Learning Hierarchical Compositional Task Definitions through Online Situated Interactive Language Instruction

by

James R. Kirk

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2019

Doctoral Committee:

Professor John E. Laird, Chair
Professor Joyce Y. Chai
Associate Professor Odest Chadwicke Jenkins
Professor Richmond H. Thomason

James Kirk

jrkirk@umich.edu

ORCID iD: 0000-0002-2459-3643

Acknowledgements

I would like to thank the many people that have mentored, helped, collaborated, and supported me over the past years of graduate school, especially my advisor, John Laird, my committee, fellow graduate student in the Soar lab, contributors to the Rosie project, and my friends and family.

TABLE OF CONTENTS

List of Tables

List of Figures

Abstract

Artificial agents, from robots to personal assistants, have become competent workers in many settings and embodiments, but for the most part, they are limited to performing the capabilities and tasks with which they were initially programmed. Learning in these settings has predominately focused on learning to *improve* the agent's performance on a task, and not on learning the actual definition of a task. The primary method for imbuing an agent with the task definition has been through programming by humans, who have detailed knowledge of the task, domain, and agent architecture. In contrast, humans quickly learn new tasks from scratch, often from instruction by another human. If we desire AI agents to be flexible and dynamically extendable, they will need to emulate these learning capabilities, and not be stuck with the limitation that task definitions must be acquired through programming.

This dissertation explores the problem of how an Interactive Task Learning agent can learn the complete definition or formulation of novel tasks rapidly through online natural language instruction from a human instructor. Recent advances in natural language processing, memory systems, computer vision, spatial reasoning, robotics, and cognitive architectures make the time ripe to study how knowledge can be automatically acquired, represented, transferred, and operationalized. We present a learning approach embodied in an ITL agent that interactively learns the meaning of task concepts, the goals, actions, failure conditions, and task-specific terms, for 60 games and puzzles. In our approach, the agent learns hierarchical symbolic representations of task knowledge that enable it to transfer and compose knowledge, analyze and debug multiple interpretations, and communicate with the teacher to resolve ambiguity. Our results show that the agent can correctly generalize, disambiguate, and transfer concepts across variations of language descriptions and world representations, even with distractors present.

Chapter 1 Introduction

If an agent is going to acquire new tasks, we must consider the different ways that tasks can be defined or formulated. Some tasks can be defined strictly by a procedure to follow, such as following a recipe or directions, where the teacher already knows the correct procedure. In contrast, we are interested in tasks that are defined by the legal actions that can be taken and the goals to be achieved, such as games (Tic-Tac-Toe), puzzles (Sudoku), or other discrete goal-oriented tasks (sorting, cleaning). We adopt Newell's proposal that goal-oriented tasks can be formulated as problem spaces (Newell, 1980). In the problem space computational model, a task is represented by an initial state, a goal state, and the constraints (preconditions) of available operators. The task may also include failure conditions, or illegal states. This research focuses on games and puzzles, which provide a large set of goal-oriented problems, but also require a large variety of concepts and capabilities. This thesis is an exploration of what is required to support learning across these types of diverse tasks and concepts, while taking advantage of the constraint provided by the problem space formulation for goal-oriented tasks.

Specifically, we are looking at how an agent can learn all the elements of a task (such as Tic-Tac-Toe) so that it can acquire and then perform the task. These **task elements** that the agent must learn to recognize and operationalize include the *goals* (three in a row), *failure conditions* (opponent three in a row), *legal actions* (marking an empty square), and *task-specific terms* (empty) used to define a task. We define operationalizing a task element as applying the recognized task element in the current environment. For example, operationalizing an action task element means proposing available actions. Rather than require an agent to be preprogrammed with this knowledge, we explore how the agent can learn the task interactively and online, drawing inspiration from how a human novice might learn a novel task from an expert teacher. Specifically, we enable an instructor to provide

natural language descriptions to the agent, sentence by sentence, of the task elements grounded in a shared situated example of the task, where each described task element is present in the external environment. The description of a task element includes conditions that must be true for that goal, action, failure condition, or term to be applicable to the current situation.

In many environments, the problem of recognizing and applying task elements is straightforward because there is an assumption of fixed vocabulary usage and representations of external environments. For example, the term "empty" will only ever have a single meaning: there is only one way "empty" maps to objects in the environment, without any consideration that the meaning of an "empty" square might vary for different versions of Tic-Tac-Toe, one where squares are marked with X's and O's and one where squares are covered with black and white stones. If a novel environment feature or word is encountered, it is assumed that it correlates directly with an internal symbol known a priori by the agent (Chai et al. 2016). For example, in these environments if the word "red" is used, learning it means finding a mapping to a pre-existing symbol, such as *R23*, that is linked to a classifier for the color red. In contrast, we do not make these assumptions: the language meanings and external world are not known a priori and there may not exist an internal symbol that directly corresponds with the vocabulary used. Therefore, the agent must learn how novel terms ground to the external world, which may require a combination of existing features and knowledge that the agent already knows.

Learning how to recognize and operationalize novel task elements is challenging given these assumptions. Characteristics of this learning problem that make it difficult include (**1**) the inability of the agent to see into the mind of the teacher and know what their words mean, (**2**) the multiplicity of possible words and meanings, (**3**) the complex compositional definitions of concepts and how they connect to the world, and (**4**) the accumulation of knowledge overtime. To avoid learning from 'scratch' each time, and

reduce teaching time, the agent should be capable of transferring knowledge from other tasks while it is accumulating knowledge over many tasks. However, accurate knowledge transfer is difficult given the other characteristics, which can cause negative transfer. The inability of the agent to know the intended meanings of a word, given the multiplicity of possible meanings, can lead to the incorrect meanings being used in a new task. These problem characteristics are discussed below in detail in Section 1.1.

The result of this work is a learning methodology embodied in an agent that can start with no task or domain specific knowledge, enter a novel setting with a teacher, and learn, through quick interactions with context-specific language, to recognize all the task elements (actions, goals, failure, etc.) and then use them to solve the task. Furthermore, this agent can use the learned knowledge on future tasks, transferring relevant knowledge, and handling ambiguous scenarios where knowledge interference and/or distractors are present.

1.1 Problem Characteristics

There are many characteristics of this problem that make it challenging for an agent to effectively learn many tasks quickly in succession, over changing context-specific terms, in disparate environments. These characteristics and the accompanying challenges are highlighted with the example depicted in Figure 1, where a human instructor and robotic agent are situated in a shared environment that contains an instance of a river crossing problem, involving actors and their managers who don't want their clients to be poached. The task is to ferry everyone from one bank to another, with at most two in the boat, and without upsetting any managers by having one of their clients on the other bank with another manager.

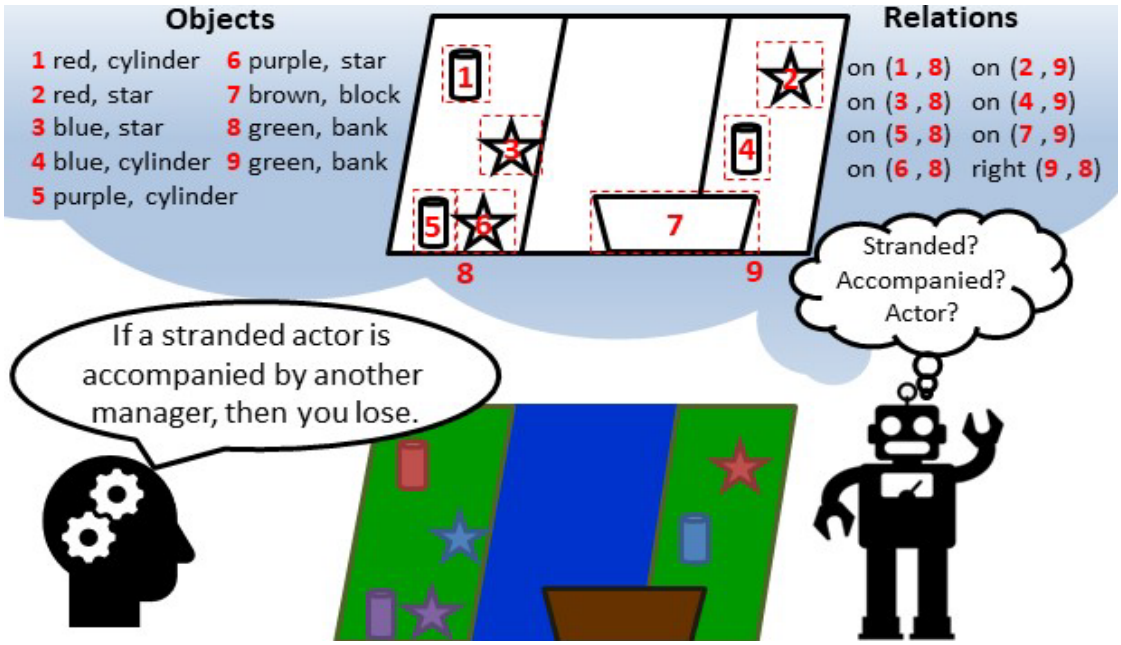


Figure 1: A depiction of an instructor attempting to teach a situated version of a river crossing problem and the agent's internal model derived from its perception and knowledge.

An example shared environment for this task is depicted on the bottom of Figure 1. The task environment, shown between the instructor (left) and agent (right), consists of blocks and surfaces, with different colors and shapes that represent actors, managers, the boat, the river, and the river banks. The green surfaces represent the banks with the blue surface between them representing the river. The brown block is the boat and cylinders and stars are the managers and actors, respectively, waiting on the banks. Each manager-actor pair shares a single color, red, blue, or purple. In the figure, the instructor in attempting to teach the failure condition of the task. This is just one possible embodiment for this task.

The agent's internal model that it derives from perception are depicted at the top of the Figure 1. The agent's detection of objects is illustrated by bounding boxes with unique identifiers (numbers 1-9) for each object. The agent's symbolic representations that it extracts from perception are listed to each side. The unary features detected from the

objects, such as *red*, are listed by their unique identifier to the (top) left. The spatial relations detected between those objects are listed as predicates where the arguments are the object identifiers, such as *on(1,8)* for the relation detected between the brown block (the boat) and the bank on the right. The agent does not initially know how the terms used by the instructor, "manager," "actor," "stranded," and "accompanied," map to its representation of the environment; it only has general knowledge that is not task-specific.

We have identified four major problem characteristics (**C1**-**C4**) that create challenges for learning:

C1: Lack of Common Ground

The first problem characteristic is that the human instructor and the agent do not have access to each other's internal representations: they lack *common ground* (Clark & Brennan, 1991). The teacher is not guaranteed to know or use terms that correspond to the agent's knowledge. In Figure 1, when the instructor describes the failure condition as "if a stranded actor is accompanied by another manager, then you lose," the instructor uses task-specific terms ("stranded," "accompanied," "actor," "manager") that may not exist in the agent's internal model of the current environment. To detect the failure condition, the term "accompanied" must be mapped to a calculation of *on the same bank*, the term "stranded" to *on a bank without their manager*, "actor" to an object *having a star shape*, and "manager" to an object *having a cylinder shape.* These mappings can connect to not only symbolic primitives; the agent can also learn a mapping to nonsymbolic classifiers, such as classifiers for color, shape, or size.

Chai et al. (2016) discuss in detail the problems that arise from the mismatch between the perceptual basis and capabilities of a robotic agent and human teacher and explore how an interactive agent can collaborate with the teacher to help resolve these mismatches. Task learning approaches often avoid this problem by assuming

that the environment, terminology, or agent knowledge and capabilities are fixed, or only vary in limited, predictable ways. This is a very constraining assumption to make, especially for robotic applications with unknown environments.

C2: Compositional Concepts

A second characteristic is that task elements (actions, goals, terms) can be defined by the instructor using compositions of other task elements (that have been learned previously in the task or other tasks). These compositions can involve conjunctions, disjunctions, modifiers, functions, set operations, and other elements described by First Order Logic as well as hierarchical compositions. Compositionality enables the learning of a theoretically infinite number (bounded only by time and memory) of concepts from a set of primitive elements (Fodor & Pylyshyn, 1988).

Approaches to task learning have generally considered learning the different elements of a task (actions, goals, failure conditions, task-specific terms) independently: both in terms of the learning algorithms used and the final representations that are learned. They don't consider how these various elements can be combined using first order logic or hierarchical composition. Rather than learning hierarchies and maintaining intermediate representations that can be composed, they learn mappings directly to subsymbolic sensorimotor representations (Socher et al., 2013; Bhargava et al., 2016; Chauhan & Lopes, 2011; Dindo & Zambuto, 2010; Orhan et al., 2013).

Through the composition of concepts, the agent can learn definitions of failure conditions using new terms, such as with "stranded" and "accompanied" in Figure 1, definitions of goals using actions (*capturable)*, and definitions of new task-specific terms using previously defined task-specific terms. For example, the teacher could define "stranded" by "if the manager of the actor is not on the same bank then the actor is stranded," but then would need to define "manager of" as well: "if a manager

has the same color as the actor then…" The learning of hierarchical compositions of these task elements is vital to supporting the types of high-level definitions that humans often use. Furthermore, supporting hierarchical representations of learned knowledge has implications for the accumulation (**C4**) and transfer of knowledge. With hierarchical, compositional representations, compared to direct mappings to nonsymbolic representations, it is easy to replace component parts to achieve partial knowledge transfer, such as learning a new grounding of "actor" for the failure condition, without having to relearn the entire failure condition.

C3: Many-to-many Mappings

A third characteristic is that the possible mappings from terms to definitions are many-to-many, due to the polysemic nature of words (a word can have many possible meanings and a meaning can be represented by many possible words) and due to variations in external environments. For example, the term "clear" can be used to mean not below anything. In another situation it can be used to mean transparent in opacity, and in another, the term "empty" can also be used instead of "clear" to mean not below anything. The required mappings from terms to meaning change depending on the terminology used, how the instructor decomposes the task elements, the setting of the problem in the external environment, and the agent's perception and prior knowledge.

Consider all the variations that could occur in the problem depicted in Figure 1. For task-specific terms, the teacher could have used the word "separated" instead of "stranded." For the task element decomposition, the teacher could instead define the failure condition by saying "If an actor is poachable then you lose." For the external environment setting, instead of *stars* and *cylinders*, the "actors" and "managers," respectively, could be represented with different objects, such as *cubes* and *cones*. For the agent's perception and knowledge, the agent could possess spatial knowledge

of *below* rather than *on* or only have knowledge of the x, y, z coordinates of each object with no known spatial prepositions. Given all these variations, the space of possible mappings from the task elements to their external embodiments is huge. If the desire is for an agent *X* (with unknown knowledge and capabilities), operating in an environment *Y* (with unknown representations), to learn and do a task *Z* (with unknown terms), this will be a significant, unavoidable, and reoccurring problem.

C4: Accumulative Learning

A fourth characteristic is that the agent should be able to accumulate many tasks (and task elements) over the learning of many different problems in different settings. A key to reducing task teaching time during accumulative learning is the ability to reuse, or transfer, knowledge previously learned to new tasks and avoid learning from scratch each time. However, this introduces the problem of potential knowledge interference from past learning: the possibility of incorrect knowledge transfer or overgeneralization. For example, due to the many possible meanings of words (**C3**), the agent could learn a meaning of the term "empty" for Tic-Tac-Toe (meaning unmarked) and then incorrectly use that definition when the term is used to define a subsequent task, such as Othello, where the meaning is different (not below a piece). With the accumulation of many tasks, correctly inferring what knowledge can be transferred will become increasingly difficult.

Consider that after learning the problem in Figure 1, the agent is asked to solve a identical puzzle, but one that uses *cubes* and *cones* to distinguish "managers" and "actors" respectively. Like a human student, the agent should adapt to the new instance by transferring all the task definition knowledge from the previous incarnation, rather than relearning all the goals, actions, and constraints of the puzzle, and only learn new mappings for "managers" and "actors" to *cubes* and *cones*. If a new river crossing puzzle is taught that only shares a goal ("all objects are on the

right bank") or an action ("move the boat"), the agent should transfer the relevant knowledge and only learn what is novel. Consider instead that the agent is presented with a similar but different river crossing problem to learn. If the puzzle is setup using the same objects (*cylinders* and *stars*), the agent might incorrectly infer that they are "managers" and "actors," or if the term "accompanied" is used in a different manner by the teacher, the agent might attempt to transfer the concept. The challenge for an ITL agent is to transfer as much knowledge as possible without overgeneralization, avoiding incorrect knowledge transfer.

Existing approaches to task definition learning do not have a theory for how knowledge transfer can occur because they don't maintain intermediate knowledge representations. Most have focused on learning mappings directly from task terms to subsymbolic sensorimotor representations (Roy, 2002; Socher et al., 2013; Bhargava et al., 2016; Yürüten et al., 2013; Chauhan & Lopes, 2011; Dindo & Zambuto, 2010; Orhan et al., 2013; Matuszek et al., 2012). Other approaches (Hinrichs and Forbus, 2014) don't learning mappings but instead compile the natural language specification into a high-level programming language. Neither approach has explored how partial knowledge transfer can occur, such as when a new task variation uses pieces of a different shape or color.

1.2 Learning Example

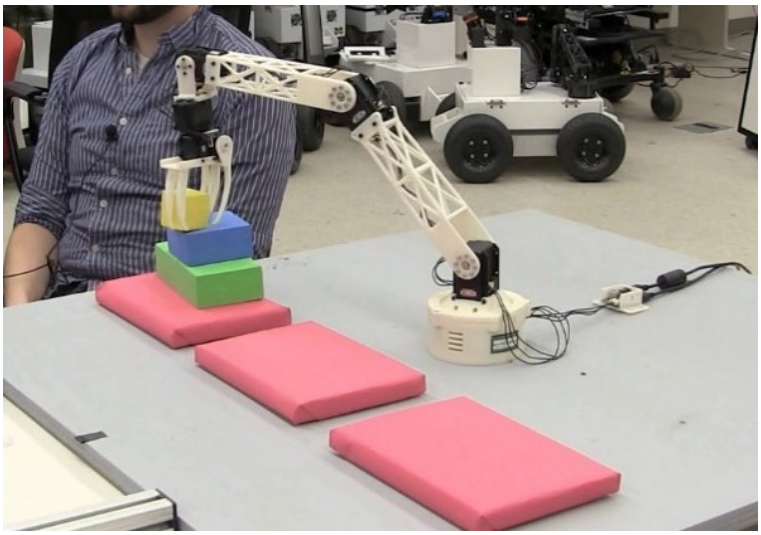


Figure 2: A tabletop environment with blocks and a robotic arm used for teaching a version of the Tower of Hanoi puzzles.

Figure 3 shows an example dialogue between our agent (marked in **bold**) and an instructor, where the agent, embodied in a tabletop robot, is taught a version of the Tower of Hanoi puzzle using blocks. The tabletop environment is shown in Figure 2.

"The name of the puzzle is Tower-of-Hanoi."

"Please setup the puzzle."

***** Teacher moves blocks to setup the initial state. *****

"Ok."

"Please describe the actions, goals, and failure conditions."

"You can move a clear block onto a clear object that is larger than the block."

"Please describe the meaning of clear in this context. "

"A location that is not below an object is clear."

"Ok, Ive learned the meaning of clear for this context."

“Please describe the meaning of larger -than in this context. ”

“If the volume of a block is more than the volume of an object then the block is larger than the object.”

“Ok, Ive learned the meaning of larger -than for this context. ”

“Ive learned the action.”

“The goal is that a small block is on a medium block and the medium block is on a large block and the large block is on the leftmost location.”

“Please setup the goal state.”

***Teacher moves blocks to setup the final state. ***

“Ok.”

“Ive learned the goal .”

“Done.”

“Ive learned Tower-of-Hanoi. Should I try to solve the puzzle?”

Figure 3: Dialogue of an instructor teaching Rosie the Tower of Hanoi puzzle using blocks.

The agent quickly learns the *task elements* that define this task: preconditions of the legal moves (*can move blocks onto larger objects if both are clear*) and the goal of the puzzle (*three blocks stacked smallest to largest on the leftmost location*), along with supporting context specific terminology (“clear” and “larger than”). This learning process is summarized in the following section and described in detail in Chapter 4.

1.3 Learning Approach

In our approach, there is a single integrated agent that quickly learns new tasks online, in real time from an instructor by learning all of the task elements that define the task. A key aspect of our approach is that all task elements, goals, failure conditions, actions, and task-specific terms, are represented uniformly and are learned through the same process. Thus,

to describe the learning of the entire task in our approach, we just describe the process of learning a task element. For most task elements, the *actions*, *goals*, and *failure conditions*, the teacher initiates the start of the learning process by teaching the name of a task element ("The name of an *action* is stack") or by starting a new description ("The *goal* is that…"). In contrast, the agent initiates the start of the task element learning process for a task-specific term. This occurs when the term is used to define another task element and the agent cannot determine the correct meaning (or has none): "Please describe the meaning of 'clear' in this context." The teacher initiates the learning of a new task by naming it: "The name of the puzzle is Tower-of-Hanoi."

We decompose the process of learning a task element through interactive language instruction into four learning phases:

L1: Internal Model Creation

In order to interact with and reason about the outside world, the agent uses its perception and existing knowledge to create an internal relational model of the current state of the environment, including its beliefs about the objects and relations between them. In order to learn a task element, the task element being described *must* be currently present in the world so that the agent can ground the terms to its internal state. An example of an internal model is depicted in Figure 1 for a river crossing task, showing the objects the agent observes and their relations (*the boat is on a bank*). It is possible that the agent could create this internal state through another method, such as having it described (as in most word problems), but this has not been explored in this thesis.[1] It vital to our learning approach that the

[1] Parallel work (Mininger & Laird, 2018) has enabled the agent to handle partially observable scenarios when learning procedural tasks (not games) where the goal is not currently present.

described concept can be grounded in an accurate internal model that contains the concept, but the source of that model could vary.

L2: Language Instruction

Next the agent converts the teacher's natural language sentences that describe each task element into an internal relational representation. This relational representation contains all the conditions that need to be true for the task element to be applicable in the current situation. For example, if the teacher describes "pinned" with the sentence "if a piece is blocking your king, then it is pinned" then the agent creates a representation equivalent to *blocking(piece, king) →pinned(piece)*.[2]

L3: Recognition Structure Learning

In order to recognize the task element in the environment, the agent creates a declarative structure that connects the language-derived relational representation to the agent's internal model. The agent grounds each referenced term (i.e. *blocking*, *piece*, *king*) so that the conditions of the task element can be evaluated. For example, the conditions for an action *stack* might be that the *destination is clear* and the *block is movable*, which are evaluated in order to determine valid actions. This process includes verifying that the task element is applicable in the current context. If the agent is unable to ground a term in its internal model, the agent asks the teacher for a definition (the meaning) of the new task-specific term, another new task element. This causes the learning to loop recursively back to **L1** for this new task element. (However, the world should already contain an instance of the term, so the agent will not need to request that the environment be modified to setup

[2] Throughout this paper we use quotes ("pinned") to denote terms used in language descriptions and italics (*pinned*) to denote elements of the agent's internal state.

a relevant state.) During this process the agent learns procedural rules that condense the agent processing during interpretation into procedural knowledge that directly evaluates the declarative recognition structure.

L4: Operationalization

In order to operationalize the task element, the agent instantiates this knowledge for the current situation in its internal model of the task environment. The exact form of the operationalization depends on the type of task element; whether it is a goal, action, failure, or task-specific term. The instantiation of an action, such as *stack*, involves proposing the primitive actions that were used to teach it, such as *move obj1 onto obj2,* using the objects in the world that satisfy the preconditions. The agent only instantiates a task-specific term, such as *pinned*¸ when it is a condition of another task element. The result of operationalizing a task-specific term is returning the detected instances of that term in the current context, in order to evaluate the conditions of the task element using the task-specific term.

After learning all the task elements for a new task, the agent uses the acquired knowledge for recognizing and operationalizing each task element to perform each associated task function and attempt to solve the problem through search or other problem-solving strategies. Thus, in order to solve the newly learned task, the agent creates an internal model of the current situation from its perception of the environment, and applies the task elements to: detect and propose legal action actions, generate the resulting substates using primitive action models, avoid failure states, and search until the goal is recognized. Building the internal state (**L1**), language processing (**L2**), and problem solving (search) have been implemented in prior research (Mohan et al. 2012; Kirk & Laird, 2016). This thesis focuses on the third and fourth phases (**L3** and **L4**), the problem of learning a

connection between the described task elements and the agent knowledge required to recognize and apply them.

1.4 Desiderata

To qualitatively evaluate the agent and provide guidance for agent design, we have developed a list of desiderata. These desiderata may not be requisites for every ITL system, but they are general principles that can guide the development and common evaluation of such systems, which are inherently difficult to compare. We have described a larger space of possible desiderata for Interactive Task Learning agents (Laird et al., 2017), but here we focus on four major desiderata for ITL agents that serve as a basis for the goals, claims, subsequent evaluation of our research on learning task elements. These desiderata focus on minimizing or maximizing certain aspects of task learning, while maintaining correctness of the learning. To be clear, the objective is reducing or increasing these ITL aspects, not finding a theoretical maximum or minimum.

D1: Maximize Generality

The agent should be able to learn a diverse set of tasks, requiring a diverse set of concepts (objects, relations, actions). The level of generality of a task learning agent increases as it becomes capable of learning more tasks and more kinds of tasks. Meeting this challenge requires avoiding task specific representations or processing mechanisms.

D2: Minimize Agent-Teacher Communication

The communication of the task should be concise, avoiding repetition, verbosity, and unnecessary communications. The agent should try to minimize the number of words, interactions, and demonstrations, as well as the overall time, required to

learn a new task, to reduce the burden on the teacher and speed the learning process. The learning must be fast, online, and in real-time.

D3: Minimize Agent Execution Time

The agent should try to learn task representations that minimize execution time, by learning procedural representations that do not need to be interpreted to be executed. Their execution should be similar in behavior to the execution of the code of the underlying agent architecture. The newly learned knowledge must integrate, and not interfere, with existing behavior and any future knowledge that might be learned.

D4: Minimize Memory Size Growth

The agent should minimize the growth of memory (working, procedural, and semantic) to avoid negative (slow) memory behavior. The agent should minimize the time required to access or add memory elements, the size of the memory, and their growth overtime as the agent learns many successive tasks. We expect different memory systems to have different growth patterns, the objective is to avoid excessive, harmful memory growth which negatively impacts agent behavior.

1.5 Contributions

We present our learning approach embodied in an agent called Rosie, an interactive task learning robotic platform built on the Soar cognitive architecture. A key aspect, and contribution of our approach is identifying the underlying structure and similarities in this learning space across the different types of task elements in order to learn them through a singular process into a single type of representation. This uniformity of representation across task elements supports the compositional nature (**C2**) of task knowledge. The ability

to learn composable representations is a critical feature to enabling better, long-term knowledge transfer during accumulative learning (**C4**). Using this approach, Rosie can learn an entire problem definition from the ground up, by using the knowledge that the agent already has and the definitions the human provides, despite a lack of knowledge overlap (**C1**), learning new task elements based on their word choice and task decomposition, learning a multiplicity of words and meanings (**C3**), and transferring knowledge over many tasks, even in ambiguous scenarios, as the agent accumulates knowledge (**C4**).

We evaluate how well how our approach addresses the challenges associated with each of the aforementioned problem characteristics. We show that our learning approach works independent of changes to the terminology used, agent knowledge representations, and environments by learning new definitions to establish common ground (**C1**). We demonstrate that Rosie can learn, through interactive instruction, compositions of existing concepts through hierarchical composition (**C2**) and can learn a multiplicity of terms and meanings for the many-to-many possible mappings between words and meanings (**C3**). We evaluate Rosie in online, accumulative long-term learning scenarios and Rosie's ability to transfer knowledge over many tasks (60) and domains, including scenarios with distractors and knowledge interference (**C4**).

We evaluate our approach across a wide variety of tasks (**D1**), from Tic-Tac-Toe to Sudoku, through quick natural language communication (**D2**). We also evaluate the execution, or processing, time of the agent during and after learning (**D3**), and the agent's growth in memory size (**D4**) as it accumulates tasks.

In brief, the core contributions of this thesis are:

1. A general, online method for learning mappings between task elements (actions, goals, failure conditions, and task-specific terms) and hierarchical compositions of the agent knowledge required to recognize and operationalize them. This learning

method enables an ITL agent to gain the necessary knowledge to formulate and solve a novel task in a novel environment.

2. A strategy for handling ambiguous learning situations, where distractors and/or knowledge interference are present, that enables Rosie to consider multiple possible interpretations and interact with the instructor to resolve ambiguities and accurately transfer knowledge.
3. A characterization of the space of learnable tasks and task elements through this approach. From this characterization, it should be possible to determine whether Rosie is capable of learning a specific novel game, task, or problem.
4. An evaluation of these methods over many tasks for each of the given desiderata. This evaluation covers agent-teacher communication (sentences used to teach the task), the growth in size of semantic, procedural, and working memory, the timing of agent execution, and the handling of ambiguous learning scenarios that could lead to negative knowledge transfer.
5. An evaluation of the generality of the learning approach across different tasks (puzzles, games, logic problems), domains (cards, board games, marking puzzles, computer games), environments (real and simulated), language usage (terminology), task decompositions, and agent embodiments (virtual, robotic).
6. A publicly available archive (www.umich.edu/~jrkirk/ijcai2019.html) of all the games (and task elements) learned and the sentences, world environments, and primitive concepts used to each them. A list of these games is shown in Table 2 in the Appendix, including examples of games that cannot be learned and the reasons why. This will be a resource for other researchers investigating Interactive Task Learning.

1.6 Outline

In **Chapter 2** we discuss the related work and briefly contrast those approaches with our own approach.

In **Chapter 3** we describe some background on the development of Rosie and supporting work.

In **Chapter 4** we provide a detailed description our task-independent approach to learning to recognize and operationalize all types of task elements, the actions, goals, failure conditions, and task-specific terms. During this description we note how the design decisions were motivated by the problem characteristics (**C1**-**C4**) and desiderata (**D1**-**D4**).

In **Chapter 5** we provide a detailed description of a complex example of the agent learning an entire task, the problem from the introduction, as well as examples of a variety of task elements that agent has learned.

In **Chapter 6** we describe evaluations of task learning and knowledge transfer we have conducted using the desiderata as criteria. These experiments have been performed in a variety of domains, learning many successive tasks, where we analyze the memory growth (agent knowledge), timing (agent processing), interactions (teacher natural language descriptions), knowledge correctness, and the accuracy of the solutions the agent produces after learning the complete task. Our results show that the agent can correctly generalize, disambiguate, and transfer concepts within variations in language descriptions and world representations of the same task, and across variations in different tasks. Rosie has successfully learned 60 distinct games and puzzles, from Missionaries and Cannibals to Othello to Sudoku, and thousands of actions, goals, failures, and terms.

In **Chapter 7** we describe handling scenarios where ambiguity and knowledge interference can negatively impact the ability to accurately learn and transfer knowledge, by enabling Rosie to create, analyze, and debug (through interactions with the teacher) *multiple* interpretations of task elements. We evaluate the correctness of learning and the number of words required to teach tasks across cases of no transfer, positive transfer, and interference from prior tasks.

In **Chapter 8** we summarize the contributions of the thesis and discuss the potential impact of the work, limitations of the work, and possible directions for future work.

Chapter 2 Related Work

In this chapter we review related research, starting with research on learning the rules of games, followed by approaches that support task execution and specification, but not incremental acquisition, to work that supports learning the grounding of novel terms, and finally ending by discussing how these approaches contrast with ours.

2.1 Learning the Rules of Games

Early research on understanding natural language descriptions of games was conducted by Simon and Hayes (1976) in a program called UNDERSTAND, which extracted the actions, or operators, and goals from natural language specifications of various isomorphism of the Tower of Hanoi puzzle. puzzle. They showed that differences in task specification led to differences in task formulation, both in humans and with UNDERSTAND. This was an early attempt at task specification through language, however a complete agent implementation was not implemented; the language translation and concept learning processes were simulated by hand.

Hinrichs and Forbus (2014) have developed an agent in the Companions architecture that learns to play Tic-Tac-Toe and Hexapawn through a combination of language instruction and sketching using CogSketch. Their system generates a GDL (Game Description Language) specification (Love, Hinrichs, Haley, Schkufza, & Genesereth, 2008) of the task, which is interpreted so that a Companions agent can play the game. Their approach focuses more on the naturalness of agent-teacher interaction than learning new concepts or maximizing the generality of task learning (**D1**). They do not have a theory for task knowledge transfer. GDL is a high-level program-like formalism that lets one specify

a large variety of games; it is used in the General Game Playing competition (Genesereth & Love, 2005). Research on transfer of knowledge with agents that use GDL specifications has so far focused only on policy transfer (Banerjee & Stone, 2007).

We have identified the following issues with the approach of converting natural language into an intermediate formal language, such as GDL or STRIPS/PDDL, and then using an interpreter to produce behavior. This approach does not include a theory of how the acquisition and execution of a new task and new task knowledge fits into the overall ongoing agent operation. These languages, and their interpreters, assume batch operation of a single task where the complete task description is available all at once. This approach does not support the accumulation of multiple tasks, nor the sharing of knowledge among tasks, nor interactive learning where the agent asks for clarification and missing knowledge, so that the instructor does not need to know exactly what must be taught.

In a very different approach, Barbu et al. (2010) describe a robotic system that learns to play simple 3×3 board games, like Tic-Tac-Toe and Hexapawn, by observing random legal game play between two other agents. Kaiser (2012) makes many improvements on learning board game rules through visual observation by reducing the amount of pre-coded background knowledge and using more expressive representations of state. This system represents the game state with relational structures, instead of formulas, but these structures are predefined, namely rows, columns, and diagonals in the board grid. Approaches using visual observation can be effective at successfully learning tasks, as well as learning competent game play, but they require a large number of demonstrations, including in some cases labeled illegal game play, in order to learn.

2.2 Task Specification Languages

Task learning research has usually focused on learning a procedure to follow or a policy. In other prior work on agents learning tasks through instruction, such as by Langley et al. (2010) and Allen et al. (2007), a human teaches a specific policy or procedure for solving

a task, such as giving directions to a location. Rather than teaching the problem specifications, these approaches teach a solution to a problem, which is sufficient for tasks where there is a single, fixed solution or policy for all problem instances (and not goal-oriented tasks such as games and puzzles). Policy task learning has been explored with Soar in the past, with Instructo-Soar (Huffman & Laird, 1995) an agent that learned simple but novel block manipulation tasks from simple natural language instructions, and in the present, with parallel work (Mininger & Laird, 2018) conducted on Rosie to enable the learning of procedural tasks.

There have been many efforts to create abstract task specification languages that make it easier for a user to develop agents. The Task Acquisition Language or TAQL (Yost, 1993) was an abstract language based on the problem space model of computation that was compiled into Soar. Other task specification languages that compile into Soar include HERBAL (Cohen, 2008) and HLSR (Jones et al., 2006). Cohen (2008) provides an extensive review of these task specification languages. Langley et al. (2010) described a related approach in which an instructional command language allows the specification of behavior for agents in ICARUS. These projects all require independent batch systems that compile the task specifications into the target language.

Salvucci (2013) introduced another approach to cognitive skill acquisitions (with ACT-R) focusing on the integration and reuse of previous skill knowledge and the proceduralization of this knowledge. The commands are limited to a restrictive syntax that only specifies the policy of a task. Cantrell et al. (2012) describe a mobile robotic system that can be taught individual commands via language by specifying preconditions ("you are at a closed door"), action definitions ("you push it one meter"), and postconditions ("you will be in the room"). It does not learn new predicates, nor can it learn tasks that involve constraints, failure states, and specific goal conditions.

The Tailor system (Blythe, 2005) allowed the teacher to modify task information though natural language instructions and ensured that there were no undesirable side effects caused by the modifications. PLOW (Allen et al., 2007) is a collaborative task-learning

agent that acquires procedural knowledge through demonstration and dialog. The instructor provides tutorial instructions accompanied by related demonstrations, which the agent uses to acquire new procedural knowledge. The teaching is done largely by demonstration, but unlike most learning from demonstration, the human is not required to provide a large number of examples due to agent generalization mechanisms. LIA (Azaria et al., 2016) is an interactive agent that learn new commands for managing email through step-by-step instructions from a human in natural language. These systems learn new tasks or modifications to tasks but are limited the types of tasks they can learn: only tasks defined by sequences of actions.

2.3 Learning New Word Grounding

Research on learning the groundings of words in situated domains has focused on learning new symbols grounded directly in the robots subsymbolic sensorimotor representations. One exception is TRIG (Gold et al., 2009), a system that learns from primitive symbolic relationships though demonstrations. Agents that do learn groundings from existing symbolic concepts only learn synonyms; they assume that for any term there is a single matching concept with the identical meaning (Goldwasser & Roth, 2014). Many of these approaches use machine learning techniques and require a large number of demonstrated positive (and sometimes negative) examples of the described concepts. These methods have been successful at learning new adjectives, prepositions, and nouns by learning mappings directly from task terms to subsymbolic sensorimotor representations (Roy, 2002; Socher et al., 2013; Bhargava et al., 2016; Chauhan & Lopes, 2011; Dindo & Zambuto, 2010; Yürüten et al., 2013; Orhan et al., 2013; Matuszek et al., 2012).

Systems that can handle novel environment features or words, often assume that they correlate directly with an internal symbol known a priori by the agent, just that the mapping has not been established (Chai et al. 2016). For example, Thomason, Zhang, Mooney, and Stone (2015) describe an agent that incrementally learns a semantic parser

through interactive natural language in robotic domains, as well as an online domain that utilizes Mechanical Turk for training. Their system creates a lambda-calculus representation that includes predicates defined over objects. However, it only learns one-to-one mapping as the semantics for novel terms, assuming that a preexisting symbol is functionally a synonym. It does not learn new tasks; the focus is on learning a semantic parser so that the agent can understand and execute the procedural commands it is given.

Another common assumption of learning systems is that each term only has a single meaning, however some research has explored learning one-to-many mappings from terms to meanings for polysemic words. Thomason (2016) presents an extension to the previously mentioned work, that enables learning multi-modal classifiers for polysemic words, for example, learning classifiers for both "light," as in lightweight, and "light," as in light-colored. The system learns one-to-many mappings to nonsymbolic representations, not to combinations of symbolic knowledge.

2.4 Contrast with our Approach

In general, these approaches assume that the agent only learns a single task (rather than a sequence of many tasks), that concepts can be directly mapped to known primitives or subsymbolic representations, and that learning and acting are separate processes, where learning can be an offline batch process.

In contrast, in our approach, there is a single integrated agent that both acquires new tasks and executes those tasks, using a single underlying cognitive architecture. Furthermore, the agent learns uniform representations for the different aspects of the problem space, the actions, goals, failure conditions, and supporting concepts, so that they can be composed and combined to support continuous and accumulative learning. Task acquisition is just another task that uses the same memories, learning mechanisms, and decision procedures as are used for task performance. An advantage of this approach is that

whenever new task knowledge is acquired, it is immediately available for both the assimilation and execution of future tasks.

We focus on learning many tasks at once with instruction that is fast, interactive, and on-the-fly, and importantly builds knowledge over time, handling the many-to-many possible matchings between language, the external environment, and the agent's own knowledge representations. A key aspect of our approach is that the elements of the problem space learned are uniform in representation, logically composable, and teachable in hierarchical combinations that enable partial transfer. This allows the agent to opportunistically engage the teacher to fill in gaps, resolve discrepancies, and transfer already learned relevant knowledge.

Chapter 3 Background

Why are humans so good at quickly teaching and learning novel tasks? From observing human-human learning scenarios, where an expert teaches a student unfamiliar with the task, it is clear that in part it is because the teaching is: *interactive*, allowing for corrections to mistakes and incomplete models of their partner; *situated* in a shared grounded experience of the task with the expert; *multimodal*, using language, demonstrations, gestures, and other means to convey information; and *continuous* and *accumulative*, leveraging past knowledge. These aspects of human learning motivate our research into how an artificial agent can interactively and naturally learn tasks though task-oriented dialog. Although there are other means for teaching an agent a new task, task-oriented dialog is ubiquitous in human instruction and can be fast, effective, and natural for humans to use (Mohan et al., 2015).

The goal of Interactive Task Learning is to support the development agents that can learn novel tasks through natural interactions with a human teacher. An ITL agent will need to integrate many capabilities, including but not limited to natural language processing, dialog management, knowledge representation, memory, computer vision, spatial reasoning, and general problem solving. Integrated approaches provide a platform for studying how each process can be more directly informed by each other in the context of the current environment.

The agent used for this research and subsequent experimentation and evaluation is Rosie (Mohan et al., 2012), an Interactive Task Learning agent built on the Soar cognitive architecture (Laird, 2012), that learns through instruction and demonstration in a shared environment with a human teacher.

3.1 Rosie

Rosie is an ITL agent that relies upon task-oriented dialog with a human instructor to acquire new knowledge. It learns many aspects of tasks, including colors, shapes, sizes, spatial relations, procedural actions, and execution policy. Rosie has been under development by the University of Michigan Soar Lab since 2012, and the project has had the benefit of many collaborators in the Soar lab, as well as other robotic labs, that has made possible the development of the capabilities an ITL robot needs, including vision, actuation, reasoning, dialog management, and natural language processing.

Natural language is processed using a parser, implemented in Soar, that is integrated with Rosie. The parser takes as input grammatical English sentences and produces a semantic interpretation. The parser was developed by John Laird for Rosie, and was not a contribution of this thesis and so is only briefly described here. There are two modes for the parser that Rosie can select to affect the content of the semantic interpretation. In the first mode, the interpretation grounds all references to objects to specific objects in Rosie's perceptual system that match the conditions for the objects specified in the sentence. For example, "a red object that is on a bank" grounds to an object in Rosie's model of the world that is both red and on a bank. If there are multiple red objects that are on a bank, such as in Figure 1 with the river crossing puzzle, then one of the objects, such as the object identified with id **1** that is a *red cylinder*, is randomly selected.

The second mode of parsing is for when the agent does not want the parser to ground directly to any specific object, and instead just reflect the conditions in the language description. This is the parsing mode that is used for the work in this thesis that enables Rosie to learn games and puzzles. The reason for using this parsing mode for learning concepts for games is to help Rosie make correct generalizations. Rosie assumes that all the features the instructor mentioned are defining characteristics, such as "red," and those not included can be ignored, such as "cylinder." The descriptions of concepts for games are not directly referencing a specific object in the world, even though at least one valid instance of the described concept must be present for later grounding. In this mode the

parser creates an internal representation of a hypothetical object which contains predicates for all the constraints that were present in its description, in the above example “red,” “object,” “bank,” and “on.” The parser produces a relational representation that includes information derived from the linguistic structure of the sentence, such as specific relative clauses.

3.2 Environment

To demonstrate the generality of agent design of Rosie, it has been ported to multiple domains and embodiments, including four robots and many simulated environments. Pictures of some of these environments for different games are shown in Figure 4. The robots include a table-top robot arm (with a Kinect for sensing) that can manipulate variously colored and shaped foam blocks, an April MagicBot that can navigate hallways, a Fetch robot that can move and manipulate objects on a table, and most recently a Cozmo toy robot that can move and pick up small cubes. The simulated domains include the April Simulator of the table-top arm, the ROS simulator (Quigley et al., 2009) of a Fetch robot, an agent internal simulation for grid based puzzles (such as Sudoku), a simulated card game environment in an external Java application, and AI2-Thor (Kolve et al., 2017).

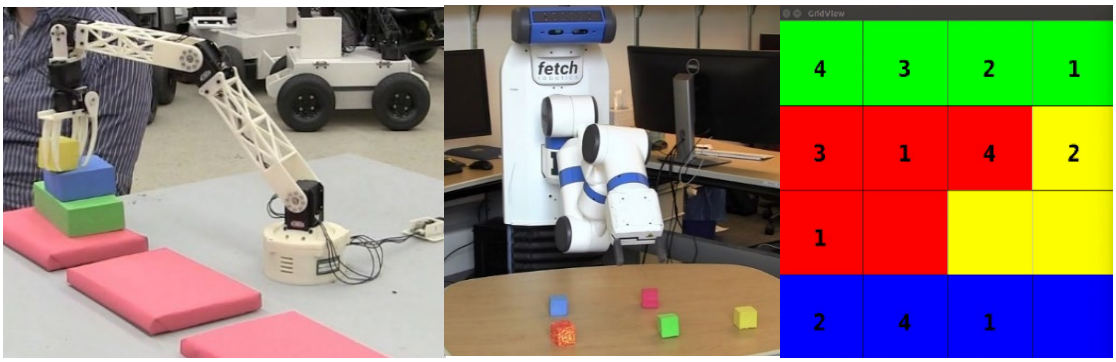


Figure 4: Pictures of different environments in which Rosie has learned games. From left to right: the tabletop arm solving Tower of Hanoi with blocks, the Fetch robot learning a block representation of the Five-Puzzle, and an internal simulation for learning the puzzle Ken-Ken.

The teaching of puzzles and games has mostly focused on the tabletop domain (real and simulated), and specifically on tasks that can be defined by movable objects, the locations those objects can occupy, and the relationships, functions, and attributes over those objects. For each domain, Rosie initially knows (and has internal models of) primitive domain-specific actions such as picking up a block (tabletop), navigating down a hallway (mobile robot), or writing a number onto a square (simulated grid).

In real world domains, Rosie translates the noisy continuous state produced by sensors into discrete symbolic states or representations of the world (Mohan et al., 2012; Mininger & Laird, 2019). For the learning of games and puzzles, we assume fully observable environments that can be mapped onto high-quality discrete symbolic representations that are not subject to noise or uncertainty.

3.3 Soar Cognitive Architecture

Since we are adopting the problem space model of problem representation, as well as attempting to integrate many capabilities, the Soar cognitive architecture (Laird, 2012), developed on the problem space model, is a natural choice for the development of an Interactive Task Learning agent. As Newell, one of the creators of Soar, defined it: "a problem space consists of a set of symbolic structures (the states of the space) and a set of operators over the space… A problem in a problem space consists of a set of initial states, a set of goal states, and a set of path constraints." (Newell, 1980)

Rosie is implemented in Soar, which has been applied to a wide variety of domains and tasks, including natural language understanding and robot control (Laird, 2012). Recent extensions to Soar, including episodic and semantic memories, as well as a visual-spatial system, enhance Soar's ability to support grounded language learning. Relevant components are described in the following paragraphs.

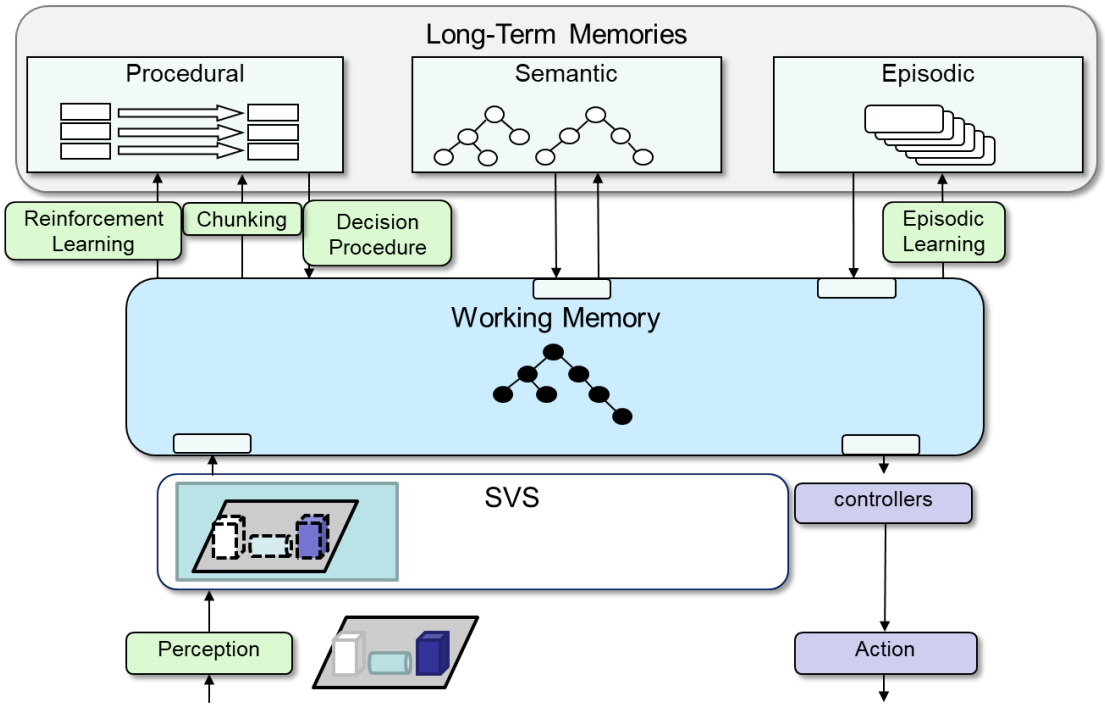


Figure 5: Soar architecture block diagram showing interaction (input and output) between the long term memories, working memory, and the Spatial Visual System (SVS) used for perception.

Soar contains a task-independent spatial visual system (SVS) that supports translations between the continuous representations required for perception and the symbolic, relational representations in Soar. The continuous environment state is represented in SVS as a scene graph composed of discrete objects and their continuous properties. Binary spatial predicates are computed when an agent issues a query for a specific predicate such as X-axis-aligned(A,B). The set of predicates is task independent and fixed, but predicate extraction is controlled using task-specific knowledge.

Figure 5 shows a block diagram of Soar picturing the organization and interactions between working memory, SVS, and the long-term memories: procedural, semantic, and episodic memory.

In Soar, working memory maintains symbolic relational representations of current and recent sensory data, current goals, and the agent's interpretation of the current situation

including mappings between objects in the scene and internal symbols and words. Working memory buffers provide interfaces to Soar's long-term memories, the perception and action systems, and the instructor interface.

Procedural memory contains Soar's knowledge of how to select and perform actions (called operators), encoded as if-then rules. The locus of decision making is not the selection of a rule. Instead, Soar fires all rules in parallel. The rules propose, evaluate, or apply operators, which are the locus of decision making. Only a single operator can be selected at a time, and once an operator is selected, rules sensitive to its selection and the current context perform its actions (both internal and external) by modifying working memory. Whenever procedural knowledge for selecting or applying an operator is incomplete or in conflict, an impasse occurs and a substate is created in which more reasoning can occur, including task decomposition, planning, and search methods. In Soar, complex behavior arises not from complex, preprogrammed plans or sequential procedural knowledge, but from the interplay of the agent's knowledge (or lack thereof) and the dynamics of the environment. In Rosie, procedural memory holds rules that implement the processing capabilities such as lexical processing, human-agent interaction, grounded comprehension, and acquisition of grounded representations of words. The agent also has rules that implement the primitive actions and their models. The acquired action-execution knowledge for verbs is stored in procedural memory.

Chunking is a learning mechanism that creates rules from the reasoning that occurred in a substate. When a result is created in a substate, a rule is compiled. The conditions of this rule are the working-memory elements that existed before the substate and were necessary for creating the result, and the actions are the result. The rule is added to procedural memory and is immediately available. Chunking is the mechanism that learns the action-execution knowledge for novel verbs.

Semantic memory stores context-independent declarative facts about the world. The agent can store working memory elements in semantic memory. Elements are retrieved by creating a cue in a working memory buffer and finding the best match (biased by recency

and frequency) in semantic memory. In Rosie, semantic memory stores linguistic mapping knowledge, such as the mapping between a word and a perceptual symbol (*red* color corresponds to symbol *r43*). Apart from linguistic mapping knowledge, semantic memory also stores compositions of spatial primitives and action-concept networks (discussed later). One advantage of semantic memory over procedural memory is that any aspect of a memory structure can be used for retrieval, whereas in procedural memory, there is an asymmetry between the conditions and actions. An agent can use *red* as a cue, or it could use *r43* as a cue, depending on what knowledge is available and what knowledge needs to be retrieved.

Chapter 4 Task Learning Process

This section provides a detailed description of the entire task learning process. In our approach, learning a task requires learning many ***task elements***: the goals, failure conditions, actions, and task-specific terms. Before showing how an entire task is learned, we describe how each task element is learned. This learning process is the same for all task elements irrespective of whether they are a goal, action, or task-specific term. The only difference, besides the way each type of element is applied, is that the teacher initiates the learning of a goal, action, or failure condition ("The name of the *goal* is three-in-a-row.") and the agent initiates the learning of a task-specific term ("Please describe the meaning of '*clear'* in this context."). Essentially, after the instructor begins teaching a new task ("The name of the game is Tic-Tac-Toe."), the agent prompts the teacher to provide definitions of the actions, goals, and failure conditions, interacting to learn the meaning of task-specific terms when necessary.

This explanation is followed by an illustrative example detailing the complete process of learning a specific puzzle, learning all the task elements, drawn from the example presented in the introduction. Finally, we describe how this knowledge is used to solve and complete the task through search. Throughout, we use examples from different games and puzzles that Rosie has learned to illustrate the learning process. We also explain how the problem characteristics (**C1-C4**) and desiderata (**D1-D4**) have motivated decisions about the agent design and learning process. In Chapter 7 we describe an extension to the task learning process described here that enables Rosie to handle sources of ambiguity during learning caused by the problem characteristics.

There are multiple types of task elements that must be learned to specify a goal-oriented task: actions, goals, failure conditions, and new task-specific terms. All of these involve detecting the states in which the task element is appropriate: when an action can

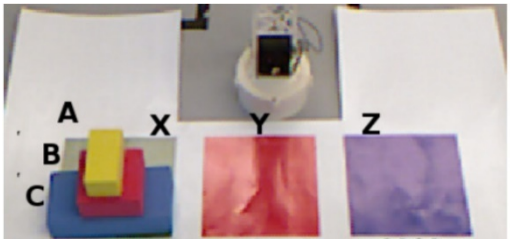
be legally applied, when a goal has been achieved, when a failure state has been reached, or when a task-specific term is applicable. Each task element is defined by a linguistic term ("stack," "three-in-a-row," "clear"), a conjunction of predicate tests, and the usage knowledge specific to that task element type. We use *predicate* to generally refer to all the components that test arguments in the logical definition of the task element. Each individual predicate can refer either to a learned task element, such as *clear(x)*, or knowledge that is part of the internal state derived from perception and primitive knowledge, such as *red(x)*.

For example, in teaching an action "stack" from the Blocks World, the following sentence could be used: "You can move a clear block onto a clear location." The linguistic term is "stack," the conjunction of predicate tests is *clear(X)* ∧ *block(X)* ∧ *clear(Y)* ∧ *location(Y)*, and the usage knowledge for an operator has to do with its actions: 'move X onto Y.'

In the sections below we describe in detail the four phases of the task element learning process: internal state creation (**L1**), language instruction (**L2**), recognition structure learning (**L3),** and operationalization (**L4**).

4.1 Internal Model Creation (L1)

In order to learn a new task element, Rosie must first be situated in a relevant environment, where the task can be performed, and it must build an internal model of the environment. The task element learning process requires that the described concept be present so that the correct grounding can be learned. Rosie starts by asking the instructor to configure the world into a state in which the conditions of the task element are satisfied. For an action, this is a state in which the action legally applies, while for the goal, it is a goal state, and so on.



Predicate instances in world state representation:
$small(A), medium(B), large(C)$
$on(A,B), on(B,C), on(C,X)$
$below(B,A), below(C,B), below(X,C)$
$location : \{X,Y,Z\}, block : \{A,B,C\}$

Figure 6: The predicate relationships extracted for the displayed external environment are to the right. The predicates are between movable blocks (A,B,C) and immovable locations (X,Y,Z).

The symbolic relational model of the external environment is maintained in Rosie's working memory. This model is automatically updated as new perceptual information comes in from the external world. This world state model consists of a set of objects and predicates over those objects. The left side of Figure 6 shows blocks and locations in an external domain (table-top robotic arm). Overlaying the picture are the internal identifying labels (A, B, C, X, Y, Z) created by the agent for the objects in the world. The right side of the figure shows the derived internal state.

Unary predicates defined over a single object, such as A or B, describe properties of that object, such as *small(A)* or *block(B)*. Binary predicates defined over two objects describe spatial relations between those objects, such as *on(A,B)* or *below(B,A)*. All relational predicates are represented symbolically, but are grounded in continuous representations maintained in the agent's spatial/visual short-term memory (when not in a simulated environment). The teacher can see the current scene (on the left), excluding the object labels, but does not have access to Rosie's internal state (on the right). The lack of common ground (**C1**) between the teacher and Rosie, and the many-to-many possible mappings between words and meanings (**C3**), makes learning more difficult.

When Rosie is embodied in a robotic platform, its perceptual systems extract the objects and primitive features and relations, including colors (*red*, *green*), sizes (*large*, *small*), spatial relations (*next to*, *below*), object types (*location*, *block*), and labels (*bank*, *destination*). A complete list of Rosie's primitive perceptual knowledge, including all concepts used for teaching the games evaluated in this thesis, is shown below in Table 1, organized by type.

Types	Primitives
Object types	location, block
Colors*	red, green, purple, yellow, orange, black, white, blue, brown, gray, pink
Labels	garbage, destination, card, bank, boat, pawn, king, knight, rook, queen, bishop, missionary, cannibal
Shapes*	cube, sphere, cylinder, rectangle, triangle, arch
Sizes*	tiny, small, medium, large
Spatial relations	on*, below*, near*, near*, left of*, right of*, under*, above*, behind*, in front of*, between (inclusive), between (exclusive), adjacent, diagonal, linear

Table 1: Primitive perceptual knowledge about the external environment initial encoded in the agent. Previous implementations have learned classifiers for the "*" concepts.

Many of these primitives were previously learned through simple KNN classifiers from a few training examples, but this has not been a recent focus of this work, which assumes initial knowledge of these primitive concepts (provided by either by stored training data for robotic platforms or directly from the world state for simulated environment).

4.2 Language Instruction (L2)

The instructor begins by teaching the name of the new task element, such as "The name of an action is stack." Again, Rosie *must* have an instance of the described task element visible (for the internal state creation **L1**) to learn the correct grounding of the task element for the current context, so Rosie requests that the instructor set up a state that contains at least one instance of the described element, be that an action, goal, or failure condition. For example, it will request that the instructor show it an example of a state in which the action *stack* is applicable. Once the request has been satisfied, Rosie asks for a description from the instructor that defines that concept in the current setting: "Teach me the action stack."

In order to understand sentences that are typed or uttered by the instructor, Rosie uses a single path, incremental, construction-based parser developed in Soar by John Laird. Off the shelf parsers are inadequate because they lack semantic precision and are difficult

to dynamically extend online with new words. Rosie can parse a restricted form of natural language that is sufficient for descriptions of the many games and puzzles studied in the thesis. Some example of sentences used to teach these games and puzzles are in Figure 7.

The parser can process multiple clauses, embedded clauses, and many natural forms of anaphoric references so that multiple references to the same object can be easily made in a sentence, thereby supporting moderately complex sentence structures. For example, Rosie can learn a goal from the sentence "The goal is that a small block is on a medium block and a large block is below the medium block." This goal state is visible in Figure 6.

The parser produces a relational representation that includes information derived from the linguistic structure of the sentence, such as specific relative clauses. Because of the many possible meanings (**C3**) that can be associated with the spelling of a new task element (such as "clear"), a unique symbol is created from the spelling of the word (such as *clear5*). A future usage of the term "clear" that has a different meaning will get its own unique symbol (*clear7*) when learned. The numbers are sequentially generated and not associated with any particular task or domain. Multiple definitions of the same term can be learned for a single task.

"An object that is not below a block is clear."

"If the volume of a block is less than the volume of an object then the object is larger than the block."

"You can move a clear block onto a clear object that is larger than the block." [Tower of Hanoi]

"If a block is adjacent to a clear location then you can move the block onto the location." [Eight Puzzle]

"The goal is that there are eight matched locations." [Eight Puzzle]

"If the value of a location is equal to the value of the block that is on the location then the location is matched. [Eight Puzzle]

"You can move a passenger of the boat onto the current bank." [Fox River Puzzle]

Figure 7**:** Example sentences used in teaching different tasks.

4.3 Recognition Structure Learning (L3)

Once a relational representation of the sentence describing the new task element has been created, Rosie builds a structure to recognize it. The task element recognition structure learning process is decomposed into two phases, structure construction and structure interpretation.

In the structure construction phase (described in detail in Section 4.3.1), the agent extracts a conjunction of predicates from the linguistic descriptions of the combinations of objects and relations. Rosie creates a declarative predicate structure that orders the conjunction of predicates to optimize later interpretation and support hierarchical composition. An example of the tree structure created is graphically depicted in Figure 8. From the goal "a small block is on a medium block and a large block is below the medium block," the agent constructs the structure with the unary predicates at the bottom (block) and the binary predicates at the top. The leaf nodes are predicate tests that are evaluated against the agent's perception of the world, and any objects that satisfy a predicate are passed up to the parent node during the grounding of concepts.

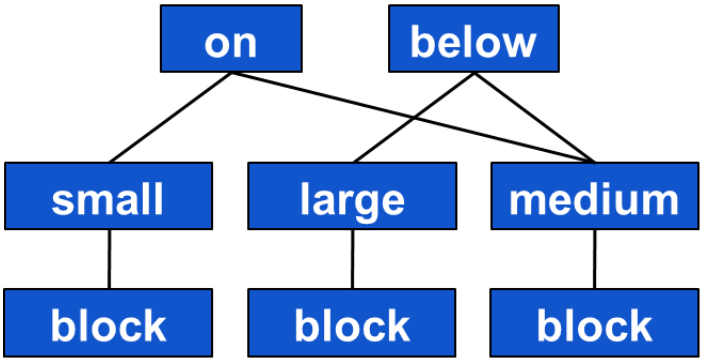


Figure 8: A graphical representations of the declarative structure Rosie creates from natural language that orders the predicate tests.

The conjunction of predicates, *p(x,...)*, is defined over a set of objects, *x*, which can be objects in the environment, as well as strings, numeric values, or sets of *x*. For example, from the goal description example, Rosie learns the conjunction: $goal(x_1,x_2,x_3) = small(x_1) \wedge medium(x_2) \wedge large(x_3) \wedge on(x_1,x_2) \wedge below(x_3,x_2)$. More generally, the conjunction can be represented as the intersection of *n* predicates $f_i()$ and *m* objects x_j as shown in Equation 1.

$$f(x_1, \ldots, x_m) = \bigcap_{i=1}^{n} f_i(x_j) \qquad 1 \leq j \leq m \tag{1}$$

Predicates, *p(x, ...)*, represent perceptual features: binary values over unary features (*red*, *large*...) and n-ary relationships (*on*, *behind,...*), but also set operations (*choose K*), functions (*count, sum*), comparators (*less than*), and actions (*move*). Predicates in the conjunction can be negated (¬*below*). A complete list of Rosie's primitive knowledge that is not directly available in the agent's internal model of the environment (**L1**), including all concepts used for teaching the games evaluated in this thesis, is shown below in Table 2. For our purposes, we label predicates as functions if they generate or return a different result from their input arguments, such as *choose K* which generates all subsets of size *K* from a larger set or *attribute of* which returns the value of the specified attribute (*color*, *size*, etc.) of an object. For the discrete primitive actions that the agent can perform, such as "write" and "move," Rosie knows the preconditions and effects, and can model each action during internal simulation.

The results of functions, where *y=f(x...),* are represented as predicate *p(y, x...)* with the result *y* as the first argument. For example, the representation created for "the number of blocks is three" is *count(3, blocks)*. Knowledge about primitives includes how they are referred to in relational representations created by the parser, so that the agent can convert utterances such as "the number of X is Y" to the predicate *count(Y, X)*.

Types	Primitives
Comparators	less than, more than, equal to, x less than, x more than, same
Functions	product, choose K, numeric between, attribute of, count, sum, difference
Actions	pick up, put down, move, write

Table 2: Primitive knowledge for actions, functions, and comparators encoded in the agent.

New predicates can be learned from the primitives as the agent learns task elements for task-specific terms, which enables the definition of hierarchical task elements through composition. Rosie converts the predicate conjunction into the tree structure (Figure 8) to help support composition of task elements and partial knowledge transfer (and reduce computation).

Next in the structure interpretation phase (described in detail in Section 4.3.2), the agent interprets that structure within the context of its representation of the external environment, also using its internal knowledge about the meaning of primitive predicates (such as *on*), learned predicates (such as *clear*), and internal functions (such as *count*). Rosie attempts to interpret the structure immediately after creating it, both to verify that the learned structure can be successfully grounded in the environment and to enable learning of procedural interpretation code that will be used for task performance. After verifying the learned structure, Rosie links it to the linguistic term for the task element and stores the structure into semantic memory (long-term declarative memory).

These two phases of recognition learning, structure construction and structure interpretation, which enable the agent to detect the task element in the external environment, are described in detail in the following sections.

4.3.1 Predicate Tree Construction

Evaluating a conjunction of predicates to determine possible matches can be computationally expensive. This is especially true for functions if they are tested on many

possible objects or sets. Rather than trying to jointly satisfy all predicate tests at once, Rosie learns a hierarchical tree structure to order the tests and minimize computation. This is also critical to supporting the compositionality (**C2**) of task elements and knowledge transfer during accumulative learning (**C4**). An example of the tree structure that Rosie constructs from a conjunction of predicates is graphically displayed in Figure 8. The predicates are ordered so they can be efficiently evaluated, from bottom to top in the following interpretation phase.

To construct this predicate structure, Rosie iterates through each predicate and adds it to the structure one by one, keeping track of the last predicate added to the structure that tested the same object or value. Rosie orders the adding of predicates based on the predicate arity and dependency information extracted during parsing, in order to minimize the number of objects tested by each predicate and thus reduce the time required to evaluate the structure. The pairwise effect of testing all combinations of objects causes evaluations of higher arity predicates to require more computation than lower arity predicates. For example, from the structure created for the goal in Figure 8, the predicate *below* will only be evaluated on object pair (A, B), the *small* and *medium* blocks from Figure 6. If the agent did not order predicate tree construction by arity and *below* was on the bottom of the structure it would have to test all objects pairwise: (A, B), (A, C), (A, D), (A, X), (A, Y), A, Z), (B, C), etc. This design decision also reduces the space of options to consider when contrasting multiple interpretations of task elements, which is an extension that we will describe later in Chapter 7, along with an explanation of how this design reduces the space of options.

Here, the objective is to reduce the computation time required to use these task elements after learning, thereby minimizing agent execution time (**D3**). First, unary predicates (*block*, *small,...*) are added, followed by any predicates created from dependent clauses (such as “the block that is on a location is...”), followed by binary predicates (*on*, *below*) and finally n-ary predicates (such as *between*).

When defining a task element for a new task-specific terms such as “clear,” rather than an action, goal, or failure condition, the sentence is always structured as an *if-then clause*. Logically the clauses are considered to be *if and only if* statements, but this does not prevent the learning of multiple definitions for “clear” due to the generation of separate versions *clear3* and *clear4*.

For example, “clear” could be defined with the sentence “if a location is not below an object then the location is clear.” The objects referenced in the *then-clause* that are tested by the predicate being taught are labeled as the inputs of the new predicate. The input object for the definition of *clear* is the location. Figure 9 shows a graphical representation of the learned structure for an action from the Eight Puzzle that contains two task-specific terms, “clear” and “adjacent to.” For unary predicates, such as *clear*, there is only one input. For binary predicates, such as *adjacent to*, the inputs are labeled input1 and input2 for predicate arguments x_1 and x_2 respectively. The subgraphs showing the recognition structures learned for the new predicates *clear* and *adjacent to* are also displayed, with the accompanying text for teaching *adjacent to*. In this figure, each predicate is labeled with its type: unary feature, relation, function, input, or learned predicate. In addition, the graph shows the argument attachment of the verb, which is exclusive to action task elements.

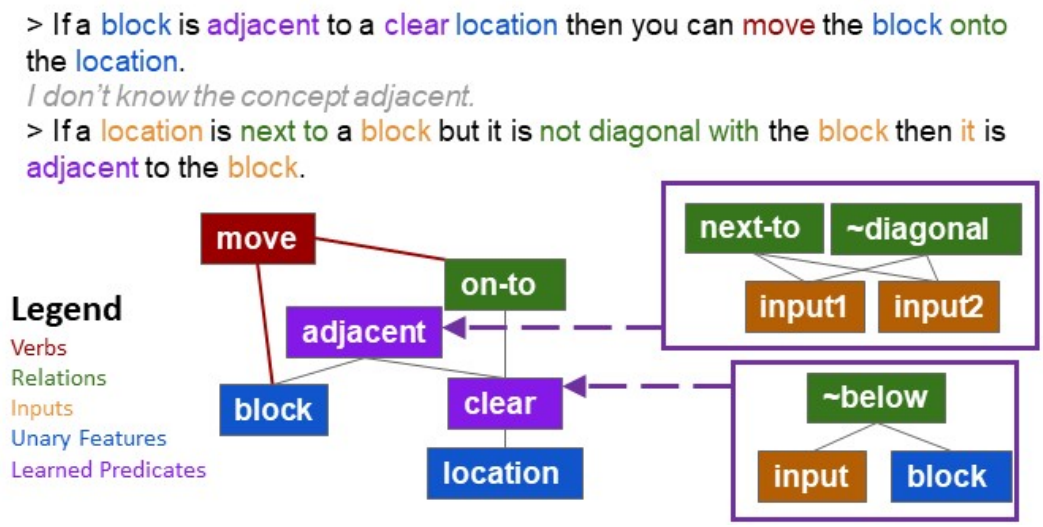


Figure 9: A simplified graphical view of the representations Rosie learns for an action in Eight puzzle. Included are graphs for the hierarchical concepts used (clear) and taught (adjacent to). The words and boxes are highlighted according to their part of speech or type.

When constructing the declarative predicate structure for "adjacent," the objects in the *then-clause* that are referenced by adjacent are labeled as the inputs. The references to the input objects from the *if-clause* establish the necessary conditions for the new predicate defined over the inputs.

Only the predicates contained exclusively in the *if-clause* are added to the predicate structure, so the structure learned for *clear(x)* only contains *~below(x, object)*. This last decision was made to allow Rosie to generalize and reuse the definition during accumulative learning (**C4**), but in some cases it causes Rosie to overgeneralize and transfer knowledge it should not. However, in the above example for "clear," where Rosie assumes that the object being a location is not relevant because the predicate exists in both *if* and *then* clauses, this is a useful generalization; the learned representation *~below(x, object)* will also work for non location objects, such as a block that is *clear*.

If instead Rosie learned *~below(x, object)* ^ *location(x)* it would need to learn another definition of *clear* for blocks. However, in a different example "if a banana is yellow then it is ripe," the agent will overgeneralize and learn that *ripe(x)* means *yellow(x)*, even though obviously not all yellow objects are ripe; it would be better to have learned

yellow(x) ^ banana(x). This is a greedy design choice (which could be easily reversed) done to maximize potential transfer and minimize teach-agent communication (**D2**). For the tasks and environments we have explored teaching, this has not caused many problems, but it could for other tasks and environments, where it may be worthwhile to prioritize correctness of learning over quickness of teaching. Future work will explore mechanisms for verifying the correctness of learning and correcting mistakes created by overgeneralization or incorrect knowledge transfer.

The existence of many possible meanings (**C3**) and the lack of knowledge of the instructors meaning (**C1**) makes correct generalization difficult. A more complex, complete example is discussed in Section 4.4.3.

Strictly non-hierarchical, or flat, representations that attempt to encode the same knowledge are problematic, not only because they increase the required computation (**D3**) and communication (**D3**), but also because they make it difficult to parse the sentence, make appropriate anaphoric references, and resolve ambiguity. For example, without using clear or adjacent to define the action in Figure 9, the sentence would become: "If a block is next to a location that is not below a block but it is not diagonal with the block then you can move the block onto the location." Determining the correct attachment of the clauses and the object references is difficult: which of the blocks referred to in the *if-clause* are the ones references in the *then-clause*? Hierarchical composition (**C3**) helps reduce the number of objects, predicates, and the overall complexity of each task element.

After Rosie has finished creating the recognition structure, it stores the declarative structure in long-term semantic memory, so that it be linked with the linguistic term for the task-element, such as "clear" or "three-in-a-row," and the name of the task being taught, such as "Tic-Tac-Toe." This structure can be retrieved from semantic memory at a later time, such as when it is used in another task, using the linguistic term or the task name as a cue.

4.3.2 Structure Interpretation (Grounding)

Once Rosie has created a recognition structure, it tries to use it to recognize the task element in its internal model of the environment (**L1**). As previously mentioned, the task element must be present in environment so that Rosie can learn the correct task-specific grounding of each term. This is necessary to verify the correctness of the recognition structure for the current context, learn procedural knowledge for directly evaluating the structure (though chunking), and to determine if the agent requires additional knowledge: it may need to learn another task element for a task-specific term used in the recognition structure. Rosie interprets the declarative predicate structure using internal knowledge about functions, primitive predicates, and learned predicates, and grounds it in the agent's perception of its external environment. Essentially Rosie is figuring out how to interpret, or ground, the declarative structure in a situated context given its current state of knowledge so that is can recognize the task element that structure defines.

Figure 6 displayed an example external environment that can be used to interpret the Blocks World goal described above. The interpretive process finds all possible matchings of the environmental state to the task element. This process is decomposed into two parts: first, the constituent predicates of the structure are individually evaluated by matching them against internal knowledge to produce sets of candidates from the environment that individually satisfy each predicate; second, the candidates generated by the predicates are joined or intersected, filtering out objects that do not simultaneously satisfy all the constraints.

Individual predicate matching:

Different types of predicates (such as spatial relations, unary features, and functions) are evaluated using different methods. Each method evaluates the predicate within the context of the world state by retrieving the predicate's associated semantics based on the linguistic term. For example, for the function referenced by "number of," Rosie uses an internal *count*

operation that calculates the size of the set. The predicate matching process completes when results have been calculated for every predicate in the learned representation.

Figure 10 shows the results of predicate matching for the example Blocks World goal and world state shown in Figure 6. The results of a predicate are depicted at the top of each box representing the predicate. Rosie first evaluates the unary predicate *block* over all the objects in the world state (A, B, C, X, Y, Z), successfully matching only against A, B, and C. Due to the ordering constraints that Rosie imposes through the learned structure, when Rosie evaluates the predicates for *small*, *medium*, and *large*, they are tested against only the blocks A, B, and C. The predicates for *on* and *below* are evaluated last and are tested against only the blocks A, B and C, B respectively. This helps Rosie to achieve faster execution (**D3**), especially when there are many predicates and functions in the description.

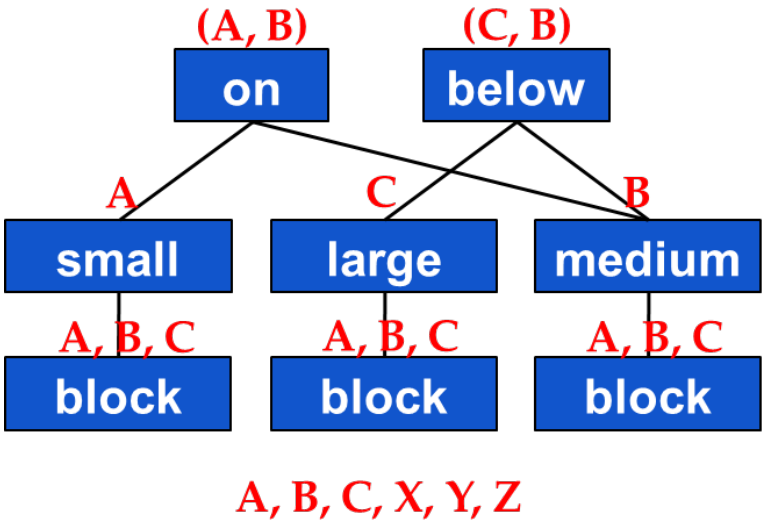


Figure 10: A graphical representations of the declarative structure from **Figure 8**, now marked with the results of predicate matching indicated by object identifier numbers in red.

Joining/satisfying intersection:

After each constituent predicate has been evaluated, Rosie attempts to jointly satisfy the arguments of the declarative predicate structure by evaluating the intersection of the results from the predicate matching. The results are the objects and values in the external world that satisfy all constraints. For the Blocks World goal in Figure 6, the join is trivial because

there is only one block of each kind, small, medium, and large, and blocks A, B, and C satisfy all the predicates jointly.

4.3.3 Learning procedural rules through chunking

The processing described above is implemented in Soar as a hierarchy of problem spaces with associated operators. Thus, the interpretation phase is dynamically decomposed into operators that perform the component processing steps of predicate matching, term linking, and joining. Each of these is implemented in its own substate through operators that manipulate the appropriate data structures.

A critical aspect of this process is that as a side effect of the processing, Soar's chunking mechanism creates rules that capture the input-output mappings of the processing. Chunking dynamically compiles the processing in a substate into procedural rules, so that when the situation that led to the substate arises in the future, the processing in the substate is bypassed by those rules. During the interpretation process, rules are learned for each of these component processing steps so that in the future, the deliberate, sequential processing is replaced by procedural rules (native Soar production code **D3**). These rules can directly recognize and apply the task elements, but much faster (approximately 80x) than the initial interpretation process. During the interpretation process, three different kinds of rules are learned.

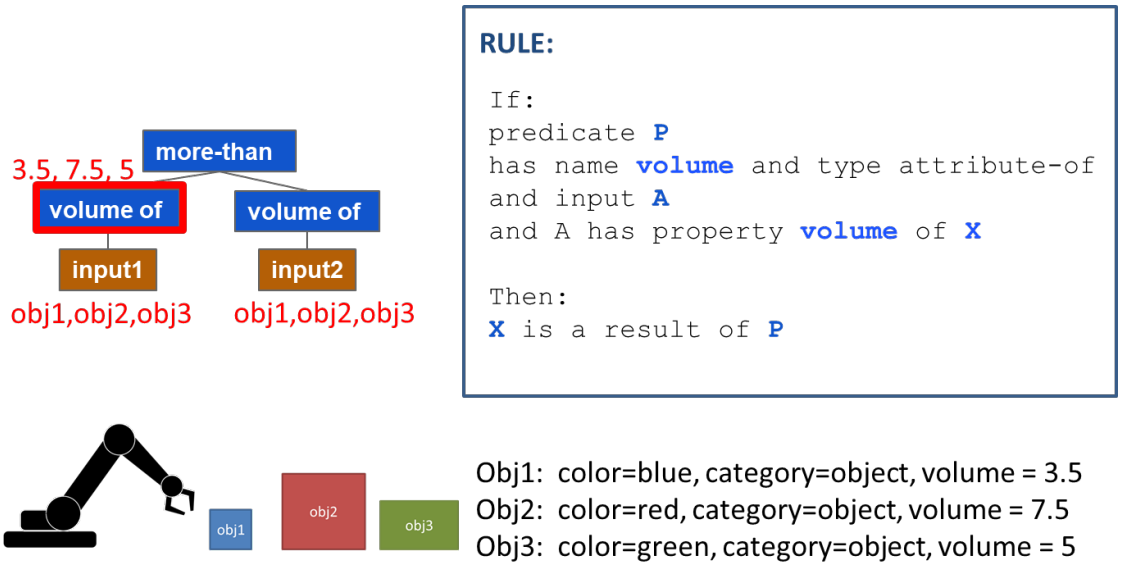


Figure 11: A representation of the procedural knowledge, the soar rule, learned through chunking for resolving "volume of."

The first type of rule Rosie learns though chunking is from predicate matching (Section 4.3.2). Rosie learns rules to evaluate primitive unary features, spatial relations, and internal functions. The processing of some predicates lead to multiple rules being learned, such as those dealing with functions and sets of objects. For example, consider that Rosie is learning the task-specific term "larger than," which the teacher has defined with "If the volume of an object is more than the volume of a block then the object is larger than the block." Figure 11 shows the recognition structure learned for "larger than" during the process of predicate matching. The figure also shows a representation of the agent's internal model of the environment (at the bottom), and a representation of the Soar rule that is learned for evaluating *volume of* (on the right).

Once a rule is learned, it replaces the processing via the hierarchy of operators used to interpret the predicate. The interpretation process is impasse driven, only the absence of results causes the predicate interpretation process to begin. In this example, this would involve retrieving knowledge about the predicates from semantic memory, and then using the context of its internal model of the current state of the world (shown on the bottom of

the figure) to determine that *volume* is an object property that is a unary primitive numeric attribute of objects that are currently visible in the world. Recent improvements to chunking have improved its ability to generalize, so that the elements highlighted in blue in the rule shown in Figure 11 are variablized: this rule will also be able to evaluate a similar predicate, such as for *color of*, without learning a new rule, thus transferring knowledge. This type of knowledge transfer does not reduce the amount of instruction required from the teacher, but does reduce the time it takes for structure interpretation during the learning of future task elements.

Figure 12 shows another example of a rule learned for the evaluation of the *more than* predicate, which can be evaluated now that volume of has generated results (for both input arguments). In this example, Rosie learns a rule for evaluating binary predicates that compare two integer objects (A and B) and returns a pair (A, B) as a positive result if the first number (A) is larger than the second number (B). This rule is not very general: the agent would need to learn another rule for less than or equal to. Not all learned rules have the level generality shown in rule from Figure 11.

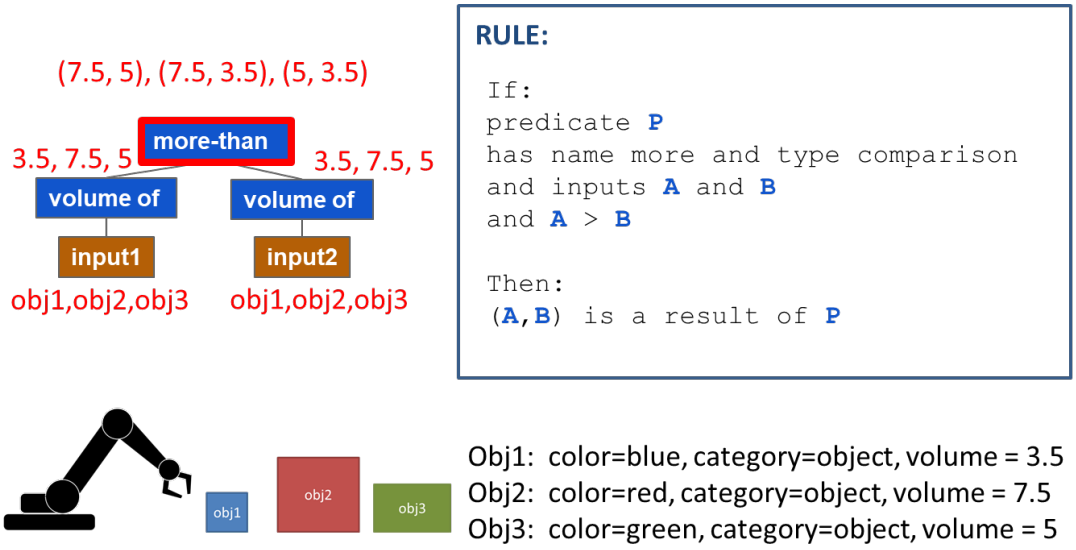


Figure 12: A representation of the procedural knowledge, the Soar rule, learned through chunking for resolving "more than."

The second type of rule Rosie learns is during the process of linking the linguistic term to the learned structure. These rules test the name of the task element used, whether it is a goal, failure, action, or new predicate for a task-specific term, and then add the corresponding recognition structures into working memory (WM). Figure 13 show a representation of the procedural knowledge learn for *larger-than1*. The existence of the predicate *larger-than1* in part of another task element, such an action, will cause this rule to match and fire, so that larger-than can be evaluated over its input arguments. These rules are not general, they are specific to a single task element.

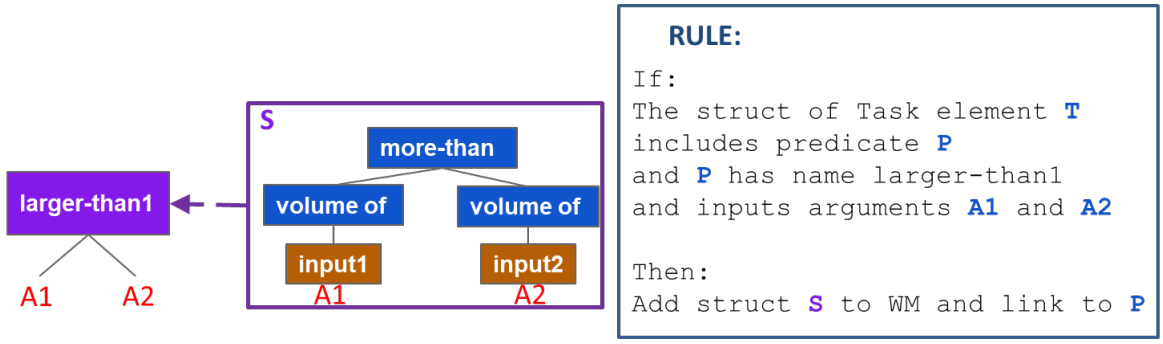


Figure 13: A representation of the procedural knowledge learned through chunking for creating the recognition structure of a task-specific term, "larger than," in working memory.

These rules prevent Rosie from retrieving the recognition structures from semantic memory, because the existence of the task element's name in working memory causes the rules to match and fire, adding the structures directly to working memory. Figure 14 shows a representation of the procedural rule learned for a goal from the blocks world example.

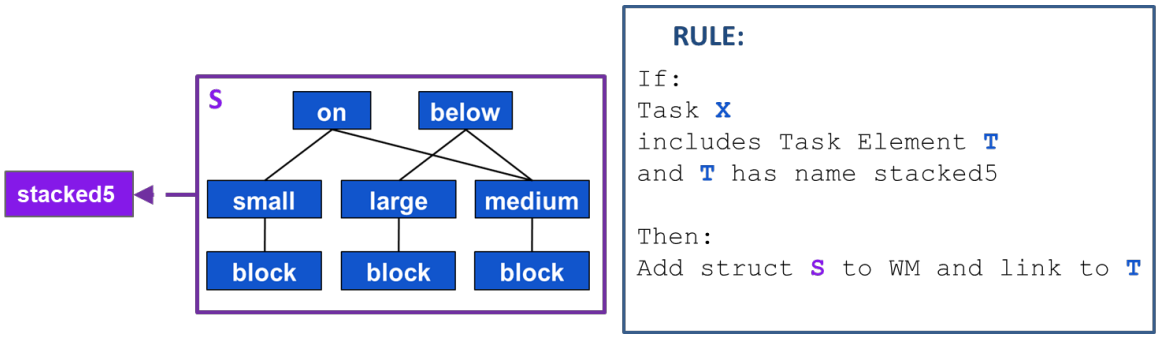


Figure 14: A representation of the procedural knowledge learned through chunking for creating the recognition structure of a task element, the goal example, in working memory.

The final type of procedural knowledge that Rosie learns is learned during the linking of the name of the task to the actions, goals, and failures. Figure 15 shows a representation of the procedural knowledge learned for the eight puzzle, which links the name of the game to its action, *slide2*, and goal, *eight-matched5*.

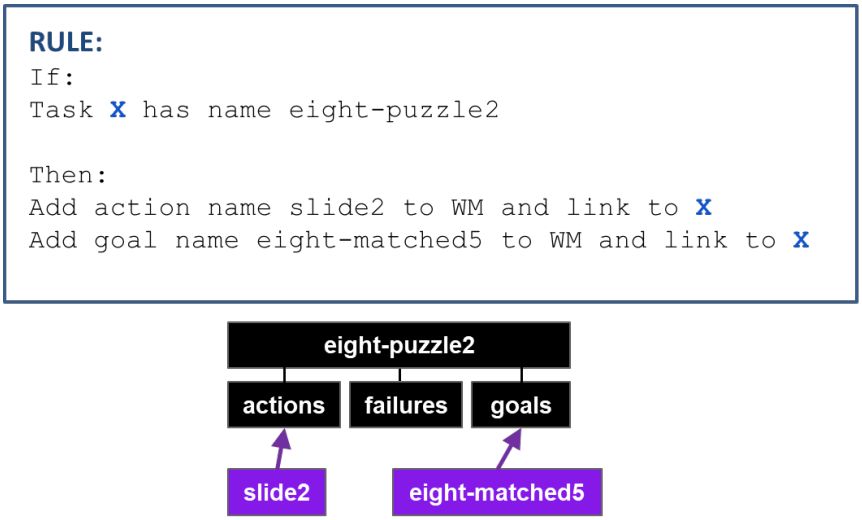


Figure 15: A representation of the procedural knowledge learned through chunking for link the names of the goal, action, and failure conditions for a task, in this case the eight puzzle.

In the future, when the task is attempted again, these rules cause Rosie to add to working memory the names of all the goals, actions, and failure conditions learned for that task, without needing to retrieve them from semantic memory. This in turn causes the second type of rule to match and fire, populating working memory with the recognition structures for those actions, goals, and failure conditions. If there are any task-specific terms, their existence in the recognition structures of the other task elements will cause additional rules to match and fire, adding into working memory the recognition structures for those task-specific terms. These structures can be directly evaluated against the agent's internal model using the first type of rules that it learned.

Although the declarative structures are still maintained in long-term semantic memory, these rules, stored in procedural memory, are all that Rosie needs to be able to

attempt or solve a task. No semantic memory retrieval or interpretation is necessary after learning. The agent runs as normal but with the addition of the procedural rules that avoid the impasses and enable the agent to recognize every task element when it is prompted to attempt the task.

4.3.4 Recursive Learning Algorithm

As mentioned previously, the learning can recursively loop through **L1**-**L4** when Rosie requests a definition for a task-specific term. However, in many cases there are multiple options to pick from: there is more than one predicate that cannot be satisfied or grounded in the current context given the agent's current state of knowledge. In the past, Rosie selected the first term mentioned, but with an increasing numbers of terms it could learn, and for cases where many task-specific terms needed to be defined for a single task-element, we have had to develop a more sophisticated recursive learning algorithm to guide the learning process.

The process of learning to ground all parts of the task element is described by the Recursive Grounding Function (RGF) depicted in Algorithm 1. The input is the generated recognition structure, here represented as a conjunction of predicates as shown in Equation 1. The terminal condition is that the input function *f(x)* can be satisfied, meaning that an instance of the task element is detected in the environment by applying the recognition structure. If no definition is known for *f(x)*, the agent prompts the teacher for a definition.

Algorithm 1 Recursive Grounding Function $RGF[f(x)]$

```
if f(x) can be satisfied then
    terminal condition
end if
if f(x) is undefined then
    Ask teacher for definition of f(x)
end if
for each f_i in f(x) do
    if f_i(x_j...) is not satisfiable then
        Propose learning new grounding RGF[f_i(x_j)]
    end if
end for
Also propose learning new grounding RGF[f(x)]
Use heuristics to select RGF[] recursive call
```

Otherwise, for each of the unsatisfied predicates $f_i()$ used to define *f(x)*, Rosie proposes a recursive function call *RGF[f_i()]*. The agent also proposes *RGF[f()]* to consider learning a new definition for *f(x)* even though it already has one (many-to-many mappings). Because there may be many unsatisfied predicates, heuristics are used to select which recursive function call to make.

These heuristics leverage the hierarchical tree structure of the task elements to bias the selection of the next unsatisfied predicate to attempt to learn a new meaning for. The heuristics used for this process are listed below in order, with heuristics listed earlier taking precedence over those that follow. Rosie prefers learning predicate *p* over predicate *q*, where *f(x)* is undefined for both, if the arguments of *p*, its child predicates in the recognition structure, have both returned results. This prevents trying to learn a new definition for a predicate that has no chance of being successfully learned: it has no values or objects as input arguments to test. Rosie prefers learning predicate *p* over *q* if *p* is a descendant of *q* (*block* is a descendant of *below* in Figure 14, *small* is not). Rosie prefers learning predicates that are lower in the hierarchy of task elements (*RGF[f_i()]* > *RGF[f()]*). This learning function terminates when it has learned to recognize and apply the task element *f(x)* and all supporting task-elements $f_i(x)$ used to describe it.

4.4 Operationalization of Task Elements (L4)

Once the declarative predicate structure has been successfully interpreted and grounded in the agent's perception of its environment, and procedural rules for this process have been learned, the task element has been recognized and can now be operationalized, or applied, in the task environment. The type of task element being taught determines the details of the operationalization learning phase. To operationalize an *action* task element is to identify available legal actions. Those actions are then proposed to search or to act in the world.

To operationalize a *goal* task element is to indicate that the goal state has been detected: the agent has won the game. For example, if Rosie recognizes the goal *three-in-a-row* in the internal state created by simulating an action in Tic-Tac-Toe, it now knows that that action will win the game. To operationalize a *failure condition* task element is to indicate that a terminal state has been detected: the agent has lost the game or found a bad solution path. For example, if Rosie recognizes the failure condition of Tower of Hanoi puzzle, *larger-on-smaller*, in an internal state while searching for the solution, Rosie stops searching down that path. To operationalize a *task-specific term* task element is to indicate a successful match of a predicate that is part of the definition of another task element. For example, if Rosie recognizes the unary term *captured* in a game state of Tic-Tac-Toe, the knowledge of which squares are *captured* allows Rosie to determine if the goal ("three captured locations in a line") is present. The procedural rules Rosie has learned (Section 4.3.3), such as the example in Figure 13, are sufficient for operationalizing a task-specific element because it only involves recognition, so no additional code is required to operationalize it, unlike the other task elements.

4.5 Task Solving

The instructor signals that they are done teaching the actions, goals, and failure condition of the task by telling the agent they are "done." At this point Rosie has learned declarative

knowledge (the recognition structures) and procedural knowledge (the chunked rules) for recognizing and operationalizing the goals, actions, failure conditions, and supporting task-specific terms, by recursively applying the learning process described (**L1-L4**) for each unknown task element.

After completing learning the task, Rosie asks the instructor if it should attempt to solve the task: "I have learned the task. Should I try to solve the puzzle?" The agent solves the task through straightforward search, by using the task element knowledge to propose actions, apply them through internal simulation, and evaluating the resulting states and whether they contain a goal or failure condition. If the task is a multiplayer game, not a puzzle, instead the agent asks "Shall we play a game?" and only searches forward one step.

For single-player puzzles, it uses iterative deepening, implemented as recursive substates in Soar (Laird, 2012), to search for the goal. For each state in the search, Rosie determines whether a goal or failure state is present, and if not, iteratively extends the search by generating the legal actions for the new state until the current depth limit is reached. If a failure state is encountered, that search path is abandoned. If a goal state is encountered, the search terminates, and the appropriate action is selected, and the associated verb command is executed. If the task is a single-player puzzle, Rosie successively selects and executes the actions it discovered that were on the path to the goal. These actions are retrieved from an "actions to perform" stack that Rosie maintains; each action that is in the path to the goal is pushed onto the stack.

The internal search is possible because even though the full action model of a task action is not specified by the instruction, the agent has the action knowledge of the associated primitive verbs. A full action effect model specifies not only the direct result of the action, but all the related relationships that change as a result. For example, an action in the Eight Puzzle, slide, does not verbally encode the fact that a new location is made empty. That is, the preconditions of an action are encoded, but not all of its effects. However, because Rosie has primitive knowledge of the verbs used to define the task actions, such as *pick up* or *move*, it can simulate these actions using built in action models.

When embodied in real non-simulated environments, Rosie also has the capability to simulate actions in its spatial visual system to determine not only their primary effects (such as the movement of a tile to a new location), but also secondary effects (such as changes in spatial relations with other objects). This capability is unique compared to other game player systems, such as those using the Game Description Language (Genesereth & Love, 2005), which must explicitly represent all primary and secondary action effects.

When playing a two-player game, Rosie only searches forward one step. The agent currently lacks knowledge about the opponent, or its actions. If the agent is one move away from a goal, the one-step search is sufficient; in other cases, the agent uses a general heuristic that approximates the state's distance to the goal. It calculates this approximation by counting the number of partially matched goal parameters, which produces better than random behavior for many games. These limitations of the solution strategies of the agent (iterative deepening and one step look-ahead) are not a result of fundamental limitations of the system (and the representation it learns) rather they have not been a focus of our research; we have not put effort into more sophisticated search techniques.

Chapter 5 Task Learning Examples

This chapter provides a verbose example of task learning, showing everything that is learned for an entire task, and examples of different task-specific terms that Rosie has been able to learn for various tasks. The example task is drawn from the example river crossing puzzle from Chapter 1.

5.1 Learning the Jealous Managers Puzzle

This section illustrates the previously described learning process through an example of learning the task elements of the puzzle introduced in the first chapter. In this instance there are only two pairs of agent-managers as shown in Figure 16, in order to simplify analysis of the learned representations and how they map to this environment.

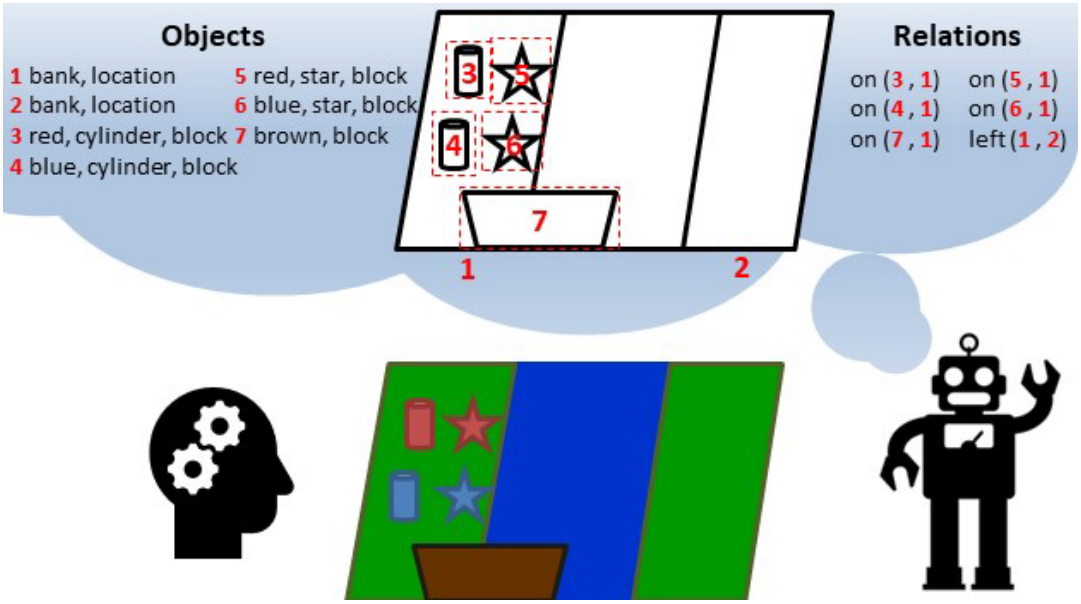


Figure 16: A depiction of an instructor attempting to teach a situated version of a river crossing problem and the agent's internal model derived from its perception and knowledge.

This puzzle requires that you ferry managers and actors across a river, with at most two in the boat at once, while avoiding the actors from being "poached:" the actor cannot be on a bank with another manager when their own manager isn't there. Given such a sparse representation of the environment, only the colors, shapes, and simple relations of blocks are known, how could a robot learn the complete puzzle?

One possible way that the task could be taught is shown in the dialog below showing the interactive dialog between the teacher and Rosie (highlighted in *red*). Each sentence is ordered with red numbers (referenced by superscripts) and each task elements that the agent learns are enumerated in bold (referenced by subscripts). The details of each task element description will be explained throughout the learning process below.

1. The name of the puzzle is jealous-managers.
2. *Please setup the puzzle.*
3. Ok.
4. *Please describe the actions, goals, and failure conditions.*
5. The name of an action is **ferry-one**$_1$.
6. *What are the conditions of the action?*

The instructor initiates the teaching a new task by give it a name: "The name of the puzzle is jealous-managers."[1] If the agent knows this task already (by name), it will ask if it should try to solve the puzzle or play the game, otherwise it will learn a new task. Rosie starts by asking the teacher to setup the puzzle in the current environment, as shown in Figure 16, and then asks the teacher to initiate the teaching of actions, goals, and failure conditions. The agent initiates the learning of task-specific terms. The instructor begins by teaching the name of an action to be learned: "The name of an action is ferry-one." [5] When Rosie decides it needs a new definition for a task element, in this case it doesn't know any task element named **ferry-one**$_1$, using the process described in section 4.3.4, it initiates an interaction with the teacher to learn that concept: "What are the conditions of the action?"[6] This learning is guided by the process described in section 4.3.4, but will not be described in detail here.

When Rosie prompts the teacher for a definition of the action the teacher responds "You can move the boat and a **person**$_2$ on the **current**$_3$ bank onto the **opposite**$_4$ bank."[7] However, Rosie does not know the meanings of the task-specific terms "person," "current," and "opposite." These terms do not show up in the task environment (Figure 16) and Rosie does not know how to ground them.

> **7.** You can move the boat and a **person**$_2$ on the **current**$_3$ bank onto the **opposite**$_4$ bank.
> ***8.*** *Please describe the meaning of current in this context.*
> **9.** If the bank is below a boat then it is the **current**$_4$ bank.
> ***10.*** *Please describe the meaning of below in this context.*
> **11.** If a block is on an object then the object is **below**$_5$ the block.
> ***12.*** *Ok, Ive learned the meaning of below for this context.*
> ***13.*** *Ok, Ive learned the meaning of current for this context.*

The first task specific term Rosie attempts to learn a new meaning for is **current**$_4$. When prompted by Rosie, the teacher describes "current" as "If a bank is below a boat then it is the current bank."[9] However, looking at Figure 16 again, the agent only knows the primitive spatial relationship *on*. The inability to detect any results for "below" prompts Rosie to ask for a new definition for **below**$_5$, which the teacher defines simply as the inverse of on: "If a block is on an object then the object is below the block."[11] With this knowledge, Rosie can now satisfy the recognition structure for **below**$_5$ which enables it to satisfy the recognition structure learned for **current**$_4$, and finally it can evaluate the action again, which prompted this recursive learning of the unknown terms. However, Rosie is still unable to ground two of the terms.

> ***14.*** *Please describe the meaning of opposite in this context.*
> **15.** If the bank is not current then it is the **opposite**$_8$ bank.
> ***16.*** *Ok, Ive learned the meaning of opposite for this context.*
> ***17.*** *Please describe the meaning of person in this context.*
> **18.** If an object is a block and it is not a boat then it is a **person**$_3$.
> ***19.*** *Ok, Ive lear ned the meaning of person for this context.*
> ***20.*** *Ive learned the action.*

When prompted by Rosie for a definition of **opposite$_8$** (line **14**), rather than defining it from primitives, the teacher uses a term that we have already defined, and defines a concept using its negative: "If a bank is not current then it is the opposite bank."[15] Once Rosie has successfully learned how to recognize and apply **opposite$_8$** in the current environment, Rosie prompts for a definition another ungroundable concept **person$_3$**. In this instance of the puzzle, from looking at Figure 16, we can see that all the people are represented by blocks, but the boat is also a block. In this example, the instructor decides to teach the meaning of **person$_3$** as "If an object is a block and it is not a boat then it is a **person3**." [18] With these new definitions, Rosie can now recognize the action **ferry-one$_1$** and operationalize it: move a person on the current bank (with the boat) and the boat on to the opposite (not current) bank.

21. The name of an action is **ferry-two$_6$**.
22. *What are the conditions of the action?*
23. You can move a **person$_3$** that is on the current bank and another **person$_3$** that is on the **current$_4$** bank and the boat onto the **opposite$_8$** bank.
24. *Ive learned the action.*

Now that Rosie knows how to ferry a single person with the boat, Rosie is taught how to ferry two persons with the boat, **ferry-two$_6$** (line **21**). When Rosie prompts the teacher for a definition "What are the conditions of the action?"[22] the teacher responds with "You can move a **person3** that is on the **current4** bank and another **person3** that is on the **current4** bank and the boat onto the **opposite8** bank."[23] Rosie has just learned meanings for these terms, and because they lead to matches in the environment, though intra task transfer Rosie does not require or ask for additional definition of terms and is immediately able to detect the new action.

Rosie has now learned all the necessary action knowledge for solving this puzzle and must now learn the failure condition and goal. This learning example is completed on the online archive at umich.edu/~jrkirk/ijcai2019.html, where we also include figures and analysis of each recognition structure that is learned for each of the task elements, which was too verbose to include here.

5.2 Examples of Learned Task-Specific Terms

A list of some of the task elements that Rosie has learned, specifically task-specific terms, is shown below in Table 3, organized by the Part of Speech (POS) of the word. This is not a complete list of everything Rosie has learned, but shows many of the different types of terms it can learn in different situations.

Part of Speech	**Task-specific Terms Learned**
Nouns	frog, toad, box, boat, actor, manager, missionary, cannibal, grapefruit
Nouns used as Functions (with of)	passenger of, husband of, wife of, occupant of, manager of, position of, neighbor of, score of
Prepositions	adjacent to, below, on, under, in a line, in a group, left of, right of, …
Adjectives	covered, free, clear, current, your, small, large, huge, wild, frog-covered, toad-covered, matched, matching, occupied, colorless, shapeless, surrounded, center, fork, raw, cooked, well-done
Comparative Adjectives	smaller than, colder than, heavier than, higher than, lower than, weaker than, stronger than, warmer than
Superlative Adjectives	coldest, hottest, largest, smallest, highest, heaviest, lowest, top, bottom
Stative Verbs	attackable by, attacking, capturable by, occupied by, conquerable by, matched by, captured by, captured

Table 3: A list of some examples of task-specific terms that Rosie has learned organized by the part of speech of the term.

The ability to compose hierarchies of task elements enables Rosie to learn many different types of task elements (**C2**). Essentially Rosie can learn new classes of knowledge based on the types of available primitives. In various domains these primitives have included colors (*red*, *green*), sizes (*large*, *small*), relations (*next-to*, *smaller-than*), labels (*location*, *destination*), and functions (*count*, *attribute-of*, *comparison*). Through hierarchical composition of these primitives, Rosie can learn new relations (*adjacent to*), labels (*captured*, *current*, *opposite*), and functions (*husband-of*, *passenger-of*). Rosie can also

learn synonyms (*huge*), antonyms (*covered* and *clear*), and homonyms (*matched*). The two different definitions learned for "matched," as well as some of the other example learned task elements, are described below.

The learned task elements can be task and domain dependent and be redefined based on the available knowledge and environment representations. For example, in an instance of the Jealous Husbands river crossing puzzle, the attribute used to designate couples is their "last-name," which is unique to each pair of men and women. In this domain, when describing the failure condition: "If a woman is on a bank and the husband of the woman is not on the bank and another man is on the bank then you lose," the unknown term "husband of" can be described by: "If the last-name of a woman is the last-name of a man then the man is the husband of the woman."

In the following subsections we show examples of sentences used to teach different task-specific terms organized by the Part of Speech of the term.

5.2.1 Nouns

Examples of sentences used to teach task elements for task-specific terms that are nouns are displayed in Figure 17.

"If an object is a block and the object is red then it is a **frog**."

"If an object is a blue block then it is a **toad**."

"If an object is a medium brown rectangle then the object is a **box**."

"If an object is brown and it is on a bank then it is a **boat**."

"If the shape of an object is a star then it is an **actor**."

"If the shape of an object is a cylinder then it is a **manager**."

"If an object is a red block and the object is on a bank then it is a **missionary**."

"If an object is a blue block and the object is on a bank then it is a **cannibal**."

"If an object is a large yellow sphere and the object is in the kitchen then the object is a **grapefruit**."

Figure 17: Example sentences used in teaching task elements for nouns.

Rosie can be taught task-specific terms for nouns by leveraging the current attributes it can detect for an object. This enables the instructor to use appropriate terms, such as "missionary" or "actor," for teaching the actions and goals to Rosie, even when the agent's sensing of the environment is limited to very simple features, such as colors and shapes. These definitions are often not very transferrable outside of the task (inter task transfer) because they are context specific, but they do support intra-task transfer, such as between the task actions and goals.

5.2.2 Nouns that act as functions

Examples of sentences used to teach task elements for task-specific terms that are nouns that act as functions, by combination with "of," are displayed in Figure 18.

"If a block is on a boat then the block is a **passenger of** the boat."

"If the last-name of a woman is the last-name of a man then the man is the **husband of** the woman."

"If the last-name of a man is the last-name of a woman then the woman is the **wife of** the man."

"If a block is on a location then the block is an **occupant of** the location."

"If the color of a manager is the color of an actor then it is the **manager of** the actor."

"If a location is below a block then the location is the **position of** the block."

"If a location is adjacent to another location then the former location is a **neighbor of** the later location."

Figure 18: Example sentences used in teaching task elements for nouns that act as functions.

Rosie can be taught task-specific terms for functional nouns, or nouns that act as functions, such as "passenger of" or "position of."

5.2.3 Prepositions

Examples of sentences used to teach task elements for task-specific terms that are prepositions are displayed in Figure 19.

> "If a location is next to an object but it is not diagonal with the object then it is **adjacent to** the object."
>
> "If a block is on an object then the object is **below** the block."
>
> "If a block is blue and the column of the block is the column of a location then the block is **below** the location."
>
> "If a block is below an object then the object is **on** the block."
>
> "If a location is above an object then the object is **under** the location."
>
> "If the blocks have the same row then they are **in a line**."

Figure 19: Example sentences used in teaching task elements for prepositions.

Rosie can also be taught task-specific terms for prepositions, such as "adjacent to" or "in a line," using primitive spatial relations, such as "diagonal" or "next to," or other learned prepositions.

5.2.4 Adjectives

Examples of sentences used to teach task elements for task-specific terms that are adjectives are displayed in Figure 20.

> "If a location is below an object then it is **covered**."
>
> "If a block is not on a location then it is **free**."
>
> "If a location is not below an object then it is **clear**."
>
> "If a bank is below the boat then it is the **current** bank."
>
> "If a block is red then it is **your** block."
>
> "If the volume of a block is more than 2 then it is **small**."
>
> "If the volume of a block is more than 5 then it is **large**."

"If a block is large then it is **huge**."

"If the value of a card is eight then the card is **wild**."

"If a location is below a red block then the location is **frog-covered**."

"If a location is below a blue block then the location is **toad-covered**."

"If the color of a location is the color of the block that is on the location then the location is **matched**."

"If the value of a location is the value of the tile that is on the location then the location is **matched**."

"If the locations have the same value then they are **matching**."

"If a location is below a blue block then it is **occupied**."

"If the color of an object is absent then the object is **colorless**."

"If the shape of an object is absent then the object is **shapeless**."

"If the number of covered locations near a clear location is eight then the clear location is **surrounded**."

"If the number of locations diagonal with another location is four then the location is a **center** location."

"If the number of captured locations near a clear location is more than one then the location is a **fork** location."

"If the color of a steak is red then the steak is **raw**."

"If the color of a steak is brown then the steak is **cooked**."

"If the temperature of a steak is more than 150 and the steak is brown then the steak is ***well-done***."

Figure 20: Example sentences used in teaching task elements for adjectives.

Rosie can also be taught task-specific terms for adjectives. For example, instructor might use (and then be forced to define) the adjectives "center" and "fork" while teaching Tic-Tac-Toe.

5.2.5 Comparative Adjectives

Examples of sentences used to teach task elements for task-specific terms that are comparative are displayed in Figure 21.

> "If the volume of a block is less than the volume of an object then the block is **smaller than** the object."
>
> "If the temperature of a block is less than the temperature of an object then the block is **colder than** the object."
>
> "If the weight of an object is more than the weight of another object then the former object is **heavier than** the latter object."
>
> "If the value of a card is more than the value of an object then the card is **higher than** the object."
>
> "If the value of a card is less than the value of an object then the card is **lower than** the object."
>
> "If the number of blocks on a location is less than the number of blocks on an object then the location is **weaker than** the object."
>
> "If the number of blocks on a location is more than the number of blocks on an object then the location is **stronger than** the object."
>
> "If the temperature of a block is more than the temperature of an object than the block is **warmer than** the object."
>
> "If the kelvin-value of a block is more than the kelvin-value of an object than the block is **warmer than** the object."
>
> "If an object is red then it is **warmer than** an object that is blue."

Figure 21: Example sentences used in teaching task elements for comparative adjectives.

By using the primitive comparative knowledge that Rosie knows, it is easy to teach it task-specific terms for many different comparative adjectives, such as "colder than" or "smaller than," by comparing different attributes of objects, such as "temperature" or "volume." There are many possible meanings of these terms (**C3**), such as shown with "warmer than," which vary widely based on the context.

5.2.6 Superlative Adjectives

Examples of sentences used to teach task elements for task-specific terms that are superlative adjectives are displayed in Figure 22.

"If an object is not hotter than any block then the former object is **coldest**."
"If an object is not colder than any block then the former object is **hottest**."
"If an object is not smaller than a block then the former object is **largest**."
"If a block is not larger than any object then the block is **smallest**."
"If an object is not lower than any card then the former object is **highest**."
"If an object is not higher than any card then the former object is **lowest**."
"If a card is on a deck and it is not below another card then it is a **top** card."
"If a card is on a deck and it is not on another card then it is a **bottom** card."

Figure 22: Example sentences used in teaching task elements for superlative adjectives.

With the comparative adjectives that Rosie can learn, shown above, it is easy to then teach Rosie task-specific terms for superlative adjectives. For example, the instructor can teach Rosie the meaning of "smallest" using "larger than," an object that is not larger than any object.

5.2.7 Stative Verbs

Examples of sentences used to teach task elements for task-specific terms that are stative verbs, which are often verb-derived adjectives, are displayed in Figure 23.

"If a location is under an object and the location is diagonal with the object then the object is **attackable by** the location."
"If you can move a piece onto a location then the piece is **attacking** the location."
"If an occupied location is above a clear location then the clear location is **capturable by** a block on the occupied location."

"If a location is below a block and the block is blue then the location is **occupied by** the block."

"If an occupied location is above a clear location then the clear location is **conquerable by** a block on the occupied location."

"If the color of a block is the color of an object then the block is **matched by** the object."

"If a location is below a red block then the location is **captured**."

"If a location is below a blue block then it is a **captured** location."

"If a location is below a red block then the location is **captured by** the block."

"If a blue block is on a location then the location is **captured by** the opponent."

"If the value of a location is X then the location is **captured**."

Figure 23: Example sentences used in teaching task elements for stative verbs.

Rosie can learn stative verbs, or verbs that describe a state, such as that a location is "occupied by" a block or that a square is "captured." Similar to the case of "warmer than" there are many possible meanings (**C3**) for "captured" depending on the usage, the task, and the environment. Although not explored in this work, there has been research conducted on Rosie (Mohan, 2015; Mininger & Laird, 2019) on teaching new procedural non-stative verbs, such as *stack.* For the games and puzzles (over 60) we have explored in this thesis, the agent has only been required to know only a couple procedural non-stative verbs, *move* and *write*, which are pre-coded as primitive knowledge (including action-model knowledge) in the agent. A list of the primitives required, and task-specific terms learned, for each game Rosie can learn are listed in the appendix and the dialogues for teaching all 60 games are available in the public archive created at umich.edu/~jrkirk/ijcai2019.html.

Chapter 6 Evaluating Task Learning

We have run many different experiments on Rosie in order to evaluate its handling of the problems associated with the problem characteristics, (**C1**) lack of common ground, (**C2**) compositional concepts, (**C3**) many-to-many mappings, and (**C4**) accumulative learning, using the criteria set by the desiderata, (**D1**) maximizing generality, (**D2**) minimizing communication, (**D3**) minimizing agent execution, and (**D4**) minimizing memory growth, that we defined in Chapter 1.

The games in the following evaluations were chosen so that there were some with considerable conceptual overlap (Eight Puzzle and Five Puzzle, Jealous Husbands problem and Missionaries and Cannibals, Picaria and Three Men's Morris), and others with very little overlap, such as Othello and the Frogs and Toads puzzle. Rosie correctly learns the task knowledge for all of these games.

Many of our evaluations concern the accumulation and transfer of knowledge between games and the ability of the system to continue running efficiently as multiple games are learned. The agent should be task general (**D1**), not be designed for any specific task, or set of tasks, and should be able to learn a large variety of different kinds of tasks and concepts. The agent should minimize communication during learning (**D2**) and not require large amounts of instructions or interactions. It should limit what is learned from scratch through the transfer (reuse) of knowledge, while avoiding negative knowledge transfer. After the agent has learn all task knowledge, the resulting learned representations should minimize agent execution time (**D3**), as if it had been hand-programmed with that task knowledge. As the agent accumulates task knowledge, the agent should minimize the growth in size of semantic, procedural, and working memory (**D4**), limiting what is added to the agent's memories.

To evaluate how well Rosie meets the criteria laid out in the desiderata, we designed an experiment where we teach thousands of randomly generated permutations of the teaching order of a set of games. In each permutation, every game is taught, one after another via scripts. The scripts ensure that only those concepts required for a game are taught, so if a concept has been previously learned in another game, it will not be taught in the current game. These scripts are auto-generated by an independent (python) program that takes hand-written scripts for each task in the set and analyzes which meanings have been previously taught for each task in each permutation to generate a script that teaches all tasks. To simplify the experiment execution, we created rules in Soar that internally simulate the external environments. Rather than physically setting up the puzzle in the world, a message "load state1" updates the internal world state to the stored symbolic representation. This simulation had no impact on what was learned, but it eliminates the time for typing instructions and setting up game states. All experiments were run on a desktop computer on a single core.

6.1 Evaluation of Generality (D1)

A major goal of ITL, and this thesis, is to support task learning that is general (**D1**): the agent is not limited to a small set of tasks that it can learn. We have attempted over the years to teach Rosie an increasing variety of different games in different settings pushing the total number of games that Rosie is capable of successfully learning to demonstrate generality. With advancements in the complexity of learnable hierarchical concepts, the addition of agent primitives, and the leveraging of multiple interpretations to handle ambiguity in knowledge transfer, we have made Rosie capable of learning many more games. Initially we were only able to teach Rosie a few games (Tower of Hanoi, Tic-Tac-Toe), which were then expanded into a small set of 11 games (Kirk & Laird, 2014). This was further expanded to 17 games (Kirk & Laird, 2016), then to 40 games (Kirk & Laird, 2019), and finally to the 60 games that we can now teach Rosie. An exciting aspect of the

work is we have reached a point where it is common that we do not need to make changes to the system in order to handle a new game or puzzle.

These 60 games include many versions and variants of different games and puzzles. In the following list of these game we indicate variants by (total number) or (names). They are Tower of Hanoi (3), N-Puzzle (5), Marking puzzles (Sudoku, Killer Sudoku, Jigsawdoku, KenKen, Product KenKen, Logi-5, Shuffle, Survo, Suko, Sujiko, Kakuro), Map 4-Coloring, Chess puzzles (N-Queens, N-Kings, N-Rooks, N-Bishops, N-Knights, Knight's tour, King's tour, Knight swapping, 4 Corner knight swapping), Peg solitaires (2), Card solitaires (Golf, Pyramid, Tri Peaks), River crossing puzzles (Fox, Goose & Bean, Missionaries and Cannibals, Jealous Husbands, Jealous Wives, Family crossing), Traveling Salesman in a grid, 3x3 stone games (Tic-Tac-Toe, Three Mens Morris, Picaria, Nine Holes, Connect-3), Othello, Breakthough, Frogs and Toads (2), Eight men on a raft, Stacking Frogs (3), Blocks World (2), Mazes (simple, block pushing), Sokoban, Mahjong puzzle, and a sorting puzzle. We have created a public archive as a resource for researchers that contains the teaching scripts and state representations for these games, as well as links to videos of Rosie learning some of these games. This data is available online at www.umich.edu/~jrkirk/ijcai2019.html.

We have also explored learning tasks that are isomorphisms of classic word problems, such as determining the ages of three children given constraints on their relative ages ("the age of Bob is 3 more than the age of Alice.") In this isomorphism, rather than describing initial state as part of the problem, the children are represented by blocks with values that the agent learns to modify to solve what their "ages" are, given a set of failure conditions. However, this is preliminary work that should be explored further in the future, especially the ability to be taught the state through language rather than demonstrating a physical external state.

One of the challenging aspects of the thesis has been establishing the coverage or scope of learnable tasks and task elements using our approach. So far, we have described this space by identifying different types of tasks (and concepts) that Rosie can and cannot

learn. We have created a large list of games and puzzles, in an attempt to define a taxonomy of games (and concepts) and show what parts of the definitions of various games make them learnable (or not) using our approach. Although this is not a formal specification it can, in many cases, enable someone to determine if there is a version of an arbitrary task that Rosie could learn. This list of games, along with the explanations of whether they are learnable and why, are shown in Table 2 in the Appendix.

Another way we demonstrate the generality of the learning capabilities of Rosie is to show it learning in different settings. Rosie is not only capable of learning large number of diverse games and puzzles, but is also capable of learning them in many different environment domains and agent embodiments. Rosie has learned games in real-world robotic environments: a table-top robot arm that manipulates blocks and a Fetch robot that can move and manipulate objects on a table. Rosie has also learned games in simulated environments: the April Simulator of the table-top arm, the ROS simulator (Quigley et al., 2009) of a Fetch robot, an agent internal simulation for grid based puzzles (such as Sudoku), and a simulated card game environment in an external Java application.

The work presented in this thesis enables Rosie to learn games and puzzles that are goal-oriented and deterministic in fully observable environments where the only dynamics in the environment come from changes made by the agent, teacher, or an opponent. Prior work on Rosie (Mohan et al. 2012) explains how these representations are used to perceive and act in noisy real-world environments, which is not a focus of this thesis.

Rosie cannot learn tasks that involve explicitly reasoning over arbitrary historic concepts, such as events in the past (such as needing to know the most recently placed knight in the Knight's tour puzzle). However, Rosie can often learn a variant of these games, by explicitly learning to deliberately mark when certain events happen, and then use those marks for later reasoning. For example, Rosie can learn a variant of Knight's Tour where the agent uses a red knight for capturing the next empty square, replacing it with a black knight every time it moves , in order to keep track of the most recently placed knight (the only red one) while covering the entire board. Parallel work (Mininger & Laird,

2018) has enabled Rosie to reason over historic information and handle partially observable scenarios when learning procedural tasks, but these capabilities have not yet been explored with respect to learning games and puzzles.

Furthermore, we have not explored the problem of how to deal with large numbers of objects in an environment or game, which could potentially be tackled using an attention mechanism. Large numbers of objects do not prevent Rosie from learning, only from learning and reasoning quickly. We also have not explored policy learning in the context of learning the rules of games and puzzles, so although Rosie can solve the tasks it learns, it can take a long time if the search space is large.

6.2 Evaluation of Communication (D2)

The agent should minimize communication with the teacher (**D2**) to avoid wasting their time, in part by limiting the number of interactions and avoid relearning when possible. One evaluation we have conducted related to the communication of task descriptions is related to whether concepts learned in games can decrease the amount of instruction required in future games, as measured by the total number of words required to teach a task. One would expect that if you learn the Five Puzzle, it should be easy to learn the Eight Puzzle. Transfer is possible not only for learned predicates, but also for goals, actions, and failure states.

Results from 3000 randomly generated permutations of 17 games, are shown Figure 24. It shows the number of words, on average, used to teach each game in each position in the teaching order. At position 0, no other games have been taught, and at position 16 all other games have been taught. Moving from left to right, many games require fewer words, with the largest decrease being by a factor of about three. As expected, games that have substantial conceptual overlap, such as Five-Puzzle and Eight-Puzzle which share actions (slide) and learned predicates (clear, matched, adjacent to), require very few words by the end.

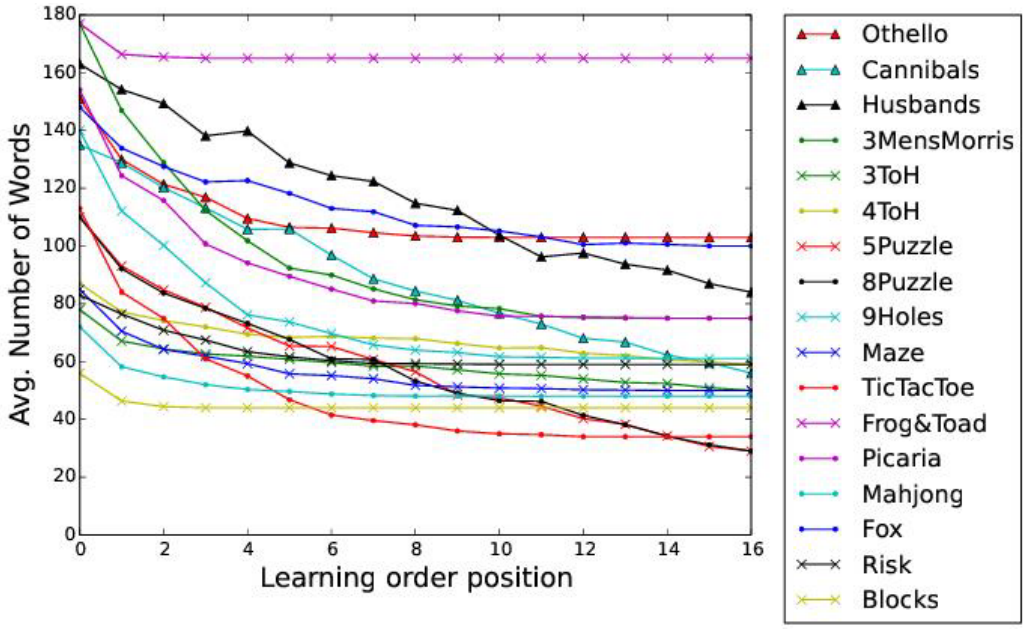


Figure 24: The number of words required to teach each game, as influenced by previously learned games. Results are averages of 3000 permutations of the 17 games.

The gradual decrease in required words going from left-to-right is a reflection of the gradual increase in the probability that a related game is previously taught. The games that have very little in common with other games conceptually still share general concepts, such as clear, and show minimal improvement: Frogs and Toads, Blocks World, Mahjong solitaire, Maze, and the Tower of Hanoi puzzles.

Recent work has expanded the number of games learnable by the agent, so this experiment was repeated with 40 games using 1000 permutations (Kirk & Laird, 2019). This is too many games to label each individually, so instead specific cases are highlighted with colors in Figure 25. More data from this experiment, with all games labeled, can be viewed in the Appendix. All 40 games are learned correctly in each permutation. The red line highlighted is for Killer Sudoku, a Sudoku variant that has constraints about the sum of values in specified section (as in KenKen). The number of words required to initially teach (position 0) this puzzle is large (239) due to the number of constraints in the puzzle. However, because of the overlap in concepts with the other tasks (Sudoku, KenKen), it

benefits the most from knowledge transfer, with a decrease of more than a factor of three. The Frogs and Toads puzzle (blue) and Blocks World puzzle (green) show the least transfer because they share only "clear" with other tasks.

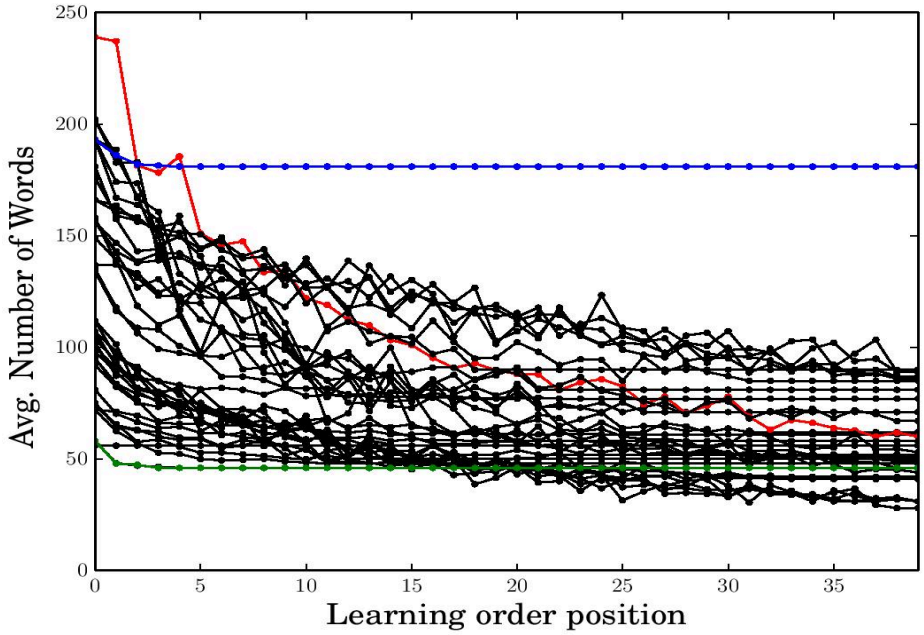


Figure 25: The number of words required to teach each of 40 games by teaching order. Results are averages of 1000 permutations.

6.3 Evaluation of Agent Processing Time (D3)

The agent should learn task representations that minimize the execution time and the learned representation should not hinder future learning (**D3**). Transfer of learned knowledge, both declarative and procedural, should decrease the overall processing time required to learn a new task, although the increase of knowledge could also potentially increase processing time.

Figure 26 shows results from the same experiment conducted for Figure 24, with the 3000 permutations of 17 games, now showing the average processing time required to teach a game based on its position in the teaching order. The processing time for a game is

measured as the total time (in milliseconds - ms) the agent took to learn the game. Because the teacher is scripted, no time is spent on speaking/typing sentences or waiting between teacher-agent interactions. The processing time is measured as the total (cumulative) time the agent takes to process each sentence delivered by the teacher (for the entire task), learn the described task elements, and generated a response. The longest total processing time for teaching an entire game is well under two seconds. The improvements, especially visible in the different game variants, are a result of not only concept transfer (which eliminates the need to teach the concept) but also transfer from the procedural rules learned (which eliminates the cost of interpretation).

The games with almost no conceptual overlap show little to no improvement (Blocks world and Tower of Hanoi) and in one case (Frogs and Toads) shows a small increase over time, due to the computational cost of added knowledge from previous games, minus the small benefit from transfer. The only benefit from transfer with Frogs and Toads is transferring the concept clear, as shown in Figure 25. There is some added computational cost to constructing the recognition structures and interpretation (described in Section 4.3) as more tasks and task elements are learned because the agent may retrieve irrelevant knowledge (and then reject it when it doesn't ground in the current context). This slowdown is minimal, even in the worst case for Frogs and Toads, which shows an increase in total average processing time from 1000 to 1104 ms for the entire game. The average processing time per instruction for Frogs and Toads, essentially Rosie's response time to a sentence, increases to ~92 ms from ~83 ms, a difference not noticeable to a human. In comparisons, the average response time over all tasks and orders is ~45 ms. The explanation for the difference in average response, a factor of ~2, between Frogs and Toads and the other tasks can be explained by the length of the sentences used to describe the Frogs puzzle, which are longer on average than many of the other tasks and involve many objects and relations. Still the difference between a 50 ms response time and a 100 ms response time are insignificant to a human: when speaking or typing sentences to Rosie, the time to type or speak dominates the interaction time.

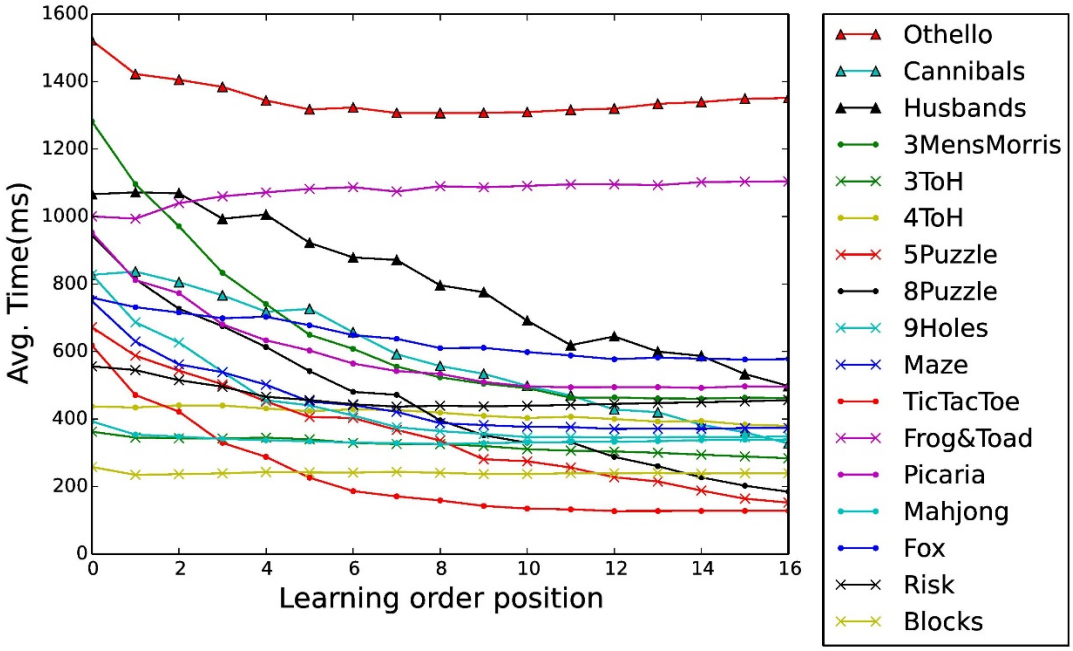


Figure 26: The total processing time required to learn each game, influenced by previously learned games. Results are averages from 3000 permutations.

In order to evaluate the execution time of the agent's post-learning operation, when it is solving tasks with the learned representations, we conducted an experiment where we measure agent processing during interpretation and then after learning during task solving. This is a measure of the impact of procedural compilation on agent performance. Our hypothesis is that evaluating the rules learned through chunking will be significantly faster than maintaining the declarative task representations in long term semantic memory and then retrieving and interpreting them by deliberately matching them against the game state each time the task is attempted. Figure 27 shows the processing time to learn each game, omitting the time taken to parse the sentences and construct the declarative predicate structure.

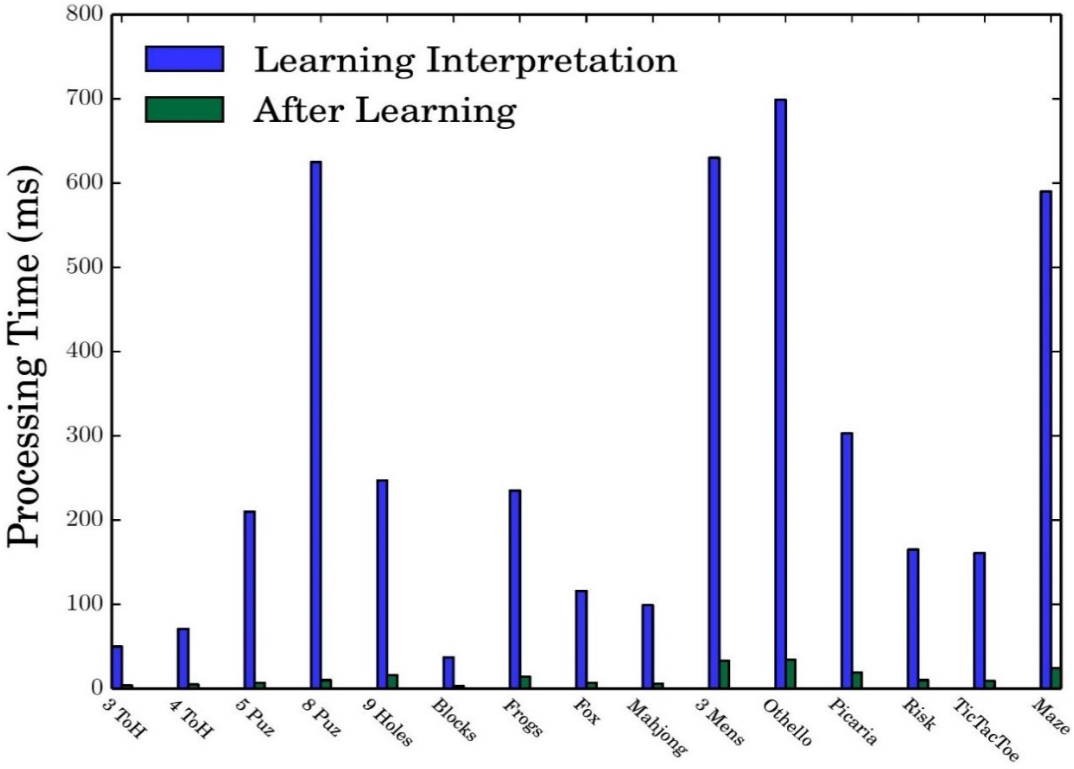


Figure 27: The processing time required to interpret (match) all concepts for each game individually compared to the processing required once learned.

For each game, the blue bars are the processing time to interpret the declarative structure when there is no transferred concepts or rules, and the green bars are the processing time required after rules have been learned. The processing required to interpret all the structures is the same processing time that would be required if the agent did not learn procedural code through chunking. The average improvement is a factor of ~20 and the processing using rules never exceeds 40 ms to propose and match all the task structures at the beginning of a game; the mean time is ~13.5 ms for the decision cycles of these tasks. This is below the roughly 50 ms cognitive cycle reported for humans in cognitive science literature, which is a commonly targeted threshold for cognitive architecture decision cycles.

6.4 Evaluation of Memory (D4)

Learning new tasks involves adding different kinds of knowledge to the agent's memories. Thus, one evaluation criterion is how the size of the working, semantic, and procedural memories (**D4**) grow as new tasks are learned, and whether that growth negatively impacts behavior by increasing the time it takes to use the memories. Soar, and most cognitive architectures, maintain semantic knowledge in long-term memories while reserving short-term working memory for data relevant to the processing of the current task.

For semantic memory (containing the declarative structures) and procedural memories (containing the procedural rules), we expect growth in memory size. In past agents developed in Soar, growth in semantic memory and procedural memory has not had a significant impact on the speed of agent processing because of the underlying implementations used for semantic memory (Derbinsky, 2012) and procedural memory, and we expect that to be true in Rosie as well. In contrast, growth in the size of working memory, even at a sublinear rate, can significantly slow agent procedural code execution.

In order to evaluate the growth in size of the agent's memories as it learns many tasks, we recorded memory data during the experiment from Figure 24 with 17 games. Figure 28 shows the growth in size of both semantic and procedural memories. As Rosie learns new tasks, knowledge in both semantic (database memory in KB) and procedural (number of rules) memory grows at a sublinear rate. This is not surprising because knowledge transfer between tasks causes a reduction in the number of procedural rules that need to be learned and the amount of new knowledge that needs to be stored in semantic memory for new tasks. The different permutations converge to the same value, confirming that the total knowledge learned for all 17 games is independent of the order in which they are taught.

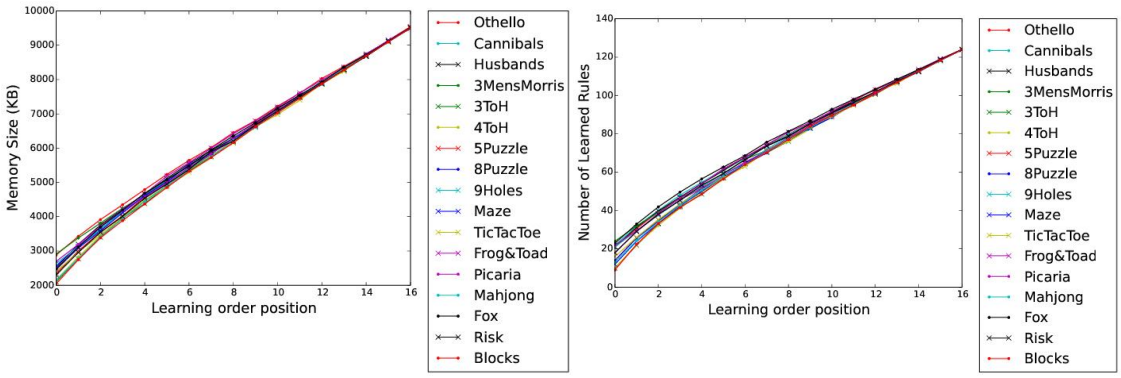


Figure 28: On the left: the cumulative growth in semantic (long-term) memory for all games. On the right: the accompanying growth in procedural memory (number of rules).

Figure 29 shows two analyses of working memory: the maximum size of working memory across learning all games, and the average number of changes to working memory for each game. Working memory is measured in working memory elements (WMEs). A WME represents each component or arc of Soar's working memory graph structure and changes are counted as the additions or removals of WMEs from working memory. The maximum size of working memory should not surpass the high-water mark set by the most computationally intensive tasks, in this case Othello, Simple Maze, or Eight Puzzle at almost 4000 WMEs, as shown in the left side of the figure. If the maximum converged to a higher level, or continued to grow, it would indicate that task-specific information is accumulating in working memory, which would likely negatively impact future processing.

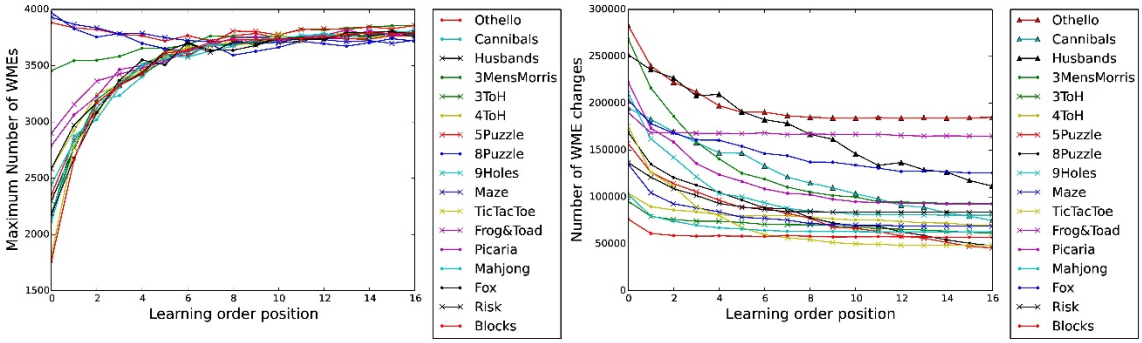


Figure 29: On the left: the growth in maximum working memory, measured in working memory elements, for each game. On the right: the number of changes to working memory to teach each task in the given order.

The number of working memory changes is an indirect measure of the total processing that occurs during the teaching of a game. It is important because the cost of rule matching is correlated with the changes to working memory in addition to the size of working memory. One possible concern is that the knowledge from previously learned tasks can "clog up" working memory and interfere with new tasks. In contrast, our results show that there is actually a decrease in WM changes through transfer from earlier tasks and that even when there are no similarities that enable transfer, there is no growth.

Chapter 7 Multiple Interpretations

Ambiguous learning situations, where multiple meanings of a word are possible (**C3**), can potentially lead to the agent incorrectly transferring knowledge. As an agent learns many tasks in many different settings, there will inevitably be many-to-many mappings between words and meanings (the components of a task). In some cases, knowledge learned in previous tasks can interfere with a new task. So far, we have avoided such scenarios in our experiments on accumulative learning. A simple example is a scenario where an agent is taught a game where their pieces are red, and in the next game they are taught, the opponent's pieces are red. In this case, Rosie would incorrectly transfer knowledge, overgeneralizing, when encounter the term "their pieces" in the second game.

During the task element learning process, sources of ambiguity can arise that make it difficult to find the correct interpretation and can cause interference when trying to transfer knowledge from previous tasks. These sources include:

> ***Multiple Definitions:*** Due to the many-to-many mappings between words (**C3**) and meanings across tasks and the teacher's lack of common knowledge(**C1**), the agent may learn multiple meanings for the same word (or know multiple meanings of primitive concepts).
>
> ***Environmental Distractors:*** The state can contain objects and features that although not relevant to the described concept, can create ambiguity when the agent attempts to ground the representations.

We have extended Rosie so that it can effectively learn and transfer knowledge in more difficult learning scenarios, where ambiguity and learning distractors are present. This extension modifies the learning strategy to enable Rosie to create, analyze, and debug *multiple* interpretations of task elements in order to handle scenarios where ambiguity and

knowledge interference can negatively impact the ability to accurately learn and transfer knowledge. Our approach also enables the agent to use the analysis of these interpretations to quickly communicate with the instructor to resolve sources of ambiguity when automated reasoning fails. Our extension improves Rosie's ability to correctly learn polysemic words and handle the *many-to-many mappings* possible from words to definitions: a word can have many task-specific meanings and a meaning can be represented by different words in different tasks.

For example, depending on the context, the polysemic word *clear* can mean that something is uncovered or that it is transparent or that it is unmarked. This extension has also increased the complexity of the hierarchical task elements and the breadth of terms and games that the agent can learn.

Below we present our learning approach, which enables Rosie to communicate, through quick, short interactions, to resolve ambiguity and select correct interpretations. We evaluate the agent's ability to correctly generalize, disambiguate, and transfer concepts across variations in natural language descriptions, world representation, and game instances, showing transfer across tasks and within tasks, with and without interference. We examine the number of words required to teach tasks across cases of no transfer, positive transfer, and interference from prior tasks.

7.1 Creating Multiple Interpretations of Task Elements

To ensure that it correctly interprets an ambiguous situation, Rosie generates *all* possible recognition structures, *f(x)* from Equation 1 reproduced below, for each known meaning of the defining terms, $f_i(x)$.

$$f(x_1, \ldots, x_m) = \bigcap_{i=1}^{n} f_i(x_j) \qquad 1 \leq j \leq m \tag{1}$$

Because Rosie generates recognition structures, *f(x)*, for all possible combinations of the known meaning for the given terms, the number of structures grows exponentially with respect to *n*, the number of $f_i(x)$ terms, and the number of definitions for each term. However, descriptions are limited to a single sentence and the number of terms (*n*) with multiple meanings is rarely more than 3 (the maximum recorded in all 60 games that the agent has learned is 4), so in practice, it is computationally feasible to generate all interpretations.

To determine the correct interpretation from the set of generated structures, Rosie leverages the situated external state example. If it finds that only one of the recognition structures can be satisfied or detected, it learns this interpretation. If instead the agent finds that multiple structures from different interpretations can be satisfied in the current state, the agent analyzes each of the potential matching interpretations to try to find ways to differentiate them. Based off this analysis Rosie, uses one of three different strategies to try to determine which interpretation is correct. In this analysis, the agent looks for task elements that return different numbers of results (the number of occurrences from each interpretation).

First, the agent determines if it can find a difference for the highest task element in the hierarchy *f(x)*, the one describing a goal, action, or failure condition, such as when the different interpretations of an action result in different numbers of actions being detected. Rosie counts the relative occurrences and determines the correct interpretation by asking the teacher: “How many actions are present X or Y?”

If the agent cannot detect a difference between the numbers of results for different interpretations of that task element *f(x)*, it examines numerical differences in the occurrence of the supporting task elements $f_i(x)$, such as *clear(x)*. If Rosie finds a task element $f_i(x)$ that produces different numbers of results in different interpretations, Rosie uses this difference to generate a similar disambiguating question: “How many clear locations are present X or Y?”

Finally, if the agent fails to detect any differences in the number of occurrences of task elements in different interpretation, it abandons attempting to resolve the differences through questioning, and asks the teacher to provide a different state demonstration: "Can you setup another state that contains the [goal, action, failure]?" The hope is that in this new state, the agent can satisfy and detect only one interpretation, or if there are multiple, find a numerical difference in the occurrence of one of the task elements, using the first two strategies. The agent will continue to ask for new state demonstrations until it can select a single interpretation.

7.2 Ambiguity Case Study

We use a series of case studies to illustrate the agent's ability to handle interference using the process described in the previous section: determining if multiple interpretations can be satisfied, detecting the relative occurrences of each task element in the competing interpretations, and asking disambiguating questions to determine the correct interpretation.

7.2.1 CASE 1: Ambiguous Word Meanings

One source of ambiguity is that the agent may know multiple definitions for the same word. Consider when the agent has previously learned two definitions for clear: that a location is not below anything (from the blocks world) and that a location is unmarked (from KenKen). Figure 30 shows the internal symbolic representation (right side) Rosie generates of the external state (left side) for a version of the Frogs and Toads side-swapping puzzle. The teacher's description of the first action of the puzzle, moving a toad, includes the term clear, as shown in the dialog below. Instead of asking for a definition of clear, Rosie attempts to use its existing definitions, which leads to two recognition structures, shown in Figure 31, one for each interpretation of clear.

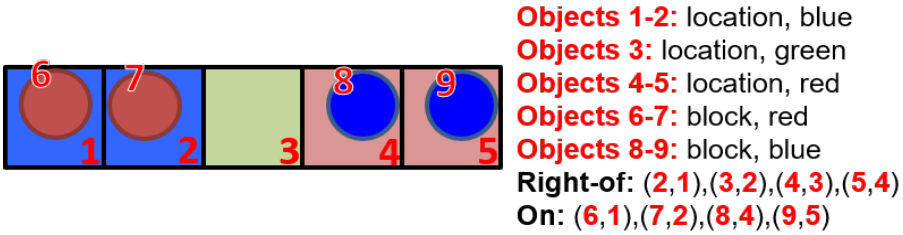


Figure 30: The internal state generated by the agent for the Frogs and Toads puzzle, with objects identified by red indexes. On the right are unary features and binary relations that the agent extracts from the state.

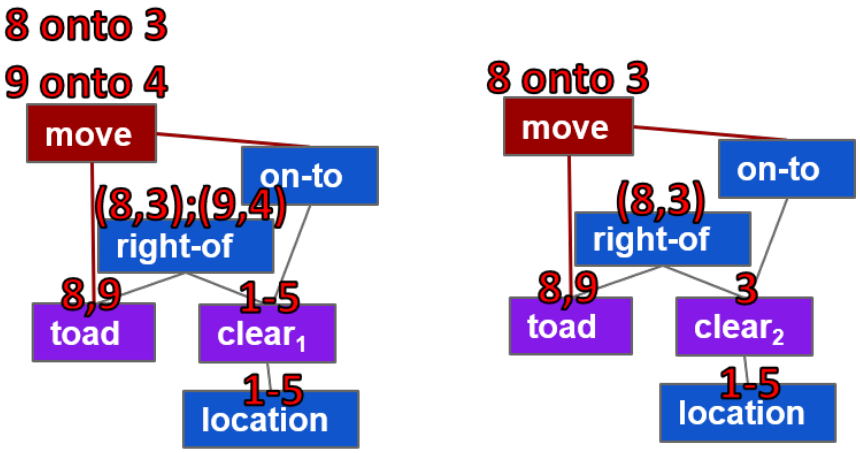


Figure 31: Recognition structures created for two interpretations of an action. Red values indicate the indexes of objects in the environment that results from grounding the structure to the external state.

> If a toad is to the right of a clear location then you can move the toad onto the location.
> **Rosie: How many actions are present two or one?**
> There is one.

Rosie analyzes these structures to find a difference in how they map to the current state. In this case, they generate different numbers of actions due to the agent believing that for the structure on the left, where clear is an unmarked location, location 4 is clear. Rosie uses this difference to disambiguate between the two interpretations by asking a simple question about the relative number of actions. When the teacher responds "There is one," Rosie can

determine that the representation on the right, that only detects the action move (object) 8 onto (object) 3, is the correct structure, and then uses it for learning this task element.

7.2.2 CASE 2: Ambiguous External State

Another source of ambiguity is that the state can contain objects, features, and relations that are distractors, irrelevant to the concept the agent is trying to learn. For example, consider the state given in Figure 32 for teaching the second action for the Frogs and Toads puzzle, jumping over a frog. When given the following action description, Rosie once again generates two recognition structures for each meaning of clear.

> If a toad is to the right of a frog that is right of a clear location then you can move the toad onto the location.
> **Rosie: How many clear locations are there one or five?**
> There is one.

When Rosie analyzes how these interpretations ground to the current example, they refer to the same action: move object 9 onto object 3. This is due to the constraint provided by the position of the frog: even in the interpretation where all locations are clear, the toad (object 9) can only jump over the frog (object 7). Although both interpretations ground to the same action, Rosie needs to determine which interpretation of clear is correct so that the correct condition is learned. Rosie finds a distinguishing result further down the recognition structures, where the predicate clear produces a different number of objects for each interpretation. As shown above, Rosie uses that difference to generate a disambiguating question. When the teacher responds indicating there is only one clear location, Rosie selects the correct interpretation structure to learn.

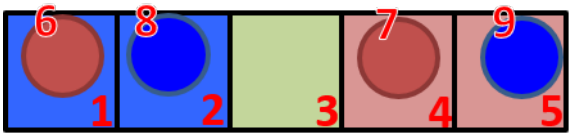


Figure 32: A representation of the internal state generated by the agent for describing jumping in the Frogs and Toads puzzle.

7.2.3 CASE 3: Symmetric State Ambiguity

In some scenarios, the occurrences of task elements in different interpretations is the same. This often occurs in states with symmetry. For example, consider when Rosie is learning the goal of Tic-Tac-Toe, but where the state contains winning conditions for both Rosie and the opponent. If Rosie knows multiple definitions for *captured* from previous games where the ownership of red and blue pieces have swapped, from its perspective, there is no way to disambiguate between the different interpretations. When the other strategies fail, Rosie's final disambiguation strategy is to ask the teacher to demonstrate another example of the concept in the environment: "Can you setup another state that contains the goal?" If the teacher creates a state that contains only the goal, or a state with more red blocks placed than blue ones, the agent can determine the correct interpretation (automatically in the first case and by asking about the number of *captured* locations in the second).

7.3 Creating Synonym/Antonym Interpretations

To further expand the ability of the agent to transfer knowledge to new tasks and situations, we created an option for Rosie to use a synonym/antonym table. Using this feature presents similar tradeoffs as before, between quickness of learning and correctness of learning. It is much easier to avoid incorrect learning by not attempting to automatically transfer any knowledge. This table was added to Rosie's primitive knowledge in semantic memory, allowing Rosie, if specified, to lookup common antonyms and synonyms of a given word, such as "clear," and automatically generate interpretations that replace the predicate with a synonym, such as *empty*, or antonym, such as *~filled*. Table 4 below shows the content and format of this knowledge.

Word	**Synonyms**	**Antonyms**
clear	empty, uncovered, transparent	covered, filled
covered	occupied, filled	clear, empty, uncovered
empty	clear, uncovered, transparent	covered, filled
filled	full, covered	empty

box	cube	

Table 4: A lookup table of common synonyms and antonyms for common words.

The ability to use synonyms and antonyms allows the examination of a greater space of interpretations without requiring additional declarative knowledge or teaching interactions: it can increase cases of knowledge transfer and decrease the amount of required instruction. However, using this ability also increases the chance of incorrectly transferring knowledge because it considers more options and doesn't verify the knowledge that is transferred if it leads to a single valid (successfully maps to the world) interpretation. This can occur in cases where the incorrect knowledge does not prevent Rosie from recognizing an instance of the term, such as "clear," in the environment. For example, if Rosie has learned a definition for *filled* (from Sudoku) that a "filled" location "has a value," when later learning Tower of Hanoi with blocks, Rosie will try *~filled* for the meaning of "clear." In this case because there are no values for the blocks Rosie will be successful at detecting *~filled* even though for this context Rosie needs a different meaning of "clear" ("not below anything"). If the agent detects multiple interpretations that map to the agent's internal model then it can still potentially learn which meaning is correct through disambiguating interactions with the teacher, as described above. Examples of each of these situations are discussed in the evaluations in Section 7.4.3.

7.4 Evaluation

To analyze knowledge transfer in different scenarios, with and without interference, we performed a set of experiments similar to the experiments in Chapter 6, where we teach different order permutations of a sequence of tasks. In these experiments, we teach small clusters of three games and examine the number of words required to teach tasks across cases of no transfer, positive transfer, and interference from prior tasks.

7.4.1 Positive Transfer

The experimental results for the task clusters **A-D** are shown in Figure 33. For each cluster, the results are averaged over all (6) possible permutations of the three tasks. To explore positive transfer, we selected very similar tasks with large conceptual overlap for each task clusters: (**A**) Tic-Tac-Toe, Three Mens Morris, and Nine Holes; (**B**) Killer Sudoku, KenKen, and Sudoku; (**C**) N Queens, N Rooks, and N Kings; and (**D**) Jealous Husbands, Fox, Goose, and Bean, and Missionaries and Cannibals. The Plots A-D in Figure 33 show the dramatic effects of transfer in clusters of similar tasks, in some cases, such as Sudoku cluster **B**, reducing the number of words required to learn roughly 4:1. Task learning approaches that learn mappings directly to nonsymbolic representation have failed to replicate this type of task transfer, which leads to dramatic learning speed up.

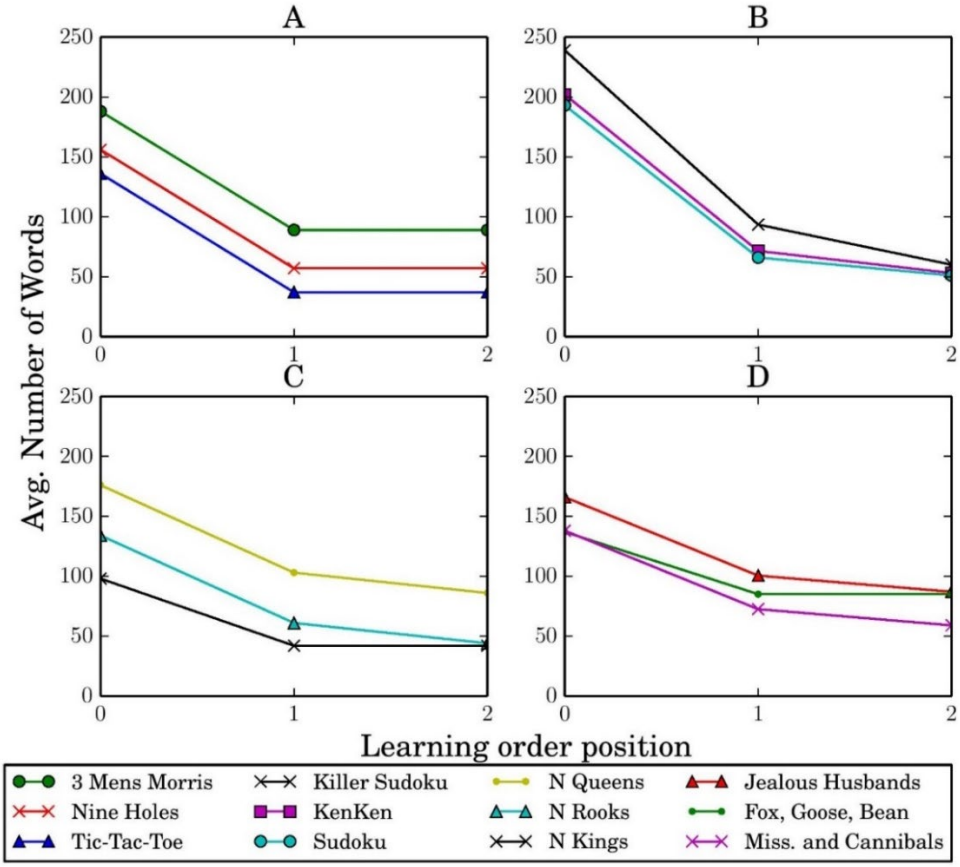


Figure 33: Number of words required to teach clusters of closely related tasks A-D.

7.4.2 No Transfer

The experimental results for the task clusters **E-H** are shown in Figure 34. Again, the results are averaged over all (6) possible permutations of the three tasks in each cluster. To explore absence of transfer, we selected dissimilar tasks with little conceptual overlap for each task clusters. The entire set is the same as before, just arranged in different clusters. The unrelated task clusters are: (**E**) Killer Sudoku, Three Mens Morris, and N Queens; (**F**) Sudoku, Nine Holes, and Missionaries and Cannibals; (**G**) Fox, Goose, and Bean, Tic-Tac-Toe, and N Rooks; and (**H**) KenKen, Jealous Husbands, and N Kings.

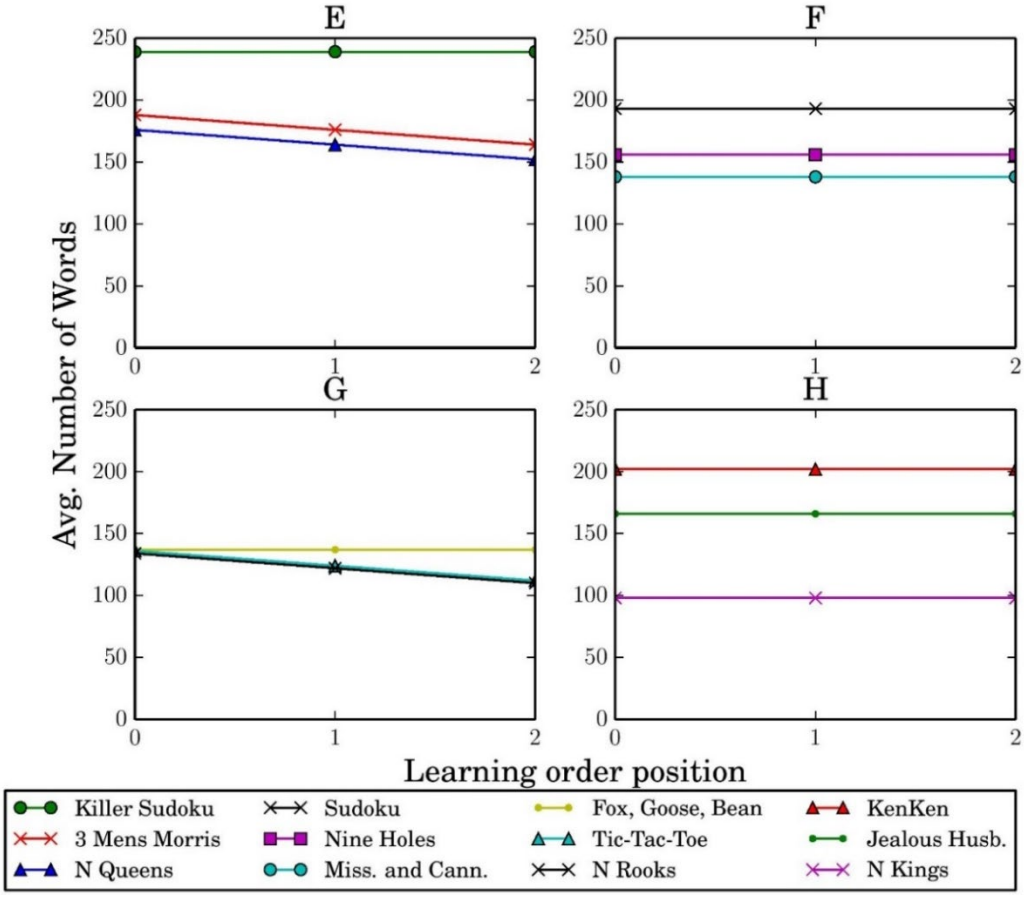


Figure 34: Number of words required to teach clusters of unrelated tasks E-H.

Plots E-H shows almost no transfer between the unrelated tasks, however there is still some positive transfer (in E and G) due to a slight conceptual overlap of very common terms: *clear* and *available*.

7.4.3 Negative Transfer

To explore negative transfer, we selected tasks from the previous experiments, but modified the simulated environments and term usage in the task element descriptions in order to create problematic cases for our learning approach (which is greedy and doesn't ask for extra verification when transferring knowledge). In these cases, not only can there be an increase in the required number of words to teach, but incorrect knowledge can be learned without the agent (or teacher) being aware of it. The experimental results for task cluster **G**, which consists of modified versions of KenKen, Frogs and Toads, and the Eight Puzzle, are shown in Figure 35. In this case, rather than an average across permutations, we show the results of each of the 6 possible permutations of the three tasks, with a graph for each task. The bar graphs for each task show the number of words used to teach the task in each position in the learning order. There are only five results on each graph rather than six, because we do not show the duplicate results for learning position 0 (both times it will be the same).

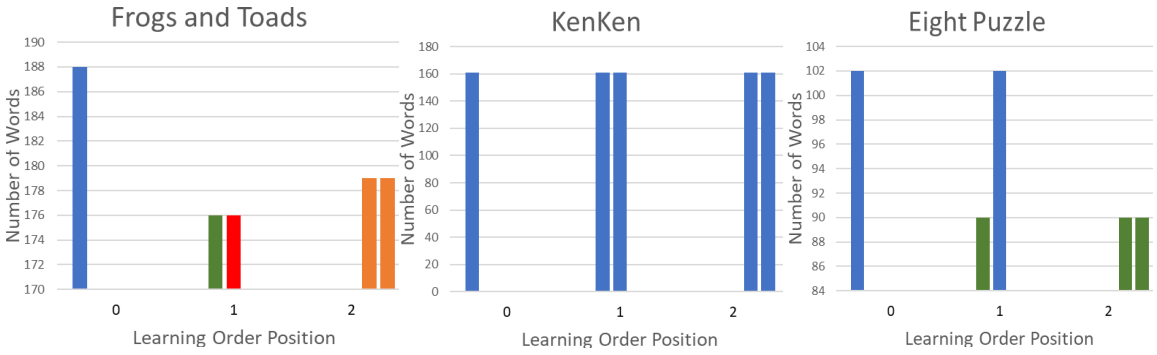


Figure 35: Number of words required to teach all permutations of task cluster G. The colors are used to highlight the cases of no transfer (in blue), positive transfer (in green), negative transfer (in orange), and incorrect knowledge transfer (in red).

Results shown in blue are cases where there was no transfer. In this version, KenKen has no overlapping concepts and shows no transfer. Results shown in green are cases where there was positive transfer, reducing the number of words required to teach. The cases of positive transfer occur for both Frogs and Toads and Eight Puzzle: Rosie takes advantage of synonym and antonym knowledge to transfer knowledge of the concepts *clear*, *empty*, and *filled*. Results shown in orange are cases where there is negative transfer: there is still overall positive transfer from learning position 0, but a reduction from the positive transfer (green) case. This occurs in the Frogs and Toads puzzle, in the permutations where the task is positioned at the end of the teacher order, because Rosie is forced to ask, "How many actions are there?" to differentiate between the multiple meanings of clear that it has learned.

Finally shown in red is the case where incorrect knowledge is transferred. In this case, Rosie transfers the meaning of "clear" from KenKen, *not marked*, when it needs the meaning *not covered*, and the state of the task environment is such that it still detects a single valid interpretation despite this meaning being incorrect. Rosie requires fewer words to learn in this case but learns incorrect groundings. If Rosie was less greedy in its attempt to reduce the number of interactions and amount of teacher instruction, it could ask for verification in transfer cases like this. This approach, and how to recover from incorrect knowledge learning, will be explored in the future.

Chapter 8 Discussion and Conclusion

Learning novel tasks through online instruction from 'scratch' (primitive non-task specific knowledge) presents many challenges for an Interactive Task Learning agent. The lack of common ground (**C1**) between the agent and the teacher, their joint inability to access each other's internal models of the environment or view the contents of each other's memories, makes it difficult for the teacher and agent to effectively communicate by using appropriate terms (that are already known) and provide the necessary knowledge (that is missing). The many-to-many possible mappings between words and meanings (**C3**) further complicates the problem caused by lack of common ground because it creates a large space of possible interpretations. Without access to the teacher's internal model, the agent cannot know the intended context specific meaning of a term, such as "clear," out of a huge space of options, such as *not covered, transparent, unmarked, etc*. The compositional nature of the meanings of concepts (**C2**) used to define tasks necessitates that the mappings are not only between synonymous concepts, such as "clear" and *transparent*, but also between a term, such as "adjacent," and a combination of concepts, such as *next to each other* but *not dialog with each other*. Compounding the difficulty of the problem further is the desire for the agent to accumulate knowledge over many tasks (**C4**), which introduces the challenge of creating and maintaining knowledge representations that enable the agent to transfer knowledge.

We have attempted to address each of these problem characteristics in the design of our learning process. Rosie learns symbolic representations of task elements that are interpretable and can be used to communicate about its knowledge representations of the task and how they map to its internal model of the environment, to help establish common ground with the teacher (**C1**). Rosie creates multiple interpretations of task elements to evaluate, through comparative analysis and interactive debugging, the many-to-many possible mappings between terms and meanings (**C3**). Rosie learns hierarchical compositions of concepts (**C2**) through the recognition structures it creates and the recursive learning algorithm it uses to learn new predicates for task-specific terms. These hierarchical recognition structures also support the transfer and accumulation of

knowledge overtime (**C4**). The modularity of the hierarchical recognition structures enables Rosie to find and relearn small portions of the structure that it cannot currently ground, such as learning a new meaning for a term, without relearning the entire structure from scratch.

These problem characteristics make learning challenging given the objectives of ITL as reflected in the desiderata we have created. These desiderata serve both as guiding criteria for agent design and metrics for evaluation. The agent should attempt to maximize its level of generality (**D1**) and not be limited to a small set of tasks, task knowledge, or types of tasks that it can learn. The agent should attempt to minimize the amount of agent-teacher communication (**D2**) to reduce the effort and time required for the human instructor to teach a new task. The agent should learn representations of the task and task knowledge that minimize its execution time (**D3**) during agent's processing of the task, or during other tasks. The agent should minimize memory size growth (**D4**) – avoiding growth that significantly increases the time it takes to use its memories.

We use these desiderata to evaluate the learning of the agent. Rosie's learning approach is general (**D1**) – it learns goal-oriented tasks that can be represented with a problem space formulation. To demonstrate generality, we have shown that the agent can learn a large number (60) of different tasks in different domains and environments, including various robotic embodiments and simulated environments. To evaluate agent-teacher communication (**D2**), agent execution time (**D3**), and memory size growth (**D4**), we ran experiments teaching many different permutations of a sequence of different tasks and analyzed the relevant data. In nearly all cases, the number of words it takes to teach new tasks decreases as it accumulates tasks due to transfer of knowledge. The procedural, native (to Soar) representations that the agent learns enable it to process and apply task knowledge much faster than when they were initially interpreted and the accumulation of knowledge over many tasks does not significantly increase the agent's decision cycle time. The growth in memory does not significantly impact (slow down) the speed of memory usage for procedural, semantic, and working memories.

A broader question that this research, and the burgeoning research area of Interactive Task Learning, tries to address is: how should AI agents be trained or taught? The current popular answer in the AI community is machine learning, and especially deep learning – learning in neural

nets through large amounts of training data. This approach has been successful at making AI agents the best at many specific tasks. For the past 3 decades, AI research has focused on making AI agents the best in the world at many different tasks: at Chess (DeepBlue), at Go (AlphaGo), and at Poker (Libratus).

However, when these agents are 'taught' the game, they still require a human to pre-encode the legal actions of the game and a reward function that helps approximate the goal. Moreover, these agents cannot simply switch from learning one task, such as Chess, to learning another, such as Poker or Go, without extensive changes to their underlying code. These code changes often require human supervised experimentation to select the proper parameters and settings for 'learning' the task in the new domain. We use quotes in jest to make a point: none of these approaches are actually *learning* the structure and definition of the task, they start with this task knowledge, and learn how to better *perform* the task.

These types of approaches require that for every single task, the agent must be programmed, debugged, trained, and evaluated offline in order for it to perform the task. Deep learning approaches have also not demonstrated how knowledge transfer between tasks, such as those explored in this thesis, can occur. A Deep RL model that has been trained on the game Breakout has been shown to fail horribly when a minor change is made to the game by simply moving the paddle down a few pixels (Kansky et al., 2017). The problem of how a generally intelligent agent, that is task agnostic, could acquire the basic knowledge of these tasks, without being handcrafted to only understand and perform that single task, has been largely ignored.

The representations learned from these Deep Learning approaches are also not interpretable. Neither a human, nor a computer program, can examine the learned representations to understand what was learned. Uninterpretable representations impede knowledge transfer and the ability to generalize and have other negative ramifications. Not being able to understand why a machine made a decision drastically hurts the ability to debug the agent, understand the choices it made, and assign fault (even possibly the legal sense of fault). With the growing proliferation of smart machines, from personal assistants (such as Siri) to self-driving cars, these properties are beginning to come under greater scrutiny and will continue to be an important problem for AI researchers to solve.

In contrast, in our approach the learned representations are both interpretable and native to the underlying architecture. Achieving both learning properties simultaneously is a difficult task, one that most systems or architectures fundamentally cannot support. In terms of interpretability, most agent architectures are written in programming language (C++, python, etc.) that can be interpreted, but the learned representations that the agent acquires is rarely in the form of a new segment of that code. One exception is the work of Hinrichs and Forbus (2014), where the English description is translated into GDL, the Game Description Language, which is interpretable, but is not the same type of code as the Companions agent that interprets them. This type of approach, where the learned representation is a high-level programming language, is the closest task learning research to easily interpretable representations, but also the furthest from native representations (they must be interpreted). Learning a native representation often requires that the representation is not the same as the underlying agent.

In our approach, the underlying agent architecture, Soar, is built on symbolic representations, making it possible to learn representations that are both native and interpretable. Although not immediately comprehensible, a person that has been taught the representation can in most case read and interpret the learned representations easily. Related research (Ramaraj & Laird, 2018) has shown that Rosie can use these representations to communicate about specific grounding failures ("A medium block is not on a large block.") and generate language descriptions ("Clear means that it is not below a block.") These sentences are not stored or generated from the sentences that were used when their meaning was initially learned. They are generated from the recognition structures the agent learns.

We have also shown that the interpretability of these representations allows Rosie to analyze and debug what was learned through straightforward strategies: it can understand what parts of a representation are and are not successfully being grounded, and then ask questions. This allows Rosie to move from a task it has learned to an unknown similar one, or from one version of a task (marking X's and O's Tic-Tac-Toe) to another version (placing black and white blocks Tic-Tac-Toe), and reuse the learned representations from the prior task, by identifying the small portions of knowledge that are incorrect for this new context and engaging the teacher to quickly

learn the new meanings. In these scenarios the agent can transfer the majority of the task knowledge.

8.1 Current Limitations and Future Work

A limitation of this work is that it is computationally expensive to detect task elements for states with large number of objects and relations (such as in Chess). Rosie's inability to quickly reason over large numbers of objects in a single state is why we often teach versions of games that have a limited number of objects, such as the 5x5 version of Othello. We hypothesize that some type of attention mechanism together with deliberate reasoning is needed so that all concepts are not simultaneously computed for every state.

A second limitation is that the language Rosie understands, although sufficient for these games, is more rigid than natural language. An instructor would not be able to teach Rosie without first becoming familiar with the types of sentences Rosie can understand. Rosie also cannot learn concepts that deal with historical events. Future work should focus on allowing Rosie to learn concepts that interface with Soar's episodic memory. Another shortcoming of our work is that it assumes error-free unambiguous instructions. In the future, we plan to study how an agent can recovery from incorrect knowledge through instructions.

Another limitation of our approach is that it can be overaggressive in generalizing or transferring concepts. We assume that the agent is situated in a grounded example where the described concept is present, but it is also possible that many such concepts or closely related concepts are present, and that could lead to incorrect generalization. For example, if we teach a version of Tic-Tac-Toe with black and red pieces after learning Red-Blue Tic-Tac-Toe, rather than prompting the teacher for a new definition of 'yours,' Rosie will incorrectly latch on to the opponent's pieces, which due to the symmetrical nature of the game will still lead to detectable actions given those definitions. Solutions to these problems are being explored in parallel research, where rather than just asking for specific redefinitions, or making generalization assumptions, the agent communicates why something didn't match or how known concepts do match to allow for the teacher to give quicker, more useful instructions. This work explores how in an ITL setting,

the teacher and agent can negotiate and communication their knowledge states and discrepancies, by asking and answering questions such as “do you see the goal,” “why did it fail,” and “what part of the goal didnt match? ”

In the future, we plan to expand the types of learnable concepts by adding primitive knowledge (quantifiers, functions, actions), support for direct disjunction (“If it is green or blue then it is a frog”), and grammar constructions. These will allow us to study transfer and scaling of knowledge across a wider variety of more complex games. Finally, our focus has been on learning the rules of a game, not on how to play well. We plan on exploring learning to perform a game well, both through additional instruction, such as teaching heuristics, action models, and value-functions, but also by learning from experience using Soar reinforcement learning mechanism.

8.2 Future of Interactive Task Learning

One of the long-term goals for the future of Interactive Task Learning is to replace programming as the primary method of adding new capabilities and knowledge to an agent with online interactive instruction. The agent architecture should supply general memory, problem solving, and reasoning mechanisms, but knowledge specific to new tasks, situations, collaborative agents, and environments should be learnable by the agent through natural online interactions with an expert (of the new task). One possibility to explore is the capabilities required to achieve ‘universal instructability,’ where the instruction mechanisms are sufficient to learn all knowledge encoded in an agent. This is not currently the case in our approach to learning goal-oriented tasks with Rosie, as there are many types of knowledge that Rosie cannot learn, such as explicit historical references, but this is not a fundamental limitation of the approach, just a current limitation we have yet to address. Future work could extend Rosie to learn concepts for games and puzzles that interface with Soar’s episodic memory and to enable this type of knowledge learning.

Historically the problem of translating from high level task descriptions to executable machine behavior has been accomplished by programmers, requiring expertise of the task being instructed ***and*** the underlying agent architecture ***and*** the associated programming language ***and*** any existing agent code. Programming an agent for novel tasks requires design, coding, testing,

and debugging. Reuse of existing functionality requires extensive knowledge of the existing code and is usually limited and difficult to achieve. There is no automatic leveraging (or transfer) of existing knowledge or code, or any guarantees that the added functionality will not break existing agent functionality. Learning though instruction allows the rapid acquiring of novel tasks, which will be essential if we want more generally intelligent autonomous agents that are not handicapped by their need to be programmed for every kind of task they encounter.

Appendix

	Game/ Puzzle	player #s	Type	Brief Rule Summary	External link	Can learn ?	Why Not
1	Tower of Hanoi	1	blocks	Move a stack of N discs (or blocks) all with different sizes that are ordered with the smallest on top, to a destination pillar (or location) by moving the objects individually onto empty pillars or discs larger than themselves	https://en.wikipedia.org/wiki/Tower_of_Hanoi	YES	
2	Eight (N) Puzzle	1	blocks	Given a 3x3 grid, slide tiles (or blocks) to adjacent clear grid locations to create the goal state with ordered numbers (or colors)	https://en.wikipedia.org/wiki/15_puzzle	YES	
3	Blocks world puzzles	1	blocks	Move clear blocks onto clear destinations in order to create a specified arrangement of those blocks.		YES	
4	Frogs and Toads	1	blocks	Given N frogs and M toads split on separate sides of a (N+M+1) strip of squares, move the frogs and toads one square, or two by jumping over an occupied square, in order to swap the sides.	https://en.wikipedia.org/wiki/Toads_and_Frogs	YES	
5	Stacking frogs	1	blocks	Given a strip of Nx1 squares (or lily pads) each covered by a single frog, move a stack of frogs on a square the same distance as the size of the stack, until	https://www.youtube.com/watch?v=X3HDnrehyDM	YES	
6	Lazy Stacking frogs	1	blocks	Stacking frogs, but end on a specified square (lily pad)	https://www.youtube.com/watch?v=X3HDnrehyDM	YES	
7	Missionaries and Cannibals	1	river cross	Move the missionaries and cannibals from one bank to the other using the boat, with capacity of two, and do not let there be more cannibals on a bank than missionaries.	https://en.wikipedia.org/wiki/Missionaries_and_cannibals_problem	YES	

8	Fox, Goose, & Beans	1	river cross	Move the fox, goose, and beans to the opposite bank, carrying one at a time, and do not let the goose alone with the beans or fox alone with the goose.	https://en.wikipedia.org/wiki/Fox,_goose_and_bag_of_beans_puzzle	YES	
9	Manager, Actor	1	river cross	Move the actors and their managers from one bank to the other using the boat, with capacity of two, and do not let an actor be on a bank with another manager when their manager is not present.	http://aperiodical.com/2016/11/a-more-equitable-statement-of-the-jealous-husbands-puzzle/	YES	
10	Jealous Husbands	1	river cross	Move the couples from one bank to the other using the boat, with capacity of two, and do not let a woman be on a bank with another man when her husband is not present.	https://brilliant.org/problems/the-jealous-husbands-problem-extended/	YES	
11	Family Crossing	1	river cross	Move adults and children from one bank to the other using the boat. The boat can hold one person, or two children.		YES	
12	Peg solitaire	1	blocks	Given an arrangement of holes, filled with pegs except for one, use a peg to jump over and remove another, until only one peg remains.	https://en.wikipedia.org/wiki/Peg_solitaire	YES	
13	Mahjong solitaire	1	blocks	Given a stack of assorted tiles, remove two tiles at a time if they are both clear (not covered or surrounded) and have the same symbol, until no tiles remain.	https://en.wikipedia.org/wiki/Mahjong_solitaire	YES	
14	Simple maze	1	blocks /grid	Move an object along adjacent clear squares, avoiding walls, to navigate to a destination square.	https://en.wikipedia.org/wiki/Maze	YES	
15	Block pushing maze	1	blocks /grid	Move an object along adjacent clear squares, avoiding walls, to navigate to a destination square. You can push blocks covering squares onto other clear squares.		YES	
16	Sokoban	1	blocks /grid	Move the player along adjacent clear squares, avoiding walls. Solve by pushing specified blocks onto their desired locations.	https://en.wikipedia.org/wiki/Sokoban	YES	
17	Map (4) Colorin	1	grid	Color the sections of a map with 4 different colors, completing all	https://en.wikipedia.org/wiki/	YES	

	g Problem			sections without any adjacent sections having the same color.	Four_color_theorem		
18	Sorting Puzzle	1	blocks	Move objects of each color (or other attribute) into a specified location, so that each location only contains blocks of that color.		YES	
19	Knight's tour	1	blocks /grid	Given a chess board move a knight (using the rules of knight jumping) around the board such that each square is visited exactly once.	https://en.wikipedia.org/wiki/Knight%27s_tour	YES	
20	Sudoku	1	mark/ grid	Fill a NxN grid with numbers 1-N such that no two grid locations in the same section, row, or column have the same number	https://en.wikipedia.org/wiki/Sudoku	YES	
21	Killer Sudoku	1	mark/ grid	Sudoku, but with no initial filled squares and additional regions that must contain numbers that sum to a specified value	https://en.wikipedia.org/wiki/Killer_sudoku	YES	
22	Jigsawdoku	1	mark/ grid	Sudoku, but with irregular shaped sections rather than squares.	http://www.thepuzzleclub.com/jigsaw/	YES	
23	Logi-5	1	mark/ grid	Jigsawdoku, but using letters (A, B, C...) instead of numbers (1,2,3…)	http://www.thepuzzleclub.com/logi5/	YES	
24	KenKen	1	mark/ grid	Sudoku, but the numbers in a section must also sum, multiply, divide, or subtract to achieve a specified number	https://en.wikipedia.org/wiki/KenKen	YES	
25	Tic-Tac-Toe	2	blocks /grid	Place stones onto empty squares on a 3x3 grid, win by achieving three in a row of your pieces before your opponent does.	https://en.wikipedia.org/wiki/Tic-tac-toe	YES	
26	Connect-4	2	blocks /grid	Drop pieces into a vertical (gravity constrained) grid, win by achieving four pieces in a row before your opponent does.	https://en.wikipedia.org/wiki/Connect_Four	YES	
27	Othello/ Reversi	2	blocks /grid	Place your stone on a clear square on a grid, such that all the squares between that one and another of your stones are opponent pieces. Flip those opponent pieces so that they become yours. Win by having more captured squares than your opponent once all are covered.	https://en.wikipedia.org/wiki/Reversi	YES	

28	Breakthrough	2	blocks /grid	Given a grid board with pawns on each side, advance your pawns (one square), or capture opponent pawns, in order to be the first cross the board with a pawn to the other side.	https://en.wikipedia.org/wiki/Breakthrough_(board_game)	YES	
29	Three Men's Morris	2	blocks /grid	Tic-Tac-Toe, but each player only has 3 pieces. One all pieces have been place, move your piece onto adjacent clear square to achieve three in a row before your opponent.	https://en.wikipedia.org/wiki/Three_Men%27s_Morris	YES	
30	Picaria	2	blocks /grid	Three men's morris, but after all pieces are placed, you can move your piece to any near by clear location.	https://en.wikipedia.org/wiki/Picaria	YES	
31	Nine Holes	2	blocks /grid	Three men's morris, but after all pieces are placed, you can move your piece to any clear location. Diagonal three in a row's do not count.	https://en.wikipedia.org/wiki/Nine_Holes	YES	
32	Simplified Risk	2+	blocks /grid	Given units dispersed over sections on a map, and more than 1 unit on a specific section, capture adjacent section occupied by opponent units by rolling higher dice rolls, until all sections are captured.	https://en.wikipedia.org/wiki/Risk_(game)	YES	
33	President	2+	cards	Card shedding game, discard a card from your hand if its value it greater or equal to the top card on the discard pile. Play a two to clear the discard deck and start again. If doubles are played, you must play two cards of the same value that is great or equal to the top card. Win by discarding all cards before your opponent.	https://en.wikipedia.org/wiki/President_(card_game)	YES	
34	Crazy Eights	2+	cards	Card shedding game, on your turn discard a card from your hand if it has the same suit or value as the top card on the discard pile. Play an eight at any time as a wild card, and choose the suit. If you cannot play a card draw a card until you can, or have drawn three cards. Win by discarding all cards before your opponent.	https://en.wikipedia.org/wiki/Crazy_Eights	YES	

35	N-Rooks	1	blocks /grid	Place N Rooks on an NxN grid such that none are attacking each other.	http://mathworld.wolfram.com/RooksProblem.html	YES	
36	N-Kings	1	blocks /grid	Place N kings on an NxN grid such that none are attacking each other.	http://mathworld.wolfram.com/KingsProblem.html	YES	
37	N-Knights	1	blocks /grid	Place N knights on an NxN grid such that none are attacking each other.	http://mathworld.wolfram.com/KnightsProblem.html	YES	
38	Bishop swap	1	blocks /grid	Move bishops according to chess rules so that the black pieces end up where the white pieces were and vice versa and at no point can opposing colors be attackable by each other	http://www.chessvariants.com/solitaire.dir/bishops.html	YES	
39	Knight swap	1	blocks /grid	Bishop swap but with knights	https://en.wikipedia.org/wiki/Mathematical_chess_problem	YES	
40	Knight corner swap	1	blocks /grid	Knight swap but with only 4 knights in the corners, and no restrictions on attacking	http://mathematicscentre.com/taskcentre/taskcent.htm#knightswap	YES	
41	N-Queens	1	blocks /grid	Place N Queens on an NxN grid such that none are attacking each other.	https://en.wikipedia.org/wiki/Eight_queens_puzzle	YES	
42	Golf solitaire	1	cards	Move a uncovered card to the discard if it has a value of one less or one more than the top discarded card, you can also draw cards from the deck onto the discard	https://en.wikipedia.org/wiki/Golf_(patience)	YES	
43	Tri Peaks solitaire	1	cards	Similar to golf solitaire but the cards are in 3 pyramids rather than a grid	https://en.wikipedia.org/wiki/Tri_Peaks_(game)	YES	
44	Pyramid Solitaire	1	cards	Cards arranged in a pyramid with each card lower in the pyramid covering two in the row above it. Remove two clear cards if their sum is 13. A card from the deck can be used for one of the cards.	https://en.wikipedia.org/wiki/Pyramid_(solitaire)	YES	

45	Shuffle	1	grid/ mark	Given a 3x3 grid filled partially with numbers, and answer grids for each column and row that are partially filled, fill each empty grid with a number, such the answer grids contain the sum of the values of their column or row.	http://www.mnneske.no/shuffle/eng/	YES	
46	Kakuro	1	grid/ mark	MxN shuffle, no row col duplicates	http://www.mnneske.no/kakuro/eng/ https://en.wikipedia.org/wiki/Kakuro	YES	
47	Sujiko	1	grid/ mark	A small version of sudoku where the groups of 4 adjacent squares must also sum to a specific number	https://en.wikipedia.org/wiki/Sujiko	YES	
48	Suko	1	grid/ mark	Sujiko but with additional extra constraints	http://sujiko.co.uk/puzzles.html	YES	
49	Survo	1	grid/ mark	Larger version of kakuro, but any numbers not just 1-9 can be used	https://en.wikipedia.org/wiki/Survo_puzzle	YES	
50	Traveling salesman in Solid grid	1	mark/ grid	Move a piece in a grid such that it visit every spot once	http://cs.smith.edu/~jorourke/TOPP/P54.html	YES	
51	Age question problem	1	logic/ mark	Age of blue is two more than age of green and …	http://mathforum.org/dr.math/faq/faq.age.problems.html	YES	
52	Blackjack	2+	cards	Starting with two card, take action hit to gain a new card or stay to end. The goal is have the sum of your cards be 21 or as close to that without going over. Aces are 1 point or 11, and face cards are all 10 points.	https://en.wikipedia.org/wiki/Blackjack	NO	*worked at one time, need to fixup language
53	Hearts	4	cards	Trick taking game, play a card of the same suit as the first played if you can, otherwise a card of any played suit. Take the cards if you play the highest card of the played suit. Gain points for every heart, and 13 for the queen of spades each round, unless you take every point, in which case your opponents gain	https://en.wikipedia.org/wiki/Hearts	NO	Minimize points taken over many actions?

				28. Win by having the least points once someone reaches 100.			
54	War	2+	cards	Play the top card of your deck, if it is greater than your opponents you win the cards. Ties are settled by playing two additional cards, only the value of the second card counts. The first player to possess all cards wins.	https://en.wikipedia.org/wiki/War_(card_game)	NO	Conditional actions (when tied)
55	Memory	2+	cards	From a grid of face-down cards, select two cards to flip over. If they match you may add them to your pile. If not they get flipped back over. The player with the most matched cards at the end wins.	https://en.wikipedia.org/wiki/Concentration_(game)	NO	Partially obsrv. & need memory to play well/subgoals
56	Hexapawn	2	blocks/grid	On a 3x3 chess board advance pawns forward by one, or capture diagonally. Win by reaching the other side of the board, capturing all opponent pieces, or prevent the opponent from taking an action	https://en.wikipedia.org/wiki/Hexapawn	NO	Winning condition to win by inaction
57	Water Jug	1	logic	Using a 5 gallon jug and 3 gallon jug and only filling them or pouring from one to another, get exactly 1 gallon in the 3 gallon jug	https://en.wikipedia.org/wiki/Water_pouring_puzzle	NO	No action models for liquid pouring
58	Hangman	1+	word game	Given the length of a word, guess the word by guessing letters, which are exposed in the answer word, in a minimum number of guesses.	https://en.wikipedia.org/wiki/Hangman_(game)	NO	No dictonary, or handling or ordered sets
59	Chain reaction	1+	word game	The first and last word of a list must be connected by filling the intermediate spaces with words such that they form two-word phrases with both the proceeding word and following word, such as STOP-SIGN-LANGUAGE	https://en.wikipedia.org/wiki/Chain_Reaction_(game_show)	NO	Needs some sort of language database
60	Lingo	1+	word game	Given an initial letter of a five letter word, guess the word. Letters that match, or are out of place, are marked. Guess correctly with *X* guesses	https://en.wikipedia.org/wiki/Lingo_(U.S._game_show)	NO	No dictionary
61	Word Search	1	word game	Find all the words in the bank in a grid of letters. The words can be spelled horizontally, diagonally, vertically, and backwards.	https://en.wikipedia.org/wiki/Word_search	NO	No handling or ordered sets

62	Bananagrams	2+	word game	Given a set of tiles each with a letter, spell words and connect them horizontally and vertically until you have used all your tiles.	https://en.wikipedia.org/wiki/Bananagrams	NO	No dictionary
63	Scattergories	2+	word game	Roll a die to get a letter, for each category on a list (ie. make of car, type of fruit), write a word that is part of that category and starts with the letter	https://en.wikipedia.org/wiki/Scattergories	NO	Needs category database
64	Nine men's morris	2	blocks /grid	Place stones on empty locations until all stones have been place. If you make a 3 in a row, remove an opponent's piece from the board. Once all pieces have been place, available actions are move them to adjacent empty locations. Once you have only 3 stones you can move to any empty location	https://en.wikipedia.org/wiki/Nine_Men%27s_Morris	NO	extra action from achieving subgoal (removing a piece when 3 in a row)
65	Triangles	2	dots /grid	Given a space covered with some number of dots, take a turn by drawing a line between two unconnected points, if that line does not cross through any other line or dot. If this completes a triangle with other lines, and the triangle does not contain a point, mark this triangle as yours. Win by having more triangles than your opponent when no actions are possible.	https://cardgames.io/triangles/#rules	NO	drawing lines
66	Dots and boxes	2	dots/ grid	Given a grid of unconnected dots, take turns by drawing lines between two adjacent unconnected points. If this completes a square with other lines, mark this square as yours and take another turn. Win by having more squares than your opponent when no actions are possible.	https://en.wikipedia.org/wiki/Dots_and_Boxes	NO	drawing lines

Table 5: An initial list of many different games and puzzle, that includes a quick description, an external link with more details, whether we can learn it, and a quick explanation of why, if we cannot learn it.

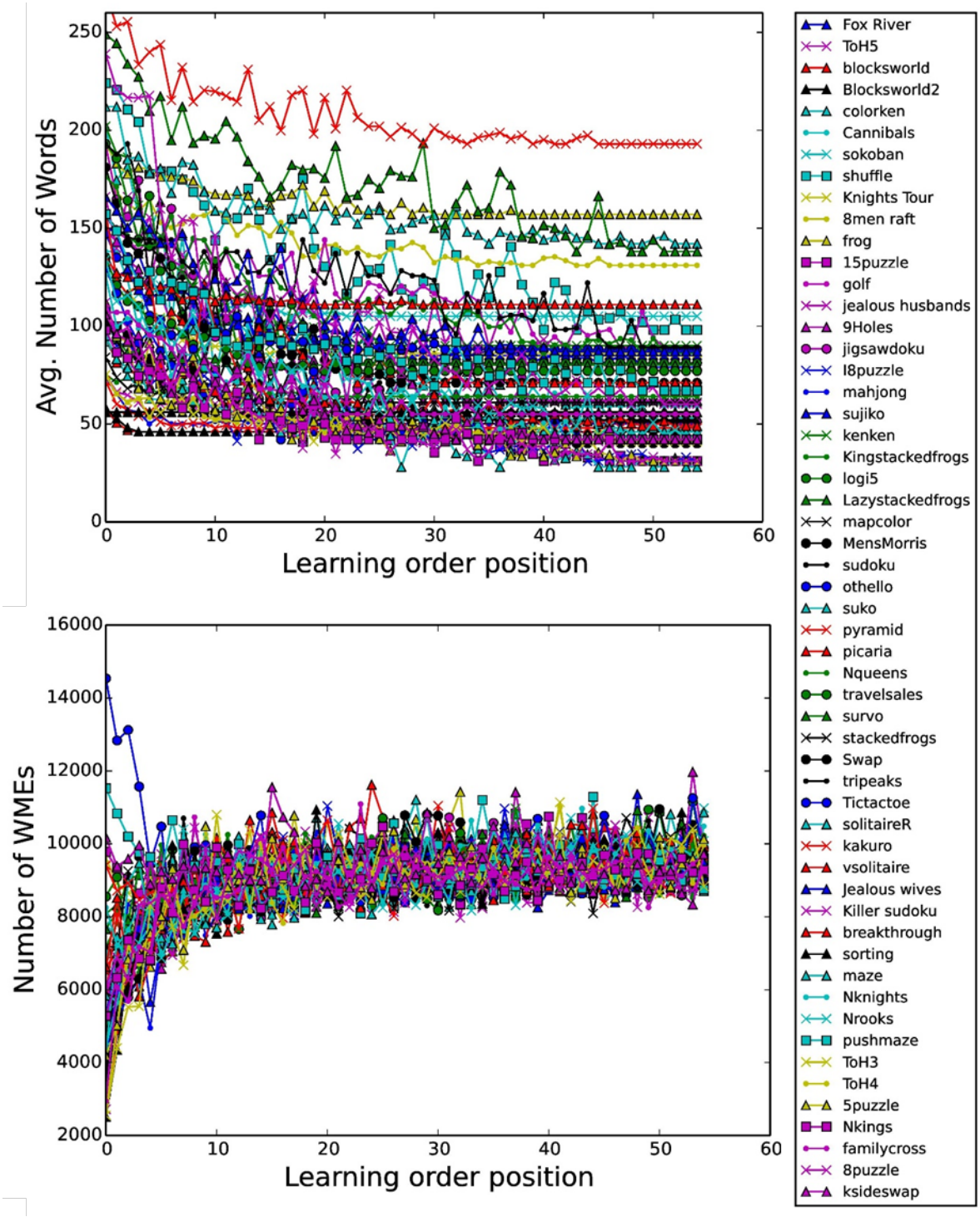


Figure 36: Experimental results learning 1000 permutations of 55 games. More permutations are required for good averages.

REFERENCES

Allen, J., Chambers, N., Ferguson, G., Galescu, L., Jung, H., Swift, M., & Taysom, W. (2007). Plow: A collaborative task learning agent. In *AAAI* (Vol. 7, pp. 1514-1519).

Azaria, A., Krishnamurthy, J., & Mitchell, T. M. (2016). Instructable intelligent personal agent. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Banerjee, B., & Stone, P. (2007). General game learning using knowledge transfer. Proceedings of the 20th International Joint Conference on Artifical intelligence (pp. 672–677).

Barbu, A., Narayanaswamy, S., & Siskind, J. M. (2010). Learning physically-instantiated game play through visual observation. Robotics and Automation (ICRA), 2010 IEEE International Conference (pp. 1879–1886).

Bhargava, D., Vega, G., & Sheffer, B. (2016). Grounded Learning of Color Semantics with Autoencoders.

Blythe, J. (2005). Task learning by instruction in tailor. In *Proceedings of the 10th international conference on Intelligent user interfaces* (pp. 191-198). ACM.

Cantrell, R., Talamadupula, K., Schermerhorn, P., Benton, J., Kambhampati, S., & Scheutz, M. (2012). Tell Me When and Why to do it! Run-time Planner Model Updates via Natural Language Instruction. Proceedings of the Seventh International Conference on Human-Robot Interaction.

Chai, J. Y., Fang, R., Liu, C., & She, L. (2016). Collaborative Language Grounding Toward Situated Human-Robot Dialogue. *AI Magazine*, *37*(4).

Chauhan, A., & Lopes, L. S. (2011). Using spoken words to guide open-ended category formation. *Cognitive processing*, *12*(4), 341.

Clark, H. H., & Brennan, S. E. (1991). Grounding in communication. *Perspectives on socially shared cognition*, *13*(1991), 127-149.

Derbinsky, N. L. (2012). Effective and Efficient Memory for Generally Intelligent Agents.

Dindo, H., & Zambuto, D. (2010). A probabilistic approach to learning a visually grounded language model through human-robot interaction. In *Intelligent Robots and Systems (IROS)* (pp. 790-796). IEEE.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. Cognition, 28(1-2), 3-71.

Genesereth, M., & Love, N. (2005). General game playing: Game description language specification (Technical Report). Computer Science Department, Stanford University, Stanford.

Gold, K., Doniec, M., Crick, C., & Scassellati, B. (2009). Robotic vocabulary building using extension inference and implicit contrast. *Artificial Intelligence*, *173*(1), 145-166.

Goldwasser, D., & Roth, D. (2014). Learning from natural instructions. *Machine learning*, *94*(2), 205-232.

Gluck, K. and Laird, John E. (2019). Interactive Task Learning. *MIT Press*.

Hinrichs, T. R., & Forbus, K. D. (2014). X goes first: Teaching simple games through multimodal interaction. Advances in Cognitive Systems, 3, 31–46.

Huffman, S. B., and Laird. J. E. (1995). Flexibly Instructable Agents. Journal of Artificial Intelligence Research, 3, 271-324.

Kaiser, L. (2012). Learning games from videos guided by descriptive complexity. Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence.

Kansky, K., Silver, T., Mély, D.A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S. and George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 1809-1818).

Kirk, J. R. and Laird, J. E. (2019) Learning Hierarchical Symbolic Representations to Support Interactive Task Learning and Knowledge Transfer. Proceedings of the 28th International Joint Conference on Artificial Intelligence.

Kirk, J. R. & Laird, J. E. (2016) Learning General and Efficient Representations of Novel Games Through Interactive Instruction. In Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems, Evanston, Illinois.

Kirk, J. R., Mininger, A., & Laird, J. E. (2016, July). A Demonstration of Interactive Task Learning. In *IJCAI* (pp. 4248-4249).

Kirk, J. R. and Laird, J. E. (2014). "Interactive Task Learning for Simple Games." Advances in Cognitive Systems, vol 3, pp. 13-30.

Kolve, E., Mottaghi, R., Gordon, D., Zhu, Y., Gupta, A., & Farhadi, A. (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv preprint arXiv:1712.05474*.

Laird, J. E. (2012). The Soar Cognitive Architecture. MIT Press.

Laird, J. E., Gluck, K., Anderson, J., Forbus, K., Jenkins, O., Lebiere, C., Salvucci, D., Scheutz, M., Thomaz, A., Trafton, G., Wray, R. E., Mohan, S., Kirk, J. R. (2017). Interactive Task Learning, IEEE Intelligent Systems, 32(4), 6-21, (invited).

Langley, P., Trivedi, N., & Banister, M. (2010). A command language for taskable virtual agents. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. Cognitive Systems Research, 10, 141–160.

Lindes, P., Mininger, A., Kirk, J. R., and Laird, J. E. (2017). Grounding Language for Interactive Task Learning. Proceedings of the 1st Workshop on Language Grounding for Robotics at ACL.

Love, N., Hinrichs, T., Haley, D., Schkufza, E., & Genesereth, M. (2008). General game playing: Game description language specification. Technical report, Stanford University.

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., & Fox, D. (2012). A joint model of language and perception for grounded attribute learning. *ArXiv:1206.6423*.

Mininger, A., and Laird, J. E. (2018). Interactively Learning a Blend of Goal-Based and Procedural Tasks. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI Press.

Mohan, S. (2015). From Verbs to Tasks: An Integrated Account of Learning Tasks from Situated Interactive Instruction.

Mohan, S., Kirk, J., Mininger, A., Laird, J. (2015). Agent Requirements for Effective and

Efficient Task-Oriented Dialog. AAAI Fall Symposium Series, North America.

Mohan, S., Mininger, A. H., Kirk, J. R., & Laird, J. E. (2012). Acquiring grounded representations of words with situated interactive instruction. In Advances in Cognitive Systems.

Mooney, R. J. (2008). Learning to Connect Language and Perception. In *AAAI* (pp. 1598-1601).

Newell, A. (1980). Reasoning, problem solving and decision processes: The problem space as a fundamental category.

Orhan, G., Olgunsoylu, S., Sahin, E., & Kalkan, S. (2013, August). Co-learning nouns and adjectives. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on* (pp. 1-6). IEEE.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y. (2009). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

Ramaraj, P., & Laird, J. E. (2018). Establishing Common Ground for Learning Robots*. RSS 2018: Workshop on Models and Representations for Natural Human-Robot Communication.* Pittsburgh, PA.

Roy, D. (2002). Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language*.

Salvucci, D. D. (2013). Integration and reuse in cognitive skill acquisition. *Cognitive Science*, *37*(5), 829-860.

Simon, H. A., & Hayes, J. R. (1976). The understanding process: Problem isomorphs. Cognitive Psychology, 8, 165–190.

Socher, R., Ganjoo, M., Manning, C. D., & Ng, A. (2013). Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems* (pp. 935-943).

Thomason, J. (2016). Continuously Improving Natural Language Understanding for Robotic Systems through Semantic Parsing, Dialog, and Multi-modal Perception.

Thomason, J., Zhang, S., Mooney, R., & Stone, P. (2015). Learning to interpret natural language commands through human-robot dialog. Proceedings of the 24th International Joint Conference on Artificial Intelligence.

Thomaz, A. L., & Breazeal, C. (2008). Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, *172*(6-7), 716-737.

Yürüten, O., Şahin, E., & Kalkan, S. (2013). The learning of adjectives and nouns from affordance and appearance features. *Adaptive Behavior*, *21*(6), 437-451.