knowledge of a game. A small penalty for each turn encourages efficiency. The overall reward quantifies the value of a question-answer pair based on how many new win conditions the agent learns, and how much this knowledge is *strategically valuable*. The goal is for the agent to not only learn as many win conditions as possible, but to prefer learning those that would increase its chances to win the game (i.e. win conditions with high strategic value). We rely on the game knowledge reasoner (GKR) to compute a *strategic value* for each question answer pair.

The GKR computes a strategic value for a new win condition at a given dialogue state as a function of the number of overlapping actions it has with existing win paths in the tree (i.e. win conditions the agent has learned so far in a dialogue). The higher the number of overlapping actions there are, the higher the strategic value will be. The intuition captured here is that win conditions that overlap more with other win conditions are strategically valuable during play, where the opponent might block a given sequence of player actions. Win conditions that have more overlap with other win conditions give the agent alternative ways to minimize the impact of an opponent's blocking action. Given a game tree with N win paths $\{W_1, W_2, ..., W_n\}$ of length m ($W_i = \{a_{i1}, a_{i2}, ..., a_{im}\}$), the Strategic Value (SV) for a new win path is a conditional summation:

$$SV(W_j) = \sum_{i=1}^{n} \sum_{k=1}^{m} 1[a_{jk} \in W_i] \tag{4.7}$$

where $W_j = \{a_{j1}, a_{j2}, ..., a_{jm}\}$, $j > n$.

The reward is a sum of three terms, designed to encourage the agent to acquire as many new win condition paths as possible, to prefer paths with higher strategic value, and to end the dialogue when the turn costs outweigh the gains in knowledge. The turn cost ensures efficient questioning. The agent learns to decide when to end a conversation by choosing the communicative action `Finish()`, based on the complexity of the game and the informativeness of the partner. Equation 4.8 shows the reward R for a single turn exchange t as a function of the number of new win conditions in the dialogue partner's response to a question (first term in the sum), the strategic value SV of the response, and a turn cost C (through tuning, we found good performance from $\alpha = 0.2$, $\beta = 3$, and $C = 2$.):

$$R_t = \alpha \times \lceil \frac{\#WaystoWin}{\beta} \rceil + SV - C \tag{4.8}$$

In the discounted cumulative reward over time, we use the discount $\gamma = 0.9$ which we found performs the best through tuning.

4.2 In Simulation Experiments

This section reports results on policy convergence behavior during training, and experiments to assess the amount of game knowledge acquired from dialogues after policies have been fully trained. All results are for training or testing with the simulator described above, where we control for the simulators informativeness. We compare the training and testing across three n-in-a-row games: Connect Four, Gobblet and Quarto.

Question 1: Our first question is *how dialogue policy learning differs across levels of dialogue partner informativeness.* Figure 4.4 shows a sensitivity analysis of the training process over $10k$ epochs, using change in total reward, for six informativeness levels ranging from 100% to 20%. The games clearly differ, with much more rapid convergence for Connect Four, the simplest game. Unsurprisingly, it is generally the case that as informativeness of the training simulator decreases, so does the total reward. For Gobblet and Quarto, however, we see occasional crossover of the reward curves. In Gobblet, the early epochs show a higher reward for the 60% informative simulator training than for the 80% informative simulator, with crossover occurring somewhere around epoch 4,000. For Quarto, there are two points of crossover near the end of training for the 50% and 60% simulator training.

Question 2: Using the fully trained policies from Figure 4.4, we ask *how communicative actions differ during game learning dialogues* across games, and across informativeness

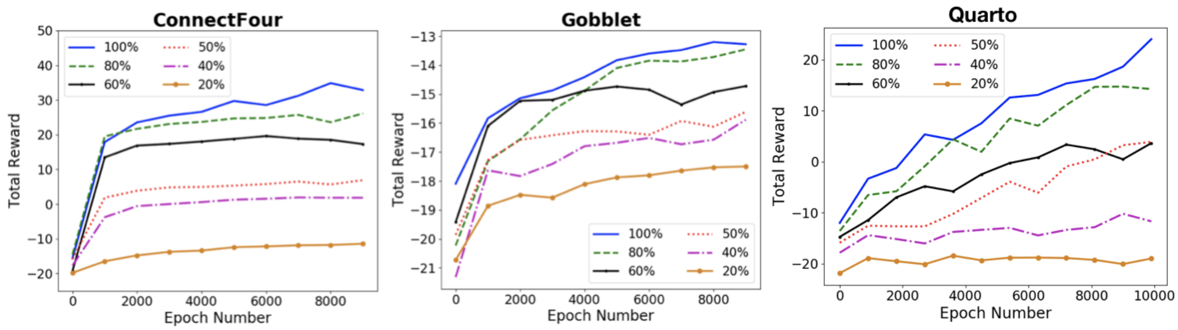


Figure 4.4: Total reward for six levels of dialogue partner informativeness.

of the simulator. For game and informativeness level, the agent engages in 100 dialogues. Tables 4.1-4.3 report the average dialogue lengths in turn exchanges, and average frequencies of each communicative act type across the 100 dialogues, excluding conventional (`Start(), Finish()`); these are always 9%, since every dialogue has an opening and a closing.

The three games lead to different dialogue lengths, with longer dialogues for learning Connect Four. We want to reiterate that we do not cap the dialogue length, and that the agent learns to end the dialogue when the cost of asking questions outweighs the knowledge it is acquiring. For Connect Four, we can see that as the informativeness decreases, so does the length of the dialogue. Since the knowledge acquired from an uninformative partner is low, the cost of asking questions soon outweights the expectation of acquiring new knowledge. For Quarto and Gobblet, the dialogue length is somewhat invariant as the agent can still learn from a low informative dialogue partner given a more complex learning goal. Gobblet and Quarto are more complex games because the game pieces have different features, and win conditions involve both the locations of pieces and their features (color, height, etc.).

For Connect Four, there is no use of `Conf(Property)`, which makes sense, since all the game pieces for one player are identical. More interestingly, `Req_WDem()` (requests for a new demonstration) and `WinC()` (resuming discussion of a previous demonstration) are equiprobable only for the 100% condition; in the other conditions the agent expects more knowledge by resuming discussion of a previous board demonstration than asking for a new one. It is more or less consistent across all three games that as the informativeness decreases, the agent's global policy asks less for a new demonstration (`Req_WDem()`) and more about the ones that are already shown (`WinC()`). When the simulator is less informative, any new demonstration is more likely to be one the agent has already seen (cf. section 4.1), thus there is a greater payoff in asking new questions about seen boards

Connect Four Policies with Six User Levels						
Commun. Action	100%	80%	60%	50%	40%	20%
Req_WDem()	0.45	0.38	0.39	0.39	0.38	0.37
WinC()	0.46	0.53	0.52	0.52	0.53	0.54
Req(ShiftBoard)	0.99	0.90	0.83	0.66	0.92	0.84
Conf(ShiftBoard)	0.01	0.10	0.15	0.32	0.05	0.06
Conf(ChangeDisk)	0.00	0.00	0.02	0.01	0.01	0.03
Conf(Property)	0.00	0.00	0.00	0.00	0.01	0.03
RequestOth()	0.00	0.00	0.01	0.01	0.01	0.04
Dialogue Length	11.2	10.8	10.4	10.5	10.1	9.9

Table 4.1: Dialogue length and action type frequencies for Connect Four.

Gobblet Policies with Six User Levels						
Commun. Action	100%	80%	60%	50%	40%	20%
Req_WDem()	0.37	0.40	0.38	0.35	0.43	0.36
WinC()	0.54	0.51	0.53	0.56	0.49	0.55
Req(ShiftBoard)	0.50	0.45	0.44	0.24	0.44	0.46
Conf(ShiftBoard)	0.08	0.12	0.19	0.23	0.18	0.09
Conf(ChangeDisk)	0.00	0.00	0.00	0.00	0.01	0.01
Conf(Property)	0.40	0.41	0.35	0.50	0.34	0.35
RequestOth()	0.02	0.02	0.02	0.03	0.02	0.07
Dialogue Length	10.4	10.4	10.3	10.3	9.8	10.4

Table 4.2: Dialogue length and action type frequencies for Gobblet.

Quarto Policies with Six Dialogue Partner Levels						
Commun Act	100%	80%	60%	50%	40%	20%
Req_WDem()	0.46	0.44	0.39	0.40	0.33	0.31
WinC()	0.45	0.49	0.52	0.51	0.56	0.60
Req(ShiftBoard)	0.17	0.22	0.23	0.21	0.16	0.20
Conf(ShiftBoard)	0.33	0.15	0.06	0.04	0.12	0.09
Conf(ChangeDisk)	0.01	0.01	0.03	0.04	0.09	0.15
Conf(Property)	0.47	0.62	0.65	0.68	0.58	0.47
RequestOth()	0.02	0.00	0.03	0.03	0.05	0.09
Dialogue Length	10.6	10.3	10.3	9.8	10.2	10.1

Table 4.3: Dialogue length and action type frequencies for Quarto.

than asking for a board it might already have seen.

In all games, frequency of `Conf()` (for *yes/no* questions) is highest for the 50% condition, the user who is neither very informative, nor very uninformative. This is an interesting and behavior we account for as follows. When the partner is very informative, the agent asks more *"wh-"* and open ended questions because the dialogue partner is volunteering complete or nearly complete information. When the partner is somewhat less informative, the agent resorts to *"yes/no"* questions to confirm specific pieces of information with the partner; the policy has learned that *"wh-"* questions do not result in much information gain except with a very informative partner. In short, the policies lead to different dialogue behavior, depending on the game and the dialogue partner informativeness.

Question 3: We next ask *how the policy affects the total game knowledge acquired from a simulator with a given informativeness, and what happens if the agent's fully trained policy for a simulator informativeness level X is used when communicating with a simulator of informativeness level Y.* Table 4.4 shows five policy-simulator (X-Y) conditions we tested, for each of the three games. Under each condition, one dialogue from a set of ten dialogues was randomly selected where we inspected the final game tree knowledge. Each game has four win condition locations, labeling the table rows (row, column, diagonal, anti-diagonal).

As one might expect, when the policy and simulator match (100-100, 50-50, and

Win Type	Policy-User Condition				
	100-100	**100-50**	**50-50**	**50-100**	**20-20**
	Connect Four				
Row	100%	50%	50%	0%	30%
Column	100%	0%	50%	50%	30%
Diagonal	100%	50%	50%	0%	30%
Anti-diagonal	100%	0%	12%	12%	0%
	Gobblet				
Row	100%	1%	50%	1%	1%
Column	100%	1%	2%	100%	2%
Diagonal	100%	100%	0%	0%	1%
Anti-diagonal	100%	0%	100%	1%	100%
	Quarto				
Row	40%	25%	40%	20%	10%
Colum	50%	25%	45%	50%	0%
Diagonal	50%	0%	25%	20%	0%
Anti-diagonal	75%	25%	50%	0%	25%

Table 4.4: Learned wins for five conditions.

20-20 conditions), the agent learns less from a less informative simulator. What is less predictable is that across all three games, if the dialogue partner is neither informative nor uninformative (50%), the agent gains the most game knowledge from using a matching policy (50-50) and not with the policy trained with a more informative partner (50-100). Further, the agent learns less from a 100% dialogue partner using the wrong policy than from a 50% dialogue partner using the right policy. The 50% policy is trained to learn the most from a 50% informative partner and it does worse when the agent interacts with a more informative partner (100%). This shows the importance of the alignment between the policy and the dialogue partner. This can be explained by returning to the observations discussed above about distributions of question types (Tables 4.1-4.3): we saw that the policy changes the frequencies of the different question types in complex ways, depending on both the game and the simulator informativeness.

Question 4: Our final question was *whether the agent could use the same policy to continue learning over a sequence of dialogues.* Because the agent stores it game knowledge in a game tree at the end of a dialogue, it can restore it later to start another dialogue and continue to learn. The ability to continue learning through a succession of dialogues is very important for agents to learn more complex activities, that cannot be learned in a single dialogue. Here we looked at three conditions: where the learned policy matched the dialogue partner informativeness of 100%, 50% and 20%. In each condition, the agent had four dialogues, starting with no knowledge. The agent began each next dialogue with the knowledge it had gained from its previous dialogue. We averaged the final reward at the end of the four dialogues. Results in Figure 4.5 show that for Quarto and Gobblet the agent continues to learn more and more about the game, especially from the 100% informative dialogue partner. However for Connect Four, there is usually little reward (knowledge) left to gain after the first or second dialogue in higher informativeness levels, so the reward plateaus after second or third dialogue.

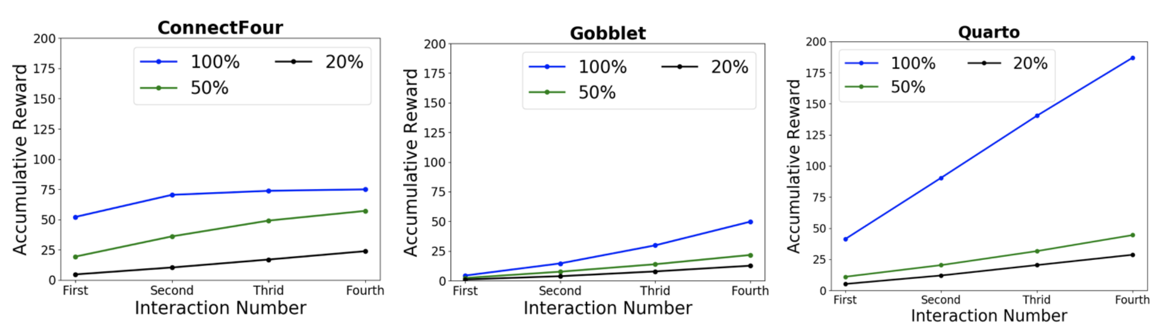


Figure 4.5: Consecutive dialogues reward trend for Connect Four, Gobblet and Quarto.

It is important to note that the reward structure is optimized to help the agent to learn as much as possible in one dialogue. Nothing about the policy training or reward needs to be altered for learning across a sequence of dialogues. The ability to continue to learn through successive dialogues follows from the architecture design, where the agent has a persistent knowledge store. Later, in Chapter 6, we test this ability in dialogues with humans.

4.3 Game Playing Experiments

So far we have seen the way in which the game tree formalism is used as a knowledge store, and for reasoning to draw inferences and to assess the quality of the agent's knowledge. We asserted above that the learned game tree can be used by the agent to engage in actual play. In this section, we report the results of an experiment where we test how well the agent plays the three games after a single learning dialogue. Testing the agent in play provides insight into the strategical value of the agent's game knowledge, by showing how often the agent ends in a win, a loss, or a draw. For this experiment, the agent uses a minimax search algorithm [68] to play with human partners.

For Connect Four and Gobblet, 16 students were recruited to play with the agent. To initialize the agent's game tree, we used the same conditions and knowledge states from Table 4.4. Although we considered having the students play with a physical robot, the slow movements of our Baxter agent (Rethink agentics) resulted in tedious, 20-minute games. Therefore, we used a simulated agent at a terminal instead. Prior to data collection, each subject played a few practice games to become familiar with the game and the interface. Each subject played 10 games, randomly ordered among the 5 conditions per game. A time limit of 2.5 minutes was set for each game; we used 2-step look-ahead in the minimax algorithm. For Connect Four, we can see that the quantity differences in knowledge acquired by the agent show up directly as quality differences (see Table 4.5): the agent wins more when the agent's knowledge state results from a dialogue in which the policy and the simulator informativeness are aligned. For Gobblet, on the other hand, the proportion of outcomes for the agent were more or less the same across the conditions involving a 50% policy and/or a 50% dialogue partner. We attributed the uniform Gobblet results to the time limit for the play and to the need for greater look-ahead, given the greater complexity of Gobblet as compared to Connect Four.

For Quarto, we altered the experiment by removing the restriction on length of play and depth of search. We also developed a graphical user interface to display game pieces

Result	Policy-User Condition				
	100-100	**100-50**	**50-50**	**50-100**	**20-20**
	Connect Four				
Wins	0.78	0.25	0.54	0.35	0.14
Losses	0.07	0.75	0.46	0.58	0.86
Draws	0.15	0.00	0.00	0.07	0.00
	Gobblet				
Wins	0.57	0.17	0.11	0.17	0.10
Losses	0.00	0.73	0.78	0.73	0.82
Draws	0.43	0.10	0.11	0.10	0.08
	Quarto				
Wins	0.94	0.19	0.50	0.18	0.12
Losses	0.00	0.81	0.47	0.78	0.82
Draws	0.06	0.00	0.03	0.04	0.06

Table 4.5: Percentage of agent wins/losses/draws. The minimax algorithm used a lookahead equal to 2 for Connect Four and Gobblet. For Quarto, we removed this restriction.

in a more realistic way. Eighteen students were recruited to play Quarto. The game results in Table 4.5 show that the agent won games more often when it had learned the game from a more informative simulator, as long as it used the corresponding policy.

4.4 Human-Agent Dialogues

As discussed in Chapter 3, the corpus we collected provides training data for natural language understanding (NLU) and natural language generation (NLG) components that will enable the agent to engage in English text-based dialogues with humans. To test how well the agent can learn from dialogues with humans using the MDP policies trained here, we asked the 18 subjects who played Quarto with our virtual agent to engage in text-based dialogues. This section describes the text-based dialogue interface, the dialogue outcomes, and the subjects' informativeness. We developed a GUI for subjects to engage in dialogues with a virtual agent, similar to the GUI used for translating MRL into English (See Figure 3.4 in Chapter 3). The MDP policy for 100% informativeness was used, and belief updating remained the same. In brief, we found that subjects understood the agent and were satisfied with the experience. We also found that subjects provided less than 100% informative answers. The agent's final knowledge states were similar to those where the agent interacted with a 50% informative simulator, using a matching policy.

Each subject engaged in two dialogues. Average dialogue length was 10.96 turn

Win Type	Mean	Min	Max	SDev
Row	20	0	35	7.8
Col	20	0	40	8.3
Diag	15	0	50	13.3
Anti-Diag	5	0	10	1.2

Table 4.6: Average final knowledge states for the 36 dialogues

exchanges (min 9, max 15, std 2.15), which is similar to dialogue lengths with the simulator. Subjects also completed a questionnaire. The questionnaire [3] asked subjects 1) whether they understood the agent's questions, 2) to list the confusing questions by turn number, 3) to rate the fluency of the dialogues on a 5-point scale (i.e., the agent's command of English), and 4) to tell us how willing they would be to have another dialog with this agent. Fourteen of the subjects said they understood the agent most of the time. Inspection of the questions listed as confusing indicated they all had incomplete or incorrect NLG output. The average fluency rating was 2.93. Eleven subjects said they would be willing to have more dialogues, one was neutral, and six were somewhat dissatisfied.

The overall quality of the NLG was good; two thirds of the agent questions were fluent and correct. Of 197 total turn exchanges, 58 were less than perfect. One of the co-authors rated all the generated questions on a five-point scale for correctness and intelligibility, yielding an average score of 4.19 (min 1, max 5, std 1.23). The NLU quality was less good. Subjects' answers were translated to MRL by one of the co-authors, and compared with the NLU output; only 60% of the answers were interpreted correctly. Despite the agent's frequent failure to understand subjects' responses, the average total reward of 12.45 was comparable to the reward for an 80% informative simulator with a matching policy (cf. Figure 4.4). Table 4.6 gives the average final knowledge states for the 36 dialogues, which is in the same range as for dialogues with a 50% informative simulator and matching policy (see Table 4.4). To assess the subjects' informativeness, we examined the 139 turn exchanges that subjects understood well, comparing the subjects' answers to 100% informative answers. Subjects' answers were 100% informative only 41% of the time.

The comparison of baseline human-agent learning dialogues with dialogues between an agent and a simulated dialogue partner shows promise for reinforcement learning of dialogue policies that are trained offline in simulation. Even without the ability to engage in clarification sub-dialogues with a human to clear up confusions, the dialogues

[3]See Appendix A for the complete list of questions.

were all completed. The agent was completely understandable two thirds of the time. The agent learned as much about Quarto as in the 50%-50% simulator condition.

The overall results in this chapter show that agents can learn MDP policies to learn board games through multi-modal dialogues using a relative knowledge goal, namely to increase the agent's game knowledge as much as possible during a short dialogue. We also show that the agent learns different dialogue policies depending on the dialogue partner's informativeness. This work exploits the benefits of a knowledge domain that has a very abstract representation in the form of game trees, where a novel meaning representation language is grounded in the game tree abstraction, as well as in a representation of physical properties of the game boards, pieces, and actions. Additionally, we have demonstrated that MDP policies trained offline in simulation can lead to fairly effective human-agent learning dialogues, based on training data for natural language modules we collected through a novel procedure (See Chapter 3 for details.).

Chapter 5
SPACe: An Adaptive Dialogue Policy for Sensing, Perceiving and Acquiring Knowledge through Communication

In the previous chapter, we introduced MDP dialogue policies trained offline through reinforcement learning. This included demonstrating the impact of the dialogue partner's informativeness on policy training and on how well the fully trained policy acquires game knowledge, testing the agent's ability to play games with human subjects, and testing its ability to communicate with human subjects. We showed that the agent learns best (i.e. learning as much as possible and learning strategically valuable win conditions) when its policy training matches the informativeness of the simulated dialogue partner, that subsequent to learning about a game from a simulator the agent can play the game with people, and that MDP policies support moderately effective natural language dialogues with people. This chapter addresses three limitations of these MDP policies. The first limitation is that MDP policies do not account for the uncertainty in the agent's interpretation of natural language responses to its questions. The second limitation is that we trained different policies for different partner informativeness, using a simulator with a fixed level of informativeness. It is unrealistic to expect that people have a fixed level of informativeness throughout a dialogue, and even more unrealistic to expect that the agent will have *a priori* knowledge of a human dialogue partner's informativeness. The third limitation is that a separate policy was trained for each game, even though the games are all closely related n-in-a-row games. We address these three limitations by presenting a single POMDP policy that can sense and adapt to the dialogue partner's informativeness, and that can determine through its questioning behavior how to ground more general communicative skills for communicating about games so that a single policy can be used for any of the three games, or even for related games unseen during policy training. We show that the POMDP policy trained in simulation transfers well to testing with human dialogue partners.

We introduce a novel dialogue policy that supports **S**ensing, **P**erceiving, and **A**cquiring knowledge through **C**ommunication, which we refer to as **SPACe**. The ability of SPACe to adapt to dialogue partners with a wide range of informativeness arises from two main design features: 1) SPACe has an additional and novel component in its belief state which we refer to as *information dynamics* (See Figure 5.1). This component monitors the information requests and information return between agent and interlocutor. This helps the policy tailor its questions to different interlocutors for maximum information gain: it constantly **senses** the impact of the joint information behavior. For example when the agent senses that the partner is informative, it triggers more open-ended (*"wh-"*) questions, and when the partner is rather uninformative, the agent asks more *"yes/no"* questions. 2) SPACe is trained through a novel approach in which the agent aggressively explores many learning situations with a wide range of simulated dialogue partners. By experiencing dialogues with varied simulators during training, SPACe learns a general policy and can adapt to humans with different informativeness.

Additionally, SPACe provides for grounding general communicative skills in a specific learning situation. As we will discuss later in this chapter, SPACe can learn both about game moves and win conditions. This ability arises from the range of communicative actions we introduced in Chapter 3, and a novel reward structure for reinforcement learning of the policy that encourages the agent to interleave two learning goals. As in the MDP policies seen previously, the SPACe policy enables the agent to ask about win conditions. Additionally, the SPACe policy chooses when to ask questions about how to ground what it **perceives** during a specific game-learning dialogue. Through these grounding communicative actions, a fully trained SPACe policy enables the agent to gradually learn the structure of a game tree for a specific game. This contrasts with the MDP policies presented in the previous chapter, where each MDP policy was for a specific game: a specific game tree structure was assumed *a priori* by each MDP policy. The SPACe policy learns about the actions in a specific game through communication, and therefore learns how to instantiate whichever game tree is required to store its new game knowledge.

We also replace the MDP policy learning with a Partially Observable MDP (POMDP)

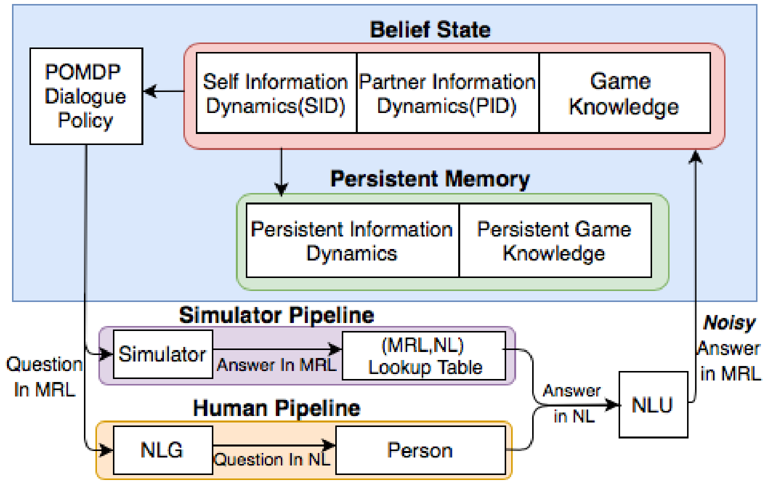


Figure 5.1: Adaptive agent architecture, with a structured belief state for tracking state changes in the dialogue pertaining to the information dynamics of the current dialogue.

dialogue policy as shown in Figure 5.1, where the agent interprets natural language utterances. The main difference between MDP and (PO)MDP policies is that in the latter, states are not observable, so must be estimated from observations. In the case of natural language dialogues, the uncertainty arises from the agent's interpretation of the natural language answers to its questions (i.e. noisy NLU output in Figure 5.1). At each dialogue turn, the agent is in an unobserved state s and learns a probability distribution over the belief state b. Similarly to MDP, reinforcement learning finds an optimal policy π to choose an action given a belief state b that will maximize the expected total reward.

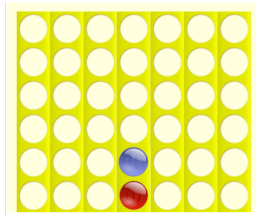
The rest of this chapter is organized as follows. First, we give an intuitive overview of SPACe through an excerpt of a Quarto dialogue in section 5.1. Section 5.2 explains how SPACe distinguishes between perception (i.e.) game knowledge shown or told by the dialogue partner) and cognition (i.e. game knowledge it can learn and ground in its game tree). Next, we introduce **information dynamics**, a component we added to the belief state to enable adaptation to informativeness in section 5.3. Finally, the POMDP and adaptive policy training method and the new reward function are explained in section 5.4.

5.1 SPACe in a Quarto Dialogue

We show a dialogue between SPACe and a simulator responding to questions about Connect Four in Figure 5.2 (again, Connect Four is played in a vertical 6 ×7 grid where players drop disks into columns. Four in a horizontal, vertical or diagonal sequence wins

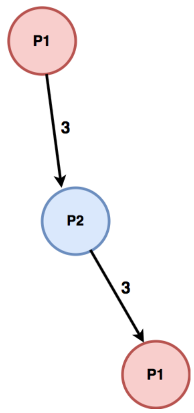
Q1 NL Can you show me a possible way of starting the game?

MR `Req_GDem()`



A1 Img

MR `Inform(Turn(who=me) ∧ Put(piece=red) ∧ Put(pos=(0,3))`
`Turn(who=you) ∧ Put(piece=blue) ∧ Put(pos=(1,3))`



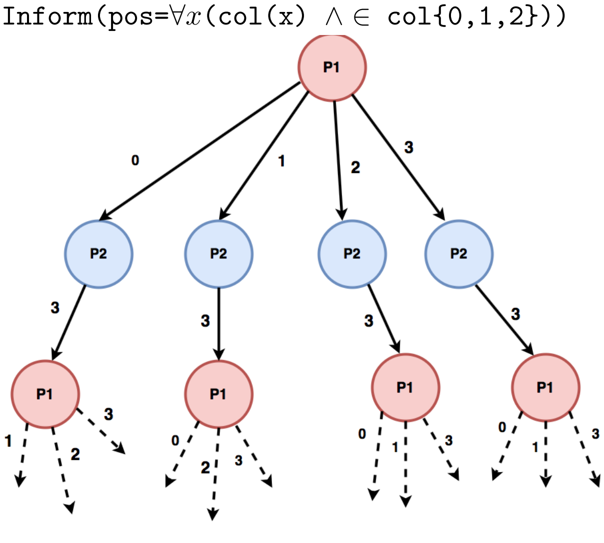
Game Tree

Q2 NL Here, where else can I put my piece?

MR `GameP(0), Req(Location())`

A2 NL Any of the columns to the left.

MR `Inform(pos=`$\forall x$`(col(x) ∧ ∈ col{0,1,2}))`

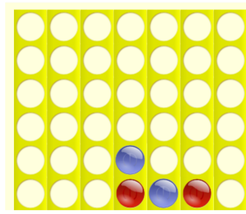
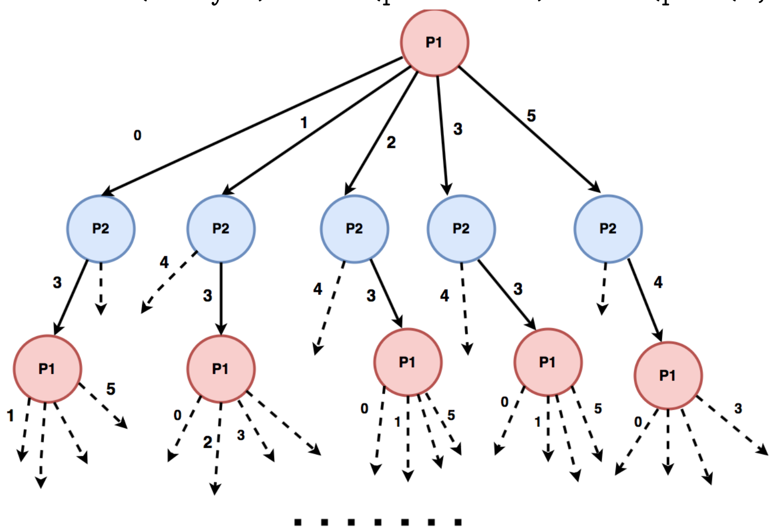
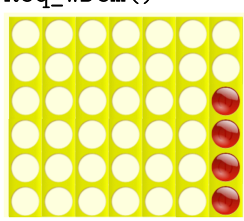
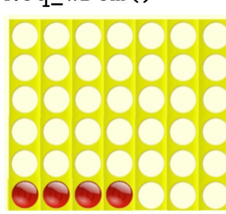


Game Tree

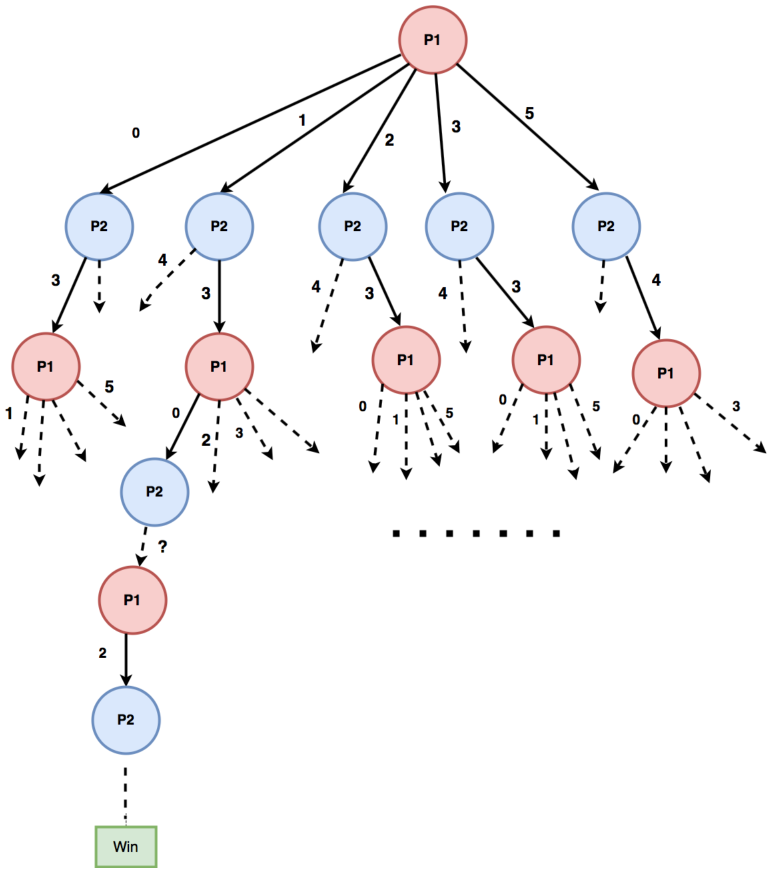
⇒ *Agent perceives and grounds that pieces can go in {(0,0), . . . , (0,3)}*

Q3 NL What's another example of turn taking?

MR `Req_GDem()`

A3	Img	
	MR	`Inform(Turn(who=me) ∧ Put(piece=red) ∧ Put(pos=(0,5))` `Turn(who=you) ∧ Put(piece=blue) ∧ Put(pos=(0,4))`
	Game Tree	
Q4	NL	Can you show me a way to win?
	MR	`Req_WDem()`
A4	Img	
	MR	`Inform (board=`*`visual_representation`*`)`
⇒ *Agent only perceives the column win*		
Q5	NL	Can you show me another way to win?
	MR	`Req_WDem()`
A5	Img	
	MR	`Inform (board=`*`visual_representation`*`)`

the game). Through this example we showcase how information dynamics allows the agent to adapt to its partner's informativeness for a maximum information gain, and how



Game Tree

⇒ *Agent perceives and grounds the row win.*
(The game tree only shows one of the $n!$ *branches terminated with win condition.)*

Figure 5.2: An excerpt of a Connect Four multi-modal dialogue with a simulator. Textual dialogues are in English; for clarity we show the underlying meaning representations (MRs). For turn exchanges where the agent acquires new game knowledge, the resulting game tree is shown; if no game tree is shown, the agent fails to integrate what it perceives into its permanent knowledge store.

the agent distinguishes between what it is told or shown versus what it can understand and learn.

As in Chapter 4, the agent's acquired knowledge is represented as extensive form game trees. SPACe, however, also learns to ground the game moves in two-person, zero-sum games like Connect Four, Gobblet or Quarto. The agent communicates with people or simulators in natural language (NL), as shown in the figure with the corresponding meaning representations (MRs). In Q1, the agent asks for a possible pair of moves for itself (red) and a partner (blue). Below, we refer to the figure to illustrate information dynamics, and the distinction between perception and cognition.

As discussed in more detail later in the chapter, information dynamics is implemented as a data structure in the agent's belief state (see Figure 5.1) to monitor the kinds of questions it asks, such as *"wh-"* questions versus *"yes/no"* questions, and the amount of information in the answers it receives. In Q2, the agent asks an open-ended question about other locations to place the pieces. In A2, the simulator answers with a rather informative response: some other columns in that first row. Information dynamics is updated to indicate the amount of information that is provided in the response to this question. By tracking its queries and the responses the agent adapts its questioning behaviour. For example, with informative dialogue partners, the policy triggers more open-ended questions. More *"yes/no"* questions occur with less informative partners, which lowers the possibility of gaining no information from a response.

The distinction between perception and cognition allows the agent to ground the notion of a game move, and therefore to communicate about many games. When a dialogue about a specific game begins, the agent can ask about individual moves, and about ways to win, where a win condition is a sequence of n pieces in a row, possibly with constraints on the shape or size. As illustrated in A1 and A2 of Figure 5.2, an answer to a question about a move can be a visual demonstration or a verbalization. Here by asking the question in A1, the agent learns a game move where the first player takes action #3 and the second player take the same action after that (See game tree at the end of A1-Q1).

The agent will always be able to interpret answers about moves. When the agent asks questions about ways to win, the answer will be grounded only if it has knowledge of each individual move in a demonstrated or described win path; it needs to know how to add each action to the game tree for a specific game, as games differ in the size of the board, in differentiation of pieces, and the like. In Q3, the agent asks for a new win path. However, it has no grounding for moves in the last column, so the visual demonstration in A4 updates only the agent's belief state, corresponding to its perception. In contrast, with A2 it acquired the grounding for moves in the first five locations of the first row (i.e., (0,0) to (0,3),and (0,5)). Thus, after the demonstration in A5, it both perceives the board and grounds the new way to win Connect Four.

5.2 Perception versus Cognition

The agent has two types of memory: its temporary belief state s, which conditions the agent's choice of communicative action a (See Chapter 4), and its persistent knowledge (See Chapter 3). The former consists of the abstract game knowledge for the game under discussion, images of game boards it has seen (stored as a visual database), and its experiential knowledge of information dynamics. As discussed and shown in previous chapters, with persistent knowledge the agent can later play a game and also have multiple dialogues to learn complex games that are too hard to master in one dialogue. In this section we explain the distinction between the agent's perception and cognition.

In its initial state before any dialogues take place, the agent has generic knowledge about games in the form of a start state G_0, a place-holder for the actions (moves) $\mu \in M$, a procedure to build a sequence of actions consisting of moves μ, and a function place-holder Λ that will define the successor states. The agent also has a notion of a terminal state in which it wins, loses or the game is a draw. Formally, the agent learns different ways to build tree paths $W_i \in W$ consisting of tuples of the player's and opponent's game moves, constrained by function Λ that defines the relation between the actions in the tuple[1]:

$$W_i = \{(\mu_{p1j}, \mu_{p2j})_{j=1}^{n}\}, \mu_{p1j}, \mu_{p2j} \in M, \Lambda(\mu_{p1j}, \mu_{p2j}) \tag{5.1}$$

where $\mu_{p_{1j}}$ represents the jth action of player 1, and $\mu_{p_{2j}}$ represents the jth action of player 2. Once a dialogue starts, the simulator or a person selects a particular game board, and the associated pieces. All other questions the agent asks are determined by its dialogue policy. Communicative action types in Table 3.1 marked with asterisks are for questions about game moves. Answers to questions about game moves provide information to construct tree paths for the current game tree.

In Figure 5.2, the agent's first question is a request for a demonstration of a pair of moves. The answer provides the agent with grounding for two pieces, and two moves on the board as shown in the game tree. With the information in the first turn exchange the agent learns a path in the tree consisting of only two moves $W_1 = \{(\mu_{p11}, \mu_{p21})\}$, where $\mu_{p21} \in \{3\}, \lambda(\mu_{p21}, \mu_{p1}) : \mu_{p21} = \mu_{p11}$. At the end of the second turn exchange, the agent learns a way to build tree paths of the form $W_i = \{(\mu_{p1j}, \mu_{p2j})_{j=1}^{n}\}$, where $\mu_{p1j} \in \{0, 1, 2, 3, 5\}, \mu_{p2j} \in \{3\}$, which means it has learned sequences of actions where the first can put pieces in one of the first five columns, with the restriction that the second player can only put a piece on the fourth column as the first player.

Other question types shown in Table 3.1 are about ways to add paths to the game tree that terminate in a win for the agent. A verbal or visual response to one of these questions gives a winning placement of n of the agent's pieces. The perceived information

[1]The opponent's moves are left unspecified.

will update the agent's belief state, but not necessarily its game knowledge. It can only update its current game tree with a path that terminates in a win if it has grounded each of the agent's individual moves, as illustrated in Q4/A4 of Figure 5.2. For clarity, each question about the grounding also shows a representation of the evolving game tree knowledge.

In sum, the agent always perceives visual or verbal responses to any question, which update its belief state. It attempts to ground the responses in its long-term knowledge about a specific game, such as Quarto. Grounding consists of updating the specification of actions M for a given game, and where possible, updating its game tree with a path of actions $\mu \in M$ that lead to a terminal state. Because the games are zero-sum, it does not need to ask about losses. It acquires implicit knowledge about draws when its knowledge includes a way to fill up the board with no wins.

5.3 New Belief State Component: Information Dynamics

We train a single adaptive dialogue policy offline and in simulation that can generalize across games, and that can adapt to people who respond to questions with different amounts of information. To do so, we extend the belief state with *information dynamics*: the means to continuously sense the joint information behavior of the agent and the dialogue partner. The ability to sense how the dialogue is progressing also leads to a policy that transfers well to dialogues with people, thus leveraging the efficiency of offline training. Information dynamics consists of two measures, *Self Information Dynamics (SID)* and *Partner Information Dynamics (PID)*. The combination of SID and PID provides the agent with the means to assess how well its questioning behavior aligns with the partner's answering behavior. Experiments presented in the next chapter demonstrate how SID and PID independently boost performance over a baseline, and how their combination leads to a the best performance.

How much the agent learns in a dialogue depends in part on the cumulative reward that follows from the types of questions it asks. The belief state uses SID to record the agent's questioning behavior. Formally, SID is defined as a concatenated vector where each element maintains the frequency counts for the 16 categories of questions the agent can generate (See MRL in Chapter 3). SID also maintains the number of times the agent switches between different dialogue contexts. This was included based on an observation that was discussed in the second experiment done in section 4.2 in the previous chapter: With fully informative dialogue partners (simulated or human), the agent rarely resumes a prior context (previously discussed board configuration). The less informative the dialogue partner, the more often the agent resumes a previous context.

The agent's cumulative reward also depends on how fully its questions are answered. Therefore, the belief state uses PID to monitor the responses to *"yes/no"* questions (`Conf()`) for each question type by incrementing the counts for the three types of responses, *"yes"*, *"no"*, or *"unknown"*, and continually updates the probability of getting a positive or negative answer versus a non-answer (*"unknown"*). This allows the agent to assess whether the partner is more or less knowledgeable. For *"wh-"* questions (`Req()`), the PID state representation updates a list of the number of different win paths provided by the dialogue partner, along with the standard deviation of the list. This gives the agent an expectation of the number of ways to win in the partner's answers, thus how generous the partner is with information. Recall that the *"wh-"* questions are open-ended queries (e.g. where else can I put these four pieces on the first row and still win the game? (`Req(Translate())`)), and prompt the partner to share as much as they can.

5.4 Adaptive POMDP Policy Learning and Reward

In this section, we first explain how we train POMDP polices that include the uncertainties inherent in natural language, using the corpus and the natural language understanding model (NLU) introduced in Chapter 3. Then we explain how we train dialogue polices to be adaptive to the partner. An agent that is adaptive to different partners who differ in their informativeness must sense the differences in the way the partners respond to questions. We developed a novel belief state component, information dynamics, that enables the agent to do exactly this. However, the agent must also have a way to explore different questioning behaviors with different partners to learn an efficient way of using its communicative actions that will maximize its expectation of information gain in a short dialogue. We introduce a training approach that allows the agent to have a dialogue with a wide range of partners in section 5.4.2 below. We conclude this section by explaining the reward function in section 5.4.3.

5.4.1 POMDP Policy Learning

To develop realistic dialogue policies where the agent interprets natural language utterances, we train Partially Observable Markov Decision Process (POMDP) policies, similar to previous work on task-oriented dialogue policies [6]. MDP and POMDP policies have only one difference: The states in POMDP are not observable, so must be estimated from observations. In the case of natural language dialogues, the uncertainty is due to the agent's noisy interpretation of the natural language responses to its questions, which represented as noisy NLU output at the far left of Figure 5.1.

To train a POMDP policy we used the dialogue corpus introduced in Chapter 3. Although the dialogue corpus is collected for Quarto, we found it possible to apply the same natural language corpus to our other games, as the vocabulary used in our various n-in-a row games is similar, and the same MRL can be used for multiple games (cf. Section 3.4 in Chapter 3). The corpus consists of close to $13k$ turn exchanges, in the form of $< MRL, NL >$ tuples corresponding to the agent's questions, or corresponding to the simulator's responses. Using this data, we trained our POMDP dialogue policy with the natural language responses (see Figure 5.1, bottom). We used the trained NLU encoder-decoder model introduced in section 3.5 of Chapter 3 to translate from natural language to the meaning representation understood by the agent. During training, when the agent picks an action a, the simulated partner receives a and responds using the MRL. Given the MRL, we sample a corresponding NL from the dataset and feed it to the NLU. The agent updates its belief state based on its interpretation of the NLU response it receives. It's interpretation can be more or less certain, depending on the probability assigned to the interpretation by the NLU model.

5.4.2 Adaptive Policy Learning

As well as developing realistic polices that can handle uncertainties in natural language, we aim to train adaptive policies that leverage the low cost of offline training in simulation, then perform well in dialogues with humans. Part of this is achieved by incorporating information dynamics into the belief state. Information dynamics continuously senses changes in communicative behavior throughout the dialogue, with respect to the information requested and provided. The other part is a training approach where the policy aggressively explores many kinds of communicative actions, and where the simulator randomly varies across epochs with respect to its informativeness, and with respect to which game is being discussed.

For each dialogue during training, we randomly sample an informativeness level for the simulated dialogue partner in the range [0, 100] and a random game. The training method combined with information dynamics allows the agent to explore dialogues with a wide range of dialogue partners while monitoring how the questions it asks and the responses it receives, effect its information gain. Similar to the previous chapter, to train the dialogue policy we use the method proposed in [63] that introduces an offline Q-learning algorithm using Gaussian process.

5.4.3 Reward

The dialogues are now about both learning game moves and about win conditions, therefore we modify the reward function proposed in section 4.1.2 of the previous chapter to not only encourage learning as many as win conditions as possible from a specific partner, but to also learn as much as possible about game moves to ground game actions. Here we use a reward R that is a weighted sum of terms for the amount of new grounding, amount of new win conditions, strategic value, and a turn cost:

$$\begin{aligned} R = &\#Tree\_Branches \times \alpha_1 \\ &+ \#WaystoWin\_InGroundedTree \times \alpha_2 + SV \\ &+ C \end{aligned}$$

where the first term weights the number of new branches (possible moves) the agent has learned, the second term weights the number of new win paths in the game tree (similar to the reward function in 4.1.2), the third term is the strategic value (SV) of the new ways to win, and the fourth term is the cost C, for each turn exchange. For the weights and cost, we found the best performance with $\alpha_1 = 0.2$, $\alpha_2 = 0.25$, and $C = 2.0$. We calculate SV as described in the previous chapter.

The reward encourages a policy that can balance the tradeoff between asking about individual moves and asking about ways to win. Answers to questions about moves ground abstract game actions to observed moves, but do not contribute to the agent's ability to play strategically. Answers to questions about win conditions can be grounded only if the component moves have been grounded. With a fully trained policy, the agent first asks about game moves, then interleaves questions about moves and win conditions. This is similar to the way people learn games: to first learn some moves to learn about winning, then to continue learning both.[2]

[2]From multiple Reddit discussions about board games.

5.5 Summary of SPACe and How to Extend SPACe to other Board Games

This chapter presented SPACe, [3] a single adaptive dialogue policy for learning board games through communication, where the agent asks questions of the dialogue partner. We introduced a novel component of SPACe's dialogue belief state, called information dynamics, that monitors the information requests and information return between agent and interlocutor. This helps the policy tailor its questions to different interlocutors for maximum information gain: it constantly senses the impact of the joint information behavior. We also presented a novel reward structure for reinforcement learning of the policy that encourages the agent to interleave two learning goals: to ground what it perceives in a specific dialogue, and to further its knowledge acquisition (i.e., cognition).

In the next chapter, we will show how we tested SPACe on five games: Connect Four, Gobblet, Quarto, Tic-Tac-Toe, and Gomoku. In one experiment we will demonstrate how with a single adaptive policy, the agent can learn any of games Connect Four, Gobblet, Quarto, under varying conditions of simulator informativeness. In another experiment we will demonstrate how the agent can communicate to learn games it never encountered in training (i.e. Tic-Tac-Toe, and Gomoku). We will show that the agent's ability to track information dynamics from turn-to-turn also supports offline learning of POMDP policies in simulation, that transfer directly to online dialogues with people. This addresses a fundamental limitation of POMDP policies: dependence on massive numbers of learning trials that people would be intolerant of. Our studies with humans subjects show that the agent can communicate effectively with people, and can keep learning a complicated game in a succession of dialogues, that people found the agent easy to interact with, and were impressed with the agent's ability to recall previous dialogues. Although our experiments are only with these five games, we believe SPACe can be easily extended to other n-in-a-row games and even other board games, as it has the ability to learn about a game tree by asking questions about game moves, and has the means to ground general communicative skills in a specific learning situation.

There are three main components of SPACe to extend so as to address new games: the MRL, the belief state, and possibly the game knowledge reasoner (GKR). Recall that the MRL supports the agent's ability to apply general question operators, e.g., to request or to confirm information, in order to formulate context-specific questions, such

[3] The code for SPACe is available at https://github.com/mry94/SPACe

as to confirm whether the color of pieces in the particular game contributes to a winning configuration. To enable the agent to learn new games, the MRL should be extended to allow the agent to communicate about any new actions that are specific to the game. For example, to learn Reversi, where game pieces have two sides (dark and light), and where a piece can be turned over during play, the agent would need a way talk about the new actions. This can be done by extending the `ShiftBoard` operator in the MRL to include a new function `FlipDisk`. This would then allow the agent to formulate open-ended (`Req(FlipDisk)`) or *"yes/no"* questions (`Conf(FlipDisk)`) about this Reversi-specific game action.

In addition to extending the MRL, the corresponding belief state components (i.e. Confirmed Belief and Informed Belief) would need to be extended as well, so that each new communicative function would have a corresponding subvector (e.g. `FlipDisk`). That is, for each new action type and its corresponding MRL expression, the belief state would need a new subvector.

As we emphasize through this thesis, we use extensive-form game trees to store the game knowledge an agent acquires, and we argue that this has multiple benefits for generality of representation. Here we point out that for extending SPACe to new games, there is no need to change the knowledge store for any game that can be represented through extensive form game trees, which covers a wide range of games. SPACe is already designed so that as part of its learning through communication, it learns the set of actions available in a specific game, thus it is already designed to generalize to games with different action sets. It should be noted, however, that for games with very large game trees, new constraints might be needed so as to manage the computational cost of reasoning over the game tree. For example, when inferring new win conditions (See Chapter 4), to control the time complexity of the algorithm, the number of inferred win conditions could be limited by a hyper-parameter k.

Chapter 6 SPACe Experiments

This chapter is dedicated to testing SPACe in experiments in simulation and in dialogues with human subjects. The experiments reported here test our single adaptive policy for how thoroughly the agent learns a given game, across different games and different levels of informativeness of responses. The agent's game knowledge is measured as proportion of all possible win paths, for a given game. To measure increase in knowledge in controlled conditions of informativeness of answers, the experimental conditions set the simulator informativeness to different levels. To assess human informativeness, we bin human subjects by informativeness through measurements made after they complete their dialogues with the agent.

Section 6.1 presents three experiments in simulation. The first demonstrates how a single adaptive policy performs relative to baseline polices and oracle polices trained for a specific informativeness level. The second tests how SPACe can learn a complex game like Quarto in successive dialogues. The third demonstrates how the agent can communicate to learn games it never encountered in training. Section 6.2 demonstrates that the agent can communicate effectively with people, and can keep learning a complicated game in a succession of dialogues. Further, we show that humans vary in informativeness when compared with others, or with their own earlier dialogues. The pattern of agent knowledge gains in a succession of dialogues with people is similar to the pattern in a succession of dialogues with the simulator. People find the agent easy to interact with, and are impressed with the agent's ability to recall previous dialogues.

6.1 In Simulation Experiments

The adaptive dialogue policy presented in the previous chapter allows the agent to adapt to the informativeness of its current dialogue partner for better learning through communication. The overall architecture which stores domain knowledge and experiential knowledge of information dynamics supports continuous learning through communication, meaning in a succession of dialogues. The experiments reported in this section test whether the adaptive policy (SPACe) performs on a par with a non-adaptive policy that has prior knowledge of a dialogue partner's informativeness. This is tested in a single learning dialogue, and again in a succession of two learning dialogues. Then we test SPACe to see how well it can learn an unseen game.

All experiments rely on measuring the difference in game knowledge between the start of the dialogue and after the dialogue ends. For the first experiment, the agent is a blank slate with no game knowledge or experiential knowledge. The results show that the agent with an adaptive policy can learn as much as, and sometimes more than, an oracle policy. In the second experiment, the agent has two dialogues, a blank slate dialogue followed by a dialogue that retains the game knowledge and experiential knowledge from the first dialogue. The second experiment also measures a behavior we call a *"retry"*, where the agent tries to get information in the second dialogue that it failed to get in the preceding one. This experiment shows that agent adapts as well to the partner in the second dialogue as in the first dialogue, and retries occur only when the situation calls for it.

6.1.1 Adaptive Grounding in one Dialogue

The experiment reported here compares the behavior of the single adaptive SPACe policy against oracle conditions consisting of three games by six levels of informativeness. Each condition has an oracle policy trained for the specific game and informativeness level. The comparison assesses how closely the one adaptive policy approximates the many oracles. The more that a SPACe dialogue leads to the same knowledge gain as an oracle dialogue, the more we can say that SPACe is truly adaptive. We also compare against a single baseline policy that lacks SPACe's SID_PID tracking of information dynamics, but has otherwise been trained under the same conditions as SPACe.

The experiment uses the same adaptive policy, trained for three games of increasing complexity: Connect Four, Gobblet and Quarto. For each of 30K training epochs, one of the three games is chosen randomly, and simulator informativeness is randomized to 20%, 40%, 50%, 60%, 80% or 100%. The baseline policy is trained the same way. The 18 oracle policies are trained for 10K epochs.

The question we ask here is: *Can we train a single adaptive policy for an agent to learn Connect Four, Gobblet or Quarto, no matter how informative the dialogue partner*

Informativeness	**Baseline**	**Oracle**	**SPACe**
	Connect Four		
100%	49±36%	98±10%	98±3%
50%	6±5%	38±9%	37±9%
20%	0±5%	6±2%	4±1%
	Gobblet		
100%	23±32%	97±11%	84±8%
50%	20±26%	25±1%	25±3%
20%	0±0%	0±0%	0±0%
	Quarto		
100%	3±5%	**30±14%**	**35 ±11%**
50%	0±0%	**2±0%**	**8±2%**
20%	0±0%	0±0%	0±0%

Table 6.1: Average percentage of learned win conditions (across 100 trials), for 3 games × 3 informativeness levels (Inf).

is? We tested the trained policy for the 3 × 6 conditions of game by informativeness. For each condition, we averaged the resulting percentage of win paths learned across 100 dialogues. To save space, Table 6.1 reports three levels of informativeness, but the trend is the same across all conditions. As informativeness decreases, the agent learns fewer win paths in that game's tree. Similarly, as games increase in complexity, the agent learns less. For example, given 100% informative answers, the agent learns nearly all win paths for Connect Four (98±3%), a large majority for Gobblet (84±5%), and only a third of the total win conditions in Quarto (35±11%). In most conditions, the adaptive policy is close to the oracle policy. SPACe always learns more than the baseline, which demonstrates that information dynamics, rather than the randomized training, leads to the adaptive ability.

An unexpected result occurred in the Quarto results for the 100% and 50% informativeness conditions: the adaptive policy outperformed the oracle (35% vs 30% for 100% inf.; 8% vs 2% for 50% inf.); it is striking that the adaptive policy outperforms the oracle. To explain this, we looked at the kinds of questions that were asked by the oracle agent versus SPACe in this condition.

When the dialogue partner is 50% or 100% informative, and the game is Quarto, SPACe asks many more questions about the properties of game pieces that contribute to win conditions, in comparison to the oracle agent. The knowledge that the Quarto game pieces have four properties, and the constraint that a player wins only if there are four pieces in a row where the pieces share at least one of the properties, is hard for any of the agents to learn in a single dialogue. However, SPACe learns more about this

Informativeness	Grounding	WinC	Length
	Connect Four		
100%	20±3%	67±7%	16.08±2
50%	49±0%	35±0%	14.±3.1
20%	78±0%	6±7%	12.7±2.0
	Gobblet		
100%	17±0%	74±2%	22.9±0.2
50%	32±0%	55±0%	16.3±4.4
20%	83±0%	0±0%	12.1±0.4
	Quarto		
100%	17+-0%	73±0%	22.9+-0.4
50%	81±0%	10±0%	17.9±3.3
20%	83±0%	0±0%	12.4±0.9

Table 6.2: Average percentage of turn exchanges spent on learning about the game moves (Grounding) versus win conditions (WinC), along with the average dialogue length (in turn exchanges) for 3 games × 3 informativeness levels (Inf.) (across 100 trials). The first two columns do not add up to 100%, as two turn exchanges of the dialogues are for greetings and goodbyes.

aspect of the game than the four other agents. It is possible that changing the reward structure to learn a greater diversity of ways to win could encourage the oracle policy to do better in a single dialogue. We speculate that SPACe is able to ask more diverse questions without an explicit reward for diversity because of the way it monitors the joint information dynamics of questioner and respondent. Because SID_PID keeps track of what questions the agent asks, the SPACe policy can tell whether a given property has been asked about.

To provide further insight about the behaviour of the agent, Table 6.2 reports the number of turn exchanges SPACe spent learning about game moves versus win conditions. When the partner is fully informative and volunteers a lot of information, SPACe spends relatively less time asking about the game moves, and more time learning about win conditions. However, when informativeness decreases, most of the questions the agent asks are about game moves. This explains why with a very uninformative partner (20%) the agent learns little about the win conditions of Quarto or Gobblet (see Table 6.1). Further, when partners are more informative, dialogues are longer. With uninformative partners, the turn cost begins to outweigh the knowledge gain. We observed that dialogue lengths for the oracle policies and SPACe were very similar, while the baseline policy dialogue length was consistently around 15 turn exchanges in all conditions.

6.1.2 Adaptation through Successive Dialogues

In this experiment we ask two questions about SPACe. Question one is: *Can the agent adapt equally well to a dialogue partner of a given level of informativeness, no matter whether it is a first or second dialogue?* For the first dialogue in the sequence of two, the agent starts with a blank slate, then carries over the game knowledge and experiential knowledge of information dynamics it acquired in dialogue one as the start state for dialogue two. We measure the total game knowledge acquired in each of the two dialogues as in experiment one. Question two is: *When the agent has a second dialogue to continue its learning through communication, can it leverage the opportunity to fill in any gaps in its knowledge that could not be addressed in the first dialogue?* More specifically, we investigate whether it always asks novel questions in a second dialogue, or whether it ever retries a question it asked in the first dialogue.

Because the dialogue policy generates context-specific questions on the fly, based on a repertoire of question types it knows how to generate, the policy has no way to identify the repeated questions we refer to as retries. Further, the reward structure discourages a policy that would repeat the same question within a dialogue. If a question has been asked already, there would be no expectation of improving the total reward, which has a trade-off between total new game knowledge and the cost of each next turn. However, if the agent is in a dialogue with a partner who is more informative than the partner in the first dialogue, a retried question could be advantageous. For this experiment, all the dialogues are for the agent to learn Quarto. As shown in Table 6.1, SPACe already learns most of the Gobblet game tree after a single dialogue with a 100% informative partner, and learns well over half the game tree with a 50% informative partner.

To assess whether a question that occurred in one dialogue is retried in a following dialogue, we rely on an aspect of the agent's memory not explicitly shown in Figure 5.1. To support context switching, the agent's belief state has unique indices for each game board configuration (demonstration) it received from the dialogue partner. These index vector representations of the board, and corresponding game board images. In this way

Condition	First Dialogue	Second Dialogue	Total
20%-100%	13%±4%	37%±7%	50%±5%
100%-20%	43%±9%	14%±5%	57%±7%
100%-100%	45%±9%	30%±8%	75%±9%

Table 6.3: Percentage of new game tree knowledge acquired in each of two successive dialogues (averaged over 100 trials).

Condition	Question Distribution		Knowledge Acquired	
	retried questions	new questions	retried questions	new questions
20%-100%	**16%**±3%	84%±5%	**18**%±3%	82%±4%
100%-20%	3%±2%	97%±1%	0%±1%	100%±1%
100%-100%	4%±2%	96%±2%	1%±1%	99%±1%

Table 6.4: Question distribution and gained knowledge in the second dialogue with respect to retried and new questions.

the agent can either reference a previous game board verbally, re-display the image of the game board to the dialogue partner, or both.

The agent's persistent memory includes a database of the game board configurations it has communicated about. Thus in a new dialogue, the agent can ask a question about a game board it discussed in a previous dialogue. This question can be a new question about the same game board, or it can be a retried question. To assess whether the agent leverages the opportunity to retry a question, we compare the game board contexts and question MRLs in a first and second dialogue.

For the second question, we report the performance of the agent under three conditions, where the simulator informativeness values in the first and second dialogues are: 1) 20%-100%, 2) 100%-20%, and 3) 100%-100%. We report results for the highest and lowest levels of informativeness to show the agent adapts easily, even when the simulators in the two dialogues are as different as possible. As in experiment 1, we report the average game knowledge acquired in 100 trials.

Table 6.3 reports the new game knowledge acquired in each first and second dialogue for the three conditions, shown as a percentage of all the winning paths in a complete game tree. It also reports the total percentage acquired by the end of the second dialogue. We observe first that the agent adapts to the dialogue partners in each dialogue, but the proportion of the game tree learned from a dialogue partner of a given level of informativeness is not always uniform.

When the dialogue partner is 100% informative, the knowledge gain diminishes from 43-49% (± 9%) in dialogue one, to 30-37% (± 7%) in dialogue two. This can be understood as a consequence of a large gain from any question to a 100% informative partner, along with diminishing returns from any question as the game tree becomes more and more complete. In contrast, with a very uninformative dialogue partner, there is little knowledge that can be acquired in any dialogue, and it is a more uniformly low value (13-14%± 4%). Thus the total percentage of the game tree learned is in the same range of 50%-57%±5-7% when one dialogue partner is 100% informative and the other is

20% informative, irrespective of the order in which they are encountered.

Table 6.4 presents more details on the results of the second dialogues, broken down by two types of questions: retried questions versus completely novel questions. In two of the conditions, the agent rarely retries a question (3-4% of its total questions), and when it does, the retried question adds very little to the agent's game knowledge. In the one condition where the agent first communicates with a very uninformative partner, then with a very informative partner, the agent retries questions 16% of the time ($\pm$ 3%), and to good effect, because this contributes a corresponding proportion of its newly acquired game knowledge (18% $\pm$ 3%).

6.1.3 Learning Games Unseen in Training

This experiment asks: *How well can the single adaptive policy support the agent to learn a game unseen in training, under different conditions of informativeness of responses?* The agent's knowledge grounding currently depends on extensive form zero-sum games, and its dialogue skills are for n-in-a-row games. To test the agent's communicative skills for unseen games, we picked one game from the n-in-a-row family that is simple, Tic-Tac-Toe, and one that is complex, Gomoku.

Given that the agent was not trained to communicate about Tic-Tac-Toe or Gomoku, this experiment confirms that the SPACe agent has communicative skills that can generalize to learning about unseen situations. Table 6.5 shows how much of the two unseen games the agent learns from 100%, 50%, and 20% informative partners, compared with the baseline. In all conditions the SPACe agent learns more than the baseline. Tic-Tac-Toe is played on a small board, and is simpler than Gomoku, therefore the agent learns more about this game. The trend here is consistent with the results shown in section 6.1.1: with decreasing informativeness, there is a decrease in knowledge gained.

We also considered the difference between the agent's perceptions and what it learned for these two games. The agent can perceive more than it can understand, as discussed

Inf.	**Tic-Tac-Toe**		**Gomoku**	
	Baseline	SPACe	Baseline	SPACe
100%	38±10%	50±26%	8±3%	12±15%
50%	2%±4%	3±11%	0±0%	1±6%
20%	0±0%	0±0%	0±0%	0±0%

Table 6.5: Average percentage of learned win conditions in 100 trials, for 2 unseen games $\times$ 3 informativeness levels (Inf.)

Inf.	Tic-Tac-Toe		
	Grounding	WinC	Length
100%	10±5%	76±8%	12.5±1.1
50%	75±7%	2±3%	12.35±0.9
20%	83±7%	1±7%	12.31±0.7
	Gomoku		
100%	21±8%	69±9%	21.7±2.7
50%	47±18%	44±18%	16.3±4.4
20%	87±15%	3±3%	13.9±3.3

Table 6.6: Average percentage of turn exchanges spent asking about game moves (Grounding) versus win conditions (WinC), along with average dialogue length, for two unseen games × three informativeness levels (across 100 trials per condition). The total is less than 100% due to turn exchanges on greetings.

above. To investigate how much information it is presented with versus how much it grounds in the current game tree, we compile averages of presented versus grounded information about win conditions.

Table 6.7 shows how much of the two new games the agent learns from 100%, 50%, and 20% informative simulators. We report the percentage of total win conditions the agent has learned (LWinC) and the information that has been presented visually or verbally to the agent during a dialogue (PInfo), which will necessarily be greater than or equal to the number of learned win conditions. Tic-Tac-Toe is simpler than Gomoku, therefore the agent learns more about this game. Also, the trend is consistent with the results shown in Table 6.5, that with each decrease in informativeness, the amount of learned win conditions or presented win conditions decreases as well.

Table 6.6 reports the proportion of time the agent spends learning about game moves versus win conditions. Again, the results are comparable to those in Table 6.2. The agent has longer dialogues for the more complex game, and with more informative responses. As informativeness decreases, the agent asks relatively more grounding questions.

Inf.	Tic-Tac-Toe		Gomoku	
	PInfo	LWinC	PInfo	LWinC
100%	68±20%	50±26%	12±15%	12±15%
50%	12%±23%	3±11%	2±7%	1±6%
20%	4±7%	0±0%	0±0%	0±0%

Table 6.7: Average percentage of the Presented Infoormation (PInfo), and Learned Win Conditions (LWinC), for 2 unseen games under 3 informativeness levels (Inf.), across 100 trials.

6.2 Human Subjects Study

To assess how well the agent can adapt to different humans in natural language dialogues using text (rather than spoken language), and how natural the humans find the dialogues, we carried out two experiments with the same eight human subjects in each, to be somewhat parallel with the simulation experiments. In experiments with humans, however, it is difficult to control for informativeness, holding all other aspects of human interaction equal. Humans vary in their informativeness in complex ways, undoubtedly combined with variations in other factors that affect their communication. As we report here, we measured their informativeness after the fact, and found much variation within and across subjects.

We also conducted a modification of a retrospective think-aloud study [89], in which participants wrote down their reflections. In a retrospective think-aloud, subjects first complete the task, then share their experience. In this way subjects experience less cognitive overload than in a think-aloud, where they simultaneously engage in a task and articulate what they are experiencing. In our study, both the dialogues with the agent, and the subjects' reflections on their experience, are textual.

We recruited 8 college students between the ages of 18 and 22 to have text-based dialogues with our adaptive agent. As in the experiments with simulated dialogue partners (previous section), we conducted a two-part study. In part one, students engaged in an initial dialogue with the agent and provided their retrospective comments, as explained below. They also responded to a questionnaire [1]. In part two, students engaged in a subsequent dialogue where the agent had some initial knowledge about Quarto, based on one of the dialogues from part two. Subjects again provided retrospective comments, and were given a second questionnaire with a subset of questions from part one. In these experiments, we measure the difference in the agent's game knowledge between the start and end of a dialogue, as in the simulation experiments discussed in the preceding section. We also assess the questionnaire responses about the students' experience engaging in the dialogues.

Subjects were instructed to reflect on their experience during each dialogue, and were given the ability to tag any turn exchange during a dialogue they might want to comment on. When they completed a dialogue, they immediately reviewed the transcript and were asked to enter positive, negative or neutral comments wherever they had an experience to share.

[1] See the complete list of questions in Appendix B.

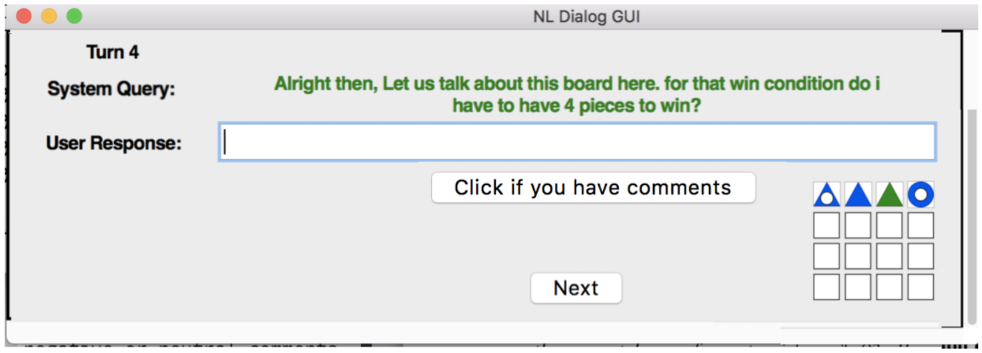


Figure 6.1: The GUI shows the agent's question, a box for a person's response, a 2D representation of the board, and a button to bookmark a turn, for later comments.

In parts one and two, each student had two dialogues each. In the part one dialogues, the agent was a blank slate with no game knowledge or experiential knowledge. In part 2, each agent had some game and experiential knowledge acquired from a part one dialogue. Given the 16 knowledge states and experiential states from the end of each part one dialogue, we randomly assigned two of each to each subject as the start states for the part two dialogues (with replacement).

For the agent to engage in dialogue with humans, it must express its questions in natural language. Therefore we used the natural language generation (NLG) module described in Chapter 3. Subjects engaged in dialogues with the agent using a text-based graphical user interface (GUI) with schematic representations of game boards, similar to the one in Figure 6.1. Each next question from the agent is displayed in the GUI, and people type their responses into a text box. To allow subjects to provide any positive, negative, or neutral comments they might have about each turn exchange, we gave them the option to click on the "click if you have comments" button.

6.2.1 Adaptation within One Dialogue

This experiment addresses three questions: *Q1. How well does the agent learn the game from a human? Q2. What assessment do subjects give of the agent's ability to communicate? Q3. What can we learn about ways to improve the agent's ability to communicate?* We gather information about the informativeness of subject's answers through inspection of their dialogues, and of the agent's belief state. We collect information about the subjects' prior experience with intelligent agents and prior knowledge of Quarto through a questionnaire. We collected information about how subjects experienced the dialogues through a questionnaire, and through the comments they provided retrospectively. Subjects were asked to mark any turns in the dialogue where they had a positive, negative or neutral experience to share with us, that could help us assess their experience or improve the agent.

Subjects were presented with three questions before engaging in any dialogues about their familiarity with Quarto and AI agents. The first question asked subjects how well they knew Quarto. The second question asked if they had previous experience with artificial agents. The third question asked if they had experience communicating with an artificial agent that does more than respond to one-off questions. Three subjects said that they knew the game very well, three said that they were familiar with the basics of the game but did not have complete knowledge of win conditions, and two said they knew very little about the game. Five subjects said they had some experience with AI agents, and three said they had no experience with AI agents. For the third question, all subjects responded negatively.

To assess the informativeness of subject's answers, we computed an informativeness score for each question. For *"wh-"* questions, the score measures what percentage of win conditions in a fully informative answer is provided in the answer. For *"yes/no"* questions where a non-response is an uninformative answer (e.g., "I don't know"), the score is computed as the total number of times the subject responded with an answer ("yes/no") divided by the total number of *"yes/no"* questions in the dialogue.

The overall informativeness of the dialogue is the average informativeness of *"wh-"* and *"yes/no"* questions. We divided the dialogues based on the average informativeness of the subject's answer in the dialogue into three groups of low ($0\% - 33\%$), medium ($33\% - 66\%$) and high ($66\% - 100\%$). Table 6.8 shows the average amount of game knowledge for all 16 dialogues, binned by informativeness of the human partner. The responses were highly informative in nine dialogues, moderately informative in 5 dialogues, and rather uninformative in 1 dialogue. On average, the agent learned 23% of the game from highly informative partners (min: 13, max: 43, std: 9.4), 13% from moderately informative partners (min: 5, max: 20, std: 5.7), and 10% from the one low informative partner.

We also rated the NLG output on a five-point scale (5 is the best) for correctness and intelligibility, resulting in an average score of 4.6 ± 0.2, 4.5 ± 0.2, and 4.6 ± 0 for high, medium, and low informative dialogues, respectively. The NLU accuracy was $83\%\pm8.6\%$, $80\%\pm8.4\%$, and $77\%\pm0\%$ for high, medium, and low informative dialogues, respectively. Subjects comments were mainly about three things: 1) Sometimes the agent repeated a question the subject had already answered; 2) there were some questions the subjects could not understand; 3) due to unfamiliarity with the game, there were questions they

Informativeness	**# Dialogues**	**Game Knowledge**
Low (0%-33%)	1	10.0%±0.0%
Medium (33%-66%)	6	13.0%±5.7%
High (66%-100%)	9	23.8%±9.4%

Table 6.8: Average game knowledge acquired from 16 dialogues from humans with low, medium, and high informative responses.

could not answer. Our inspection of the dialogues in light of the questionnaire responses revealed that case one happened when there was high uncertainty in the NLU, and case two was mainly related to incomplete NLG output. In sum, most of the confusion subjects experienced resulted from imperfect natural language modules. For cases of high uncertainty in the NLU, an improvement would be to have the agent ask the dialogue partner for clarification. To handle confusing NLG, an improvement would be to allow the dialogue partner to ask the agent for clarification. Adding these capabilities would require extending the agent's actions with meta-communicative actions.

The questionnaire that subjects completed after part one or part two asked subjects 1) how often the questions sounded natural or artificial, 2) whether they could answer better if they knew the game better, 3) what aspects of the dialogue they found interesting, and 4) how willing they would be to have another dialogue with this agent. All 8 subjects said most of the questions were like those a human would ask. Six subjects said they might have provided wrong answers because they were unsure of the correct answer. Three subjects mentioned they found it interesting that the agent is able to generalize from win demonstrations by asking about re-configurations of the board demonstration. Seven subjects said they would be willing to have more dialogues, and one was neutral.

The comparison of human-agent dialogues with those between an agent and simulator shows the agent was able to learn from humans that provide answers with different informativeness. Given that the agent is not able to engage in any clarification dialogues, the amount of knowledge the agent acquires from humans is comparable to those learned from simulated dialogue partners. This study also shows that, unsurprisingly, the agent would benefit from clarification sub-dialogues to handle misunderstandings or confusions, especially when there are shortcomings in the NLG or NLU output.

6.2.2 Adaption through Successive Dialogues

In this part we ask the following questions: *Q1. How much can the agent learn from two successive dialogues? Q2. How does the agent's ability to re-display a board it*

Inform.	Prev Dial.	Initial	Additional	Total
Low	Medium: 1	16.0%±0%	5.0 %±0%	21.0%±0%
Medium	Low: 1	10% %± 0%	5% %± 0%	15% %± 0%
	Medium: 2	15%± 1.0%	18%± 1.3%	33%± 1.3%
	High: 3	25.5%± 3.5%	15%± 6.0%	40.5%± 6.0%
High	Medium: 3	9.6 %± 2.8%	14.6 %± 8.7	24.2 %± 6.0%
	High: 6	22.0 %± 5.5%	13.0%± 6.2%	35.0 %± 7.7%

Table 6.9: Columns from left to right: Informativeness of the human in the part 2 dialogue, informativeness of the human in the agent's part 1 dialogue, the proportion of total game knowledge the agent had at the start of the part 2 dialogue, the additional knowledge the agent acquired in the part 2 dialogue, and the total knowledge at the end of both dialogues.

saw in a previous dialogue affect the current dialogue? Q3. Does a difference in the informativeness of the subjects in a sequence of two dialogues affect the agent's ability to learn? As in the previous section, this part is also a retrospective think-aloud where each subject has two dialogues with the agent. Again, to initialize the agent, we randomly sampled a final state from the end states at the end of the dialogues from the previous study; this includes the agent's game knowledge and the agent's knowledge of information dynamics. Subjects were informed that the agent might know a few win conditions, but otherwise they had no knowledge that the agent had already had a dialogue with a human.

Table 6.9 shows the second dialogues grouped by the informativeness of the human partners (column 1). For each group, the informativeness of the partner in the agent's part 1 dialogue is shown (column 2), and how much it learned in that earlier dialogue (column 3). As in the dialogues discussed in the previous section, one human subject in this part had low informativeness, six had medium informativeness and 9 had high informativeness. It is notable that a given human subject did not maintain the same level of informativeness in all dialogues. Two subjects who provided medium informative answers in the previous part provided high informative answers in this part. The one subject with one uninformative dialogue in the previous part had one dialogue in this part in the medium group and one in the high group. One subject whose previous part dialogues were in the high group had one dialogue in the low group in this part. Thus individual human subjects did not have a static informativeness level across both dialogues. Factors such as changes in attention, in engagement, in what aspects of the game were being asked about, could all contribute to such differences.

In the nine dialogues of this part where the human partners had high informativeness, there were three cases where the agent had relatively less initial knowledge (9.6%) due to a previous dialogue with a human partner with low informativeness, compared with six where the agent had relatively more initial knowledge (22.0%). If the agent started with less knowledge, it gained 14.6% of the remaining knowledge (min: 5, max: 22, std: 8.7), concluding the succession of dialogues with 24.2% of the total knowledge. Where the dialogue with an informative partner was begun with more knowledge, the agent learned 13.9% of the remaining knowledge (min: 0, max: 34, std: 6.2), ending this part with 35.0% of the total game knowledge. The six dialogues in this part where the partner had medium informativeness broke down into three groups, depending on the informativeness of the prior partner and the initial knowledge state in this part. The agent started out with 10%, 15% and 25.5% of the game knowledge, and learned 5%, 18%, and 15% of the remaining knowledge, respectively. In the dialogue with an uninformative partner, the agent started the dialogue with 16.0% of the game knowledge, and learned 5.0% of the remaining knowledge, for a total of 21.0% of the game knowledge.

We cannot directly compare results in Table 6.9 with the results from the simulation experiments because the simulated partners had a fixed level of informativeness, whereas we binned the humans by their average informativeness within a dialogue, which was presumably not static. Yet we do see a similar trend: when the agent has little knowledge and is given informative answers it can adapt and benefit from the new dialogue. When it begins a second dialogue with greater knowledge, it can still continue to learn from a more informative person.

Given a human dialogue partner categorized into low, medium or high informativeness, the agent's gain in knowledge from a first or second dialogue is not as uniform as with a simulated dialogue partner, where the informativeness of the partner is fixed and static. With a few exceptions, the trend is that the amount of additional game tree knowledge the agent acquires from a human partner of a given informativeness is rather similar, whether the dialogue is an initial or subsequent dialogue. The major exception is where the agent has a dialogue with a moderately informative partner after a prior dialogue with an uninformative partner.

Here the amount of new knowledge gained in the second dialogue is very low, compared with the cases where the moderately informative dialogue partner followed a dialogue with a moderately or very informative partner. However, the very low gain in knowledge we see in the one condition is based on only two dialogues, given that there was only one dialogue from part one where the human had low informativeness, and is therefore less reliable.

The retrospective comments from subjects in this study indicated there were incom-

1	It's interesting how the agent seems to pull in old information as if it has a memory.
2	This agent asked questions that seemed human since it claimed to know boards from past conversations.
3	These dialogues talked about previous boards they have seen. This dialogue was far more human-like then the previous one.
4	This dialog seemed a lot cleaner and that the agent knew more about the game and how to communicate.

Figure 6.2: Comments from four subjects, after having the second dialogue.

plete questions from the agent due to NLG errors, as in the previous study comments. The questionnaire for this part asked subjects whether they felt like they helped the agent learn more about the game. All but one responded with a yes and one responded no.

All subjects said the dialogues in this part were more interesting, and that the agent's ability to display an image of a win condition it had seen in a previous dialogue very human-like. Figure 6.2 shows representative responses from four subjects to the last item on the questionnaire, which elicited additional comments about their experience. All four comments rate this experience as better than in the first dialogue, and three refer to the agent's ability to switch contexts to previous demonstrations, or even previous dialogues.

The comment from subject two is interesting (*"the agent . . . claimed to know boards from past conversations"*) because the agent lacks the ability to discuss its previous dialogues, and subjects were not told that the agent's initial knowledge came from previous dialogues. However, when the agent asks about a previous demonstration, it always displays the associated image of the board configuration, and says things like *"Going back to this earlier board, ..."*.

Subject two must have inferred that the agent had a previous dialogue because it displayed a board that the subject had not seen before. This subject had high informativeness in both his part 2 dialogues, whereas the initial dialogues had been with a low and a medium informative subject. The agent presumably sensed that with this new highly informative partner, it could profitably switch the context to a demonstration from one of its previous dialogues. In both dialogues with this subject, 7 of 14 questions were about demonstrations from the earlier dialogue. Note that here we did not save the questions from the previous dialogues, only the knowledge states, so we cannot assess whether the agent retried a specific question.

Chapter 7 Conclusion and Discussion

We introduced an agent that learns using a general, meaning representation language (MRL) that directly represents the underlying meanings of questions and answers, and therefore avoids the ambiguity inherent in natural language. The MRL is compositional, so that the agent can generate context-dependent questions. With a novel dialogue policy, the agent senses, perceives and adds to its knowledge through multi-modal dialogues with humans. The reward function for reinforcement learning is designed so that the agent can interleave two learning goals within a dialogue. The first communicative goal is how to ground what the agent is shown and told, meaning how to convert its perceptions into knowledge. The second goal is how to pursue a specific knowledge goal (i.e., increase its cognition). To demonstrate that the fully trained policy is a step towards general communication skills, we showed that the agent can apply its communicative skills to learn about situations never seen during training.

The sensing ability supports POMDP policy training fully offline through simulation. The motivation to develop an adaptive POMDP policy that relies on a belief state that senses the informativeness of the dialogue partner is twofold. One motivation is to investigate whether the informativeness of the dialogue partner affects the communicative actions that the agent chooses, which are by definition conditioned on the current state. The second is to investigate whether such an agent can be trained in simulation and then act in the real world, thus at least partly solving the problem of the high cost to train a POMDP policy in the real world. We showed that the agent learns to choose questions conditioned on its knowledge of information dynamics that allows it to surpass an oracle condition (See Chapter 6).

In experiments reported here where the agent interacts with human subjects, we have demonstrated that the agent learns a different amount of game knowledge, depending on the informativeness of the dialogue partner, similar to the results from the simulation experiments. Further, given a human whose average informativeness over a dialogue is $\mathcal{X}$, the agent generally gains about the same amount of additional knowledge from that human, regardless of whether it had prior dialogues, and regardless of the informativeness of the previous partner. We also showed that human subjects found it easy and interesting to communicate with the agent.

We expected that a human could have different levels of informativeness, due to a wide range of factors such as different levels of familiarity with different aspects of a game, different levels of attention, and the like. We have collected multiple dialogues from the same and different humans. After-the-fact measurement of human subjects' informativeness shows that humans clearly differ in the informativeness of their responses to questions, both in comparison to other humans, and when comparing the same subject across dialogues. This provides partial evidence supporting our approach to train in simulation by randomly varying the simulator's informativeness. The fully trained agent constantly updates its estimation of a dialogue partner's informativeness, and its persistent knowledge of information dynamics.

A particularly distinctive component of our agent is that it has a long term knowledge state, one which incorporates its game knowledge, direct experience as reflected in the game boards it has seen, and its knowledge of information dynamics. This is important for an agent to learn activities that are too complex to master in a single dialogue. Although we did not explicitly design our agent to be able to retry a question, the ability to do so when it is expedient is a natural outcome of the agent's design. The human subjects found the agent's ability to build on its prior knowledge, and to present a game board on its own for discussion, particularly interesting.

7.1 Limitations and Future Work

Here we discuss a few lines of research for future work that could address some of the limitations. In a dialogue, each utterance is an communicative action. For a dialogue to be successful the utterances should be informative, truthful, relevant, and clear (the four maxims of communication [86]). As mentioned in Chapter 4, this thesis addresses the first maxim, as we make no assumption that the dialogue partners will be fully informative all the time. We developed adaptive agents that learn from their partners no matter how informative they are. Dealing with partners that are deceptive is outside of the scope of this work. However, here we discuss how future work can address the two other maxims: clarity and relevance.

7.1.1 Clarification Sub-Dialogue Capabilities

Clarification questions are asked when a dialogue participant does not (fully) understand what the interlocutor meant by an utterance. Existing work usually differentiates between questions (communicative actions) that are to address the dialogue goal, and questions whose function is to resolve confusions [90]. Distinct sub-dialogue mechanisms for clarification of confusions have long been incorporated into spoken dialogue systems, but mainly to resolve speech recognition errors, due to a high rate of so-called disfluencies in spontaneous speech (e.g., repeated words, word fragments, self-correction) [91]. We intentionally work with textual interaction to avoid issues specific to speech recognition, and we assumed dialogue partners would communicate clearly. However, we observed in our studies with human subjects that confusions about the intended meanings of the question or the responses of the dialogue partner do occasionally arise in the game dialogues. This is mainly due to the shortcomings in the natural language modules (NLG and NLU). For example, sometimes the output of the NLG module is incoherent.

Future work could investigate the kinds of confusions that arise in the dialogues, and add new communicative actions for clarification. Deciding when to resort to clarification actions would be the job of the global policy, and the local policy would choose the specific type of clarification action to formulate in a given context. There are different strategies for clarifying a confusion. A naive approach would be to prompt the partner to repeat or rephrase a confusing utterance. This can lead to frustration if the confusion is not resolved. A more successful strategy would be to ask follow-up questions using knowledge about the semantics of the domain as clues to clarify misunderstandings at the intent level, rather than at the word level, as done in [92]. As a future work, it would be interesting to investigate how the agent might use its abstract game representation as a vehicle to resolve confusion.

7.1.2 Mixed Initiative Dialogue Policies

In the dialogues presented here, the agent is the active partner in the dialogue, deciding what is relevant to discuss in a dialogue, and seeking to learn what it does not know from any available interlocutor. For greater flexibility in conversational interaction, it would be interesting to investigate how to support dialogues where the interlocutor volunteers information without being asked, or takes the initiative in other ways. As observed in [93], the human teacher might have a plan to offer feedback on the learner's work, or the agent might need to explain that it is confused. In human-human interaction, turn-taking behaviors vary considerably, depending on the task [94].

Mixed initiative is used in current plan-based human-robot systems [95]. There has been little if any work, however, on reinforcement learning of mixed initiative dialogue policies. In [96], a distribution was learned from dialogue data for the probability of each participant in a task-oriented dialogue to have the initiative, and was incorporated into a manually engineered dialogue management system that separates the tasks of determining initiative from goal-based actions.

Because SPACe already has a hierarchical policy, we can assign the task of initiative to the global policy level, and because polices are trained in simulation, different strategies for switching the dialogue initiative can be explored with a low cost. In [97], a restricted initiative strategy that had been identified by analyzing a human-human problem solving corpus was found to perform best. In this strategy, initiative remains with a given dialogue participant for the duration of a sub-dialogue corresponding to a sub-task in the problem solution. We speculate that in the kind of learning dialogue we investigate here, restricted initiative [97] could benefit the communication and learning by allowing the human participant to take the initiative during a particular sub-dialogue (dialogues about a specific board demonstration). A particular signal that can trigger changing the initiative to the user, could be the confusions or misunderstanding that occurs during a dialogue, as discussed in previous section, the user can take charge of the dialogue to explain something. It will be important to also test whether humans prefer dialogues with such policies, and to compare how well the agent learn in dialogues with different initiative.

7.1.3 Reasoning Capabilities across Games

By reasoning across games, we mean the ability for an agent to go beyond learning about a particular game, to learning about and developing the ability to discuss how different games relate to one another. This could allow the agent to to learn games that are similar to the ones it has already learned more efficiently. For example Gobblet and Quarto are similar in a few ways: they are both played on a 4×4 horizontal grid where a player wins once four pieces are placed on a straight line. They differ in terms of game piece properties; in Quarto pieces have four different features. If the agent had the ability to reason about similarities and difference between games, once it learned Gobblet, then it could learn Quarto more efficiently, and perhaps learn games that are increasingly dissimilar.

As there is no prior work we know of on such dialogues, the first step towards such agent would be collect human-human dialogues in which an expert in Quarto and Gobblet engage in dialogues with a novice, first to teach the novice Gobblet and then Quarto. Verbal and visual questionnaires could be developed to probe how candidates discuss strategy, and how they perceive and recall images of random versus meaningful game configurations. In [98], authors investigate how novices and experts recall Othello or Chess boards, and how this ability allows the experts to develop complex strategies faster. Because the agent relies on a persistent game tree that can operate as a long-term memory, it can access what it has learned in previous dialogues. On top of a memory of the games it has learned, the agent needs a mechanism to enable reasoning across different game trees, and means to quantify similarity or differences of game trees as well.

Appendix A
Questionnaire for Dialogues of Human Subjects with Non-adaptive Agent

Here we present the complete list of questions asked from students. This was for our first study with human subjects that was discussed in Chapter 4.

1. Were you able to understand what the agent was asking you even when it was not completely fluent English.
2. If there were questions you could not understand, please list them below by turn number.
3. For questions you could not understand please try to explain to us your confusion for each of the turns you listed above. (Please use the turn number again)
4. On a scale of 1 to 5 (5 is the best) how likely you would be think that this dialog was typed in by an English speaker?
5. What aspects of this dialog did you find interesting, if any?
6. How likely you would come back and have another dialog about a game with this agent?
7. What aspects of the GUI do you think can be improved?

Appendix B
Questionnaire for Dialogues of Human Subjects with SPACe

Here we present the complete list of questions asked from subjects when having dialogues with SPACe.

1. How often were the questions like those a human might have asked?
2. How often were the questions like those an artificial type of "decision maker”?
3. How often did you feel you could understand the agent’s question better if you knew the game better yourself?
4. Did you feel like you helped the agent learn the game?
5. What aspects of the dialogues did you find interesting, if any?
6. How willing would you be to have another dialogue about a game with this agent?
7. You are among a very small group of people who has had a "dialogue" with an agent that learned how to do something new from you through communication. The final optional part of this study is for you to let tell us if you have any additional comments.

Bibliography

[1] Laird, J. E. (2012) *The Soar Cognitive Architecture*, The MIT Press, Cambridge, MA, US.

[2] Ritter, F. E., F. Tehranchi, and J. D. Oury (2019) "ACT-R: A cognitive architecture for modeling cognition," *WIREs Cognitive Science*, **10**(3), p. e1488, https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcs.1488.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wcs.1488

[3] Fusaroli, R., J. Raczaszek-Leonardi, and T. Kristian (2014) "Dialog as interpersonal synergy," *New Ideas in Psychology*, **32**, pp. 147–157.

[4] Reitter, D. and J. D. Moore (2014) "Alignment and task success in spoken dialogue," *Journal of Memory and Language*, **76**, pp. 29–46.
URL https://www.sciencedirect.com/science/article/pii/S0749596X14000576

[5] Yu, W., J. Tan, C. K. Liu, and G. Turk (2017) "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*.

[6] Young, S., M. Gašić, B. Thomson, and J. D. Williams (2013) "POMDP-Based Statistical Spoken Dialog Systems: A Review," *Proceedings of the IEEE*, **101**(5), pp. 1160–1179.

[7] Yu, Y., A. Eshghi, and O. Lemon (2016) "Training an adaptive dialogue policy for interactive learning of visually grounded word meanings," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Association for Computational Linguistics, Los Angeles, pp. 339–349.
URL https://www.aclweb.org/anthology/W16-3643

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

[9] Rana, M. A., M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots (2017) "Towards Robust Skill Generalization: Unifying Learning from Demonstration and Motion Planning," in *Conference on Robot Learning*, PMLR, pp. 109–118.

[10] Kaiser, L. (2012) "Learning Games from Videos Guided by Descriptive Complexity," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, MIT Press, Toronto, Ontario, Canada, pp. 963–69.

[11] Hinrichs, T. R. and K. D. Forbus (2013) "X Goes First: Teaching Simple Games through Multimodal Interaction," in *Proceedings of the Second Conference on Advances in Cognitive Systems*, pp. 31–46.

[12] Whitney, D., E. Rosen, J. MacGlashan, L. L. Wong, and S. Tellex (2017) "Reducing Errors in Object-Fetching Interactions through Social Feedback," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1006–1013.

[13] Racca, M. and V. Kyrki (2018) "Active robot learning for temporal task models," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, pp. 123–131.

[14] Cakmak, M. and A. L. Thomaz (2012) "Designing robot learners that ask good questions," in *Proceedings of the seventh annual ACM/IEEE International conference on Human-Robot Interaction*, pp. 17–24.

[15] Kirk, J. R. and J. E. Laird (2014) "Interactive task learning for simple games," in *Advances in Cognitive Systems*, pp. 11–28.

[16] ——— (2019) "Learning hierarchical symbolic representations to support Interactive Task Learning and knowledge transfer," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Macao, China, pp. 6095–6102.
URL `https://doi.org/10.24963/ijcai.2019/844`

[17] Mohan, S. and J. 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, pp. 387–394.
URL `http://dl.acm.org/citation.cfm?id=2893873.2893934`

[18] Scheutz, M., E. Krause, B. Oosterveld, T. Frasca, and R. 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, pp. 1378–1386.
URL `http://dl.acm.org/citation.cfm?id=3091282.3091315`

[19] Melo, F. S., C. Guerra, and M. 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, pp. 2567–2573.
URL `https://doi.org/10.24963/ijcai.2018/356`

[20] Matuszek, C. (2018) "Grounded language learning: where robotics and NLP meet," in *IJCAI*, pp. 5687–5691.

[21] Pillai, N., F. Ferraro, and C. Matuszek (2019) "Deep Learning for Category-Free Grounded Language Acquisition," *UMBC Student Collection*.

[22] Chai, J. Y., Q. Gao, L. She, S. Yang, S. Saba-Sadiya, and G. Xu (2018) "Language to Action: Towards Interactive Task Learning with Physical Agents." in *IJCAI*, pp. 2–9.

[23] Scheutz, M., E. Krause, B. Oosterveld, T. Frasca, and R. 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*.

[24] Liu, C., S. Yang, S. Saba-Sadiya, N. Shukla, Y. He, S.-C. Zhu, and J. Chai (2016) "Jointly Learning Grounded Task Structures from Language Instruction and Visual Demonstration," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, pp. 1482–1492.
URL `https://www.aclweb.org/anthology/D16-1155`

[25] Kasenberg, D., A. Roque, R. Thielstrom, and M. Scheutz (2019) "Engaging in Dialogue about an Agent's Norms and Behaviors," in *Proceedings of the 1st Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence (NL4XAI)*, pp. 26–28.

[26] Kasenberg, D., A. Roque, R. Thielstrom, M. Chita-Tegmark, and M. Scheutz (2019) "Generating justifications for norm-related agent decisions," in *Proceedings of the 12th International Conference on Natural Language Generation*, Association for Computational Linguistics, Tokyo, Japan, pp. 484–493.
URL `https://www.aclweb.org/anthology/W19-8660`

[27] Steedman, M. and J. Baldridge (2011) "Combinatory Categorial Grammar," in *Non-Transformational Syntax* (R. Borsley and K. Börjars, eds.), Wiley, pp. 181–224.

[28] Gatt, A. and E. Reiter (2009) "SimpleNLG: A Realisation Engine for Practical Applications," in *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, Association for Computational Linguistics, Athens, Greece, pp. 90–93.
URL `https://www.aclweb.org/anthology/W09-0613`

[29] Galescu, L., C. M. Teng, J. Allen, and I. Perera (2018) "Cogent: A Generic Dialogue System Shell Based on a Collaborative Problem Solving Model," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, Association for Computational Linguistics, Melbourne, Australia, pp. 400–409.
URL `https://www.aclweb.org/anthology/W18-5048`

[30] Perera, I., J. Allen, C. M. Teng, and L. Galescu (2018) "Building and Learning Structures in a Situated Blocks World Through Deep Language Understanding," in *Proceedings of the First International Workshop on Spatial Language Understanding*, Association for Computational Linguistics, New Orleans, pp. 12–20. URL `https://www.aclweb.org/anthology/W18-1402`

[31] Gašić, M., N. Mrkšić, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young (2017) "Dialogue manager domain adaptation using Gaussian process reinforcement learning," *Computer Speech & Language*, **45**, pp. 552–569.

[32] Yu, Y., A. Eshghi, and O. Lemon (2017) "Training an adaptive dialogue policy for interactive learning of visually grounded word meanings," *arXiv preprint arXiv:1709.10426*.

[33] Torrey, C., A. Powers, M. Marge, S. R. Fussell, and S. Kiesler (2006) "Effects of adaptive robot dialogue on information exchange and social relations," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, ACM, Utah, USA, pp. 126–133.

[34] Wang, Z. and M. E. Taylor (2018) "Interactive Reinforcement Learning with Dynamic Reuse of Prior Knowledge from Human/Agent's Demonstration," *arXiv preprint arXiv:1805.04493*.

[35] Sutton, R. S. (1990) "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Machine learning proceedings 1990*, Elsevier, pp. 216–224.

[36] Peng, B., X. Li, J. Gao, J. Liu, K.-F. Wong, and S.-Y. Su (2018) "Deep dyna-q: Integrating planning for task-completion dialogue policy learning," *arXiv preprint arXiv:1801.06176*.

[37] Wu, Y., X. Li, J. Liu, J. Gao, and Y. Yang (2019) "Switch-based active deep dyna-q: Efficient adaptive planning for task-completion dialogue policy learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7289–7296.

[38] Abraham, I., A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox (2020) "Model-based generalization under parameter uncertainty using path integral control," *IEEE Robotics and Automation Letters*, **5**(2), pp. 2864–2871.

[39] Bousmalis, K., A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. (2018) "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, IEEE, Brisbane Convention and Exhibition Centre, Brisbane, Australia, pp. 4243–4250.

[40] Tan, J., T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke (2018) "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*.

[41] Tobin, J., R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel (2017) "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, pp. 23–30.

[42] Chen, T., A. Murali, and A. Gupta (2018) "Hardware conditioned policies for multi-robot transfer learning," *arXiv preprint arXiv:1811.09864*.

[43] Zhang, J., T. Zhao, and Z. Yu (2018) "Multimodal hierarchical reinforcement learning policy for task-oriented visual dialog," *arXiv preprint arXiv:1805.03257*.

[44] Yu, W., C. K. Liu, and G. Turk (2018) "Policy transfer with strategy optimization," *arXiv preprint arXiv:1810.05751*.

[45] Levin, E., R. Pieraccini, and W. Eckert (1997) "Learning dialogue strategies within the Markov decision process framework," in *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, IEEE, Santa Barbara, CA, pp. 72–79.

[46] Kaelbling, L. P., M. L. Littman, and A. R. Cassandra (1998) "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, **101**(1-2), pp. 99–134.

[47] Bohus, D. and A. Rudnicky (2006) "A "k hypotheses+ other" belief updating model," .

[48] Williams, J. D. and S. Young (2007) "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech & Language*, **21**(2), pp. 393–422.

[49] Young, S., M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu (2010) "The hidden information state model: A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, **24**(2), pp. 150–174.

[50] Henderson, M., B. Thomson, and S. Young (2014) "Word-based dialog state tracking with recurrent neural networks," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 292–299.

[51] Thomson, B. and S. Young (2010) "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech & Language*, **24**(4), pp. 562–588.

[52] Wang, Z. and O. 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*, pp. 423–432.

[53] Wen, T.-H., D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young (2016) "A network-based end-to-end trainable task-oriented dialogue system," *arXiv preprint arXiv:1604.04562*.

[54] Mairesse, F., M. Gasic, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young (2009) "Spoken language understanding from unaligned data using discriminative classification models," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, pp. 4749–4752.

[55] Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014) "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*.

[56] Sutskever, I., O. Vinyals, and Q. V. Le (2014) "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*.

[57] Bahdanau, D., K. Cho, and Y. Bengio (2014) "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*.

[58] Luong, M.-T., H. Pham, and C. D. Manning (2015) "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*.

[59] Dušek, O. and F. Jurčíček (2016) "Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings," *arXiv preprint arXiv:1606.05491*.

[60] Tran, V.-K. and L.-M. Nguyen (2017) "Neural-based natural language generation in dialogue using rnn encoder-decoder with semantic aggregation," *arXiv preprint arXiv:1706.06714*.

[61] Su, S.-Y., K.-L. Lo, Y.-T. Yeh, and Y.-N. Chen (2018) "Natural Language Generation by Hierarchical Decoding with Linguistic Patterns," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Association for Computational Linguistics, New Orleans, Louisiana, pp. 61–66.

[62] Schatzmann, J., K. Weilhammer, M. Stuttle, and S. Young (2006) "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies," *Knowledge Engineering Review*, **21**(2), pp. 97–126.

[63] Gašić, M. and S. Young (2013) "Gaussian Processes for POMDP-based dialogue manager optimization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **22**(1), pp. 28–40.

[64] Shah, P., D. Hakkani-Tur, and L. Heck (2016) "Interactive reinforcement learning for task-oriented dialogue management," *Workshop on Deep Learning for Action and Interaction, NIPS 2016 (2016)*.

[65] Gašić, M., N. Mrkšić, P.-h. Su, D. Vandyke, T.-H. Wen, and S. Young (2015) "Policy committee for adaptation in multi-domain spoken dialogue systems," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, IEEE, pp. 806–812.

[66] Tresp, V. (2000) "A Bayesian committee machine," *Neural computation*, **12**(11), pp. 2719–2741.

[67] Chai, J. Y., Q. Gao, L. She, S. Yang, S. Saba-Sadiya, and G. 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, pp. 2–9.
URL `https://doi.org/10.24963/ijcai.2018/1`

[68] Osborne, M. J. and A. Rubinstein (1994) *A course in game theory*, MIT press.

[69] Newell, E., A. Schang, D. Margolin, and D. Ruths (2017) "Assessing the Verifiability of Attributions in News Text," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Asian Federation of Natural Language Processing, Taipei, Taiwan, pp. 754–763.
URL `https://www.aclweb.org/anthology/I17-1076`

[70] Hermann, K. M., F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. M. Czarnecki, M. Jaderberg, D. Teplyashin, et al. (2017) "Grounded language learning in a simulated 3d world," *arXiv preprint arXiv:1706.06551*.

[71] Winograd, T. (1972) "Understanding natural language," *Cognitive psychology*, **3**(1), pp. 1–191.

[72] Quine, W. V. O. (1960) "Word and object MIT press," *Cambridge MA*.

[73] Matuszek, C. (2018) "Grounded Language Learning: Where Robotics and NLP Meet (early career spotlight)," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, pp. 5687–5691.

[74] D., S., H. T., S. J., A. I., L. M., G. A., L. M., L. Sifre, K. D., G. T., L. T., S. K., and H. D. (2017) "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *arXiv Reprint arXiv:1712.01815*.

[75] Silver, D., A. Huang, M. C. J., G. A., S. L., D. G. v. d., S. J., A. I., P. V., L. M., D. S., G. D., N. J., K. N., and S. I. (2016) "Mastering the game of Go with deep neural networks and tree search," *Nature*, **529**, pp. 484–489.

[76] Henderson, M., B. Thomson, and J. Williams (2013) "Dialog State Tracking Challenge 2 & 3 Handbook," *camdial. org/mh521/dstc*.

[77] Asri, L. E., H. Schulz, S. Sharma, J. Zumer, J. Harris, E. Fine, R. Mehrotra, and K. Suleman (2017) "Frames: A corpus for adding memory to goal-oriented dialogue systems," *arXiv preprint arXiv:1704.00057*.

[78] Rosset, S. and S. Petel (2006) "The Ritel Corpus-An annotated Human-Machine open-domain question answering spoken dialog corpus." in *LREC*, vol. 6, pp. 1640–1643.

[79] Bennett, C. and A. I. Rudnicky (2002) "THE CARNEGIE MELLON COMMUNICATOR CORPUS," *Seventh International Conference on Spoken Language Processing*.

[80] Brennan, S. E., K. S. Schuhmann, and K. M. Batres (2013) "Entrainment on the Move and in the Lab: The Walking Around Corpus," *Proceedings of the Annual Meeting of the Cognitive Science Society*.

[81] Heeman, P. A. and J. F. Allen (1995) *The TRAINS 93 Dialogues.*, *Tech. rep.*, Rochester University, Dept. of Computer Science.

[82] Asher, N., J. Hunter, M. Morey, B. Farah, and S. Afantenos (2016) "Discourse Structure and Dialogue Acts in Multiparty Dialogue: the STAC Corpus," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 2721–2727.

[83] Afantenos, S., N. Asher, F. Benamara, M. Bras, C. Fabre, L.-M. Ho-Dac, A. Le Draoulec, P. Muller, M.-P. Pery-Woodley, L. Prévot, et al. (2012) "An empirical resource for discovering cognitive principles of discourse organisation: the ANNODIS corpus," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pp. 2727–2734.

[84] Passonneau, R. J. and E. Sachar (2014), "Loqui Human-Human Dialogue Corpus (Transcriptions and Annotations)," Columbia University Academic Commons.

[85] Kingma, D. P. and J. Ba (2014), "Adam: A Method for Stochastic Optimization," 1412.6980.

[86] Grice, H. P. (1975) "Logic and conversation," in *Syntax and semantics 3: Speech acts*, Academic Press, pp. 41–58.

[87] Gašić, M. and S. Young (2014) "Gaussian processes for POMDP-based dialogue manager optimization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **22**(1), pp. 28–40.

[88] Sutton, R. S. and A. G. Barto (1998) *Introduction to Reinforcement Learning*, 1st ed., MIT Press, Cambridge, MA, USA.

[89] van den Haak, M., M. D. Jong, and P. J. Schellens (2003) "Retrospective vs. concurrent think-aloud protocols: Testing the usability of an online library catalogue," *Behaviour & Information Technology*, **22**(5), pp. 339–351.

[90] Litman, D., M. Swerts, and J. Hirschberg (2006) "Characterizing and Predicting Corrections in Spoken Dialogue Systems," *Computational Linguistics*, **32**(3), pp. 417–438.
URL `https://www.aclweb.org/anthology/J06-3004`

[91] Bohus, D. and A. I. Rudnicky (2009) "The RavenClaw dialog management framework: Architecture and systems," *Computer Speech & Language*, **23**(3), pp. 332–361.

[92] Passonneau, R. J., S. L. Epstein, T. Ligorio, J. B. Gordon, and P. Bhutada (2010) "Learning about voice search for spoken dialogue systems," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 840–848.

[93] Das, R. and H. Pon-Barry (2018) "Turn-taking strategies for human-robot peer-learning dialogue," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 119–129.

[94] Gravano, A. and J. Hirschberg (2011) "Turn-taking cues in task-oriented dialogue," *Computer Speech & Language*, **25**(3), pp. 601–634.

[95] Perera, I., J. Allen, C. M. Teng, and L. Galescu (2018) "A situated dialogue system for learning structural concepts in blocks world," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 89–98.

[96] Chu-Carroll, J. (2000) "MIMIC: An adaptive mixed initiative spoken dialogue system for information queries," in *Sixth Applied Natural Language Processing Conference*, pp. 97–104.

[97] Yang, F. and P. A. Heeman (2007) "Exploring initiative strategies using computer simulation," in *Eighth Annual Conference of the International Speech Communication Association*, pp. 1069–1072.

[98] Wolff, A. S., D. H. Mitchell, and P. W. Frey (1984) "Perceptual skill in the game of Othello," *The Journal of psychology*, **118**(1), pp. 7–16.

Vita

Maryam Zare

Education

2016 – 2021	Ph.D. Pennsylvania State University Computer Science and Engineering
2012 – 2016	B.Sc. Sharif University of Technology Computer Engineering

Work Experience

Summer 2020	Pinterest INC. Research Intern at Pinterest Labs

Publication

- **M Zare**, A Ayub, A Liu, S Sudhakara, A Wagner, R Passonneau, *"Dialogue Policies for Learning Board Games through Multimodal Communication"*, Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 2020.
- **M Zare**, A Ayub, A Wagner, R Passonneau, *"Show me how to win: a robot that uses dialog management to learn from demonstrations"*, Proceedings of the 14th International Conference on the Foundations of Digital Games (FDG), 2019.
- N Rohbani, Z Shirmohammadi, **M Zare**, S Miremadi, *"LAXY: A location-based aging-resilient Xy-Yx routing algorithm for network on chip"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017.
- A Mirhosseini, M Sadrosadati, **M Zare**, H Sarbazi-Azad, *"Quantifying the difference in resource demand among classic and modern NoC workloads"*, IEEE 34th International Conference on Computer Design (ICCD), 2016.

Teaching Experience

Teaching assistant for

Spring 2021	Fundamental of C programming
Fall 2020	Computer Security
Fall 2018	Computer Vision
Fall 2016	Computer Organization And Design