Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

Learning Hierarchical Symbolic Representations to Support Interactive Task Learning and Knowledge Transfer

James R. Kirk* and **John E. Laird**
Computer Science and Engineering, University of Michigan
{jrkirk, laird}@umich.edu

Abstract

Interactive Task Learning (ITL) focuses on learning the definition of tasks through online natural language instruction in real time. Learning the correct grounded meaning of the instructions is difficult due to ambiguous words, lack of common ground, and the presence of distractors in the environment and the agent's knowledge. We present a learning strategy embodied in an ITL agent that interactively learns in one shot the meaning of task concepts for 40 games and puzzles in ambiguous scenarios. Our approach learns hierarchical symbolic representations of task knowledge rather than learning a mapping directly from perceptual representations. These representations enable the agent to transfer and compose knowledge, analyze and debug multiple interpretations, and communicate efficiently with the teacher to resolve ambiguity. We evaluate the efficiency of the learning by examining the number of words required to teach tasks across cases of no transfer, positive transfer, and interference from prior tasks. 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.

1 Introduction

How can an agent not only learn a single task, but quickly learn and pursue many different tasks without any prior task-specific knowledge? Approaches such as those used in AlphaZero [Silver *et al.*, 2018] learn a specific policy from massive training for only a single task – one implicitly defined by the agent's environmental interface and reward function. In contrast, we are interested in approaches that allow an agent to quickly learn many different tasks.

Interactive Task Learning (ITL) [Laird *et al.*, 2017] is such an approach, where an agent learns new tasks from scratch, each in one shot, via natural language interactions with a human instructor in a shared environment. Using the acquired task definition together with its planning, reasoning, and learning capabilities, the agent can immediately pursue the task [Mininger and Laird, 2016]. As additional tasks are learned, the agent *transfers* knowledge from previous tasks to new tasks, speeding task learning.

Many task learning systems assume that task descriptions use either a fixed set of words, or that new words and task elements (terms, goals, actions, ...) can be directly mapped (one-to-one) to known primitives or subsymbolic representations in a single domain [Chai *et al.*, 2018; Goldwasser and Roth, 2014; Azaria *et al.*, 2016]. However, as an agent learns many tasks, there will inevitably be many-to-many mappings between words and the components of a task, and in some cases knowledge learned in a previous task can interfere with a new task.

Think of a child learning Tic-Tac-Toe by marking X's and O's on paper. They should also be able to learn a new version of Tic-Tac-Toe that involves placing colored stones on a board, without having to relearn the rules of the game (within-task transfer). Moreover, the concept of "three in a row" from Tic-Tac-Toe should transfer to similar games, such as Three Men's Morris, without additional instruction (across-task transfer). However, in a game where stones are not markers but movable pieces, such as Breakthrough or Checkers, blindly trying to use the previously learned knowledge will lead to confusion and potentially an inability to solve the problem. However, with help from an instructor, the child can refine and specialize the task-specific meaning of the stones for the new game (avoiding negative transfer).

We focus on goal-oriented tasks, which Newell [1993] proposed can be formulated as problem spaces. In this formulation, a task (such as a marking version of Tic-Tac-Toe) can be decomposed into an initial state (a blank 3x3 grid), goal states (three in a row of the player's marks), legal actions and their preconditions (marking a blank square), and failure conditions (the opponent achieving three in a row). This formulation can be extended to also include terms with task-specific meanings (a square is *occupied* by a mark) that are used in defining these states, conditions, and actions. In this formulation, task learning corresponds to learning to recognize and apply these *task elements*: goals (learning to detect three in a row when achieved), legal actions (learning how to legally take an action by marking a square), task terms (learning to detect if an square is *occupied*), and so on.

In previous work [Kirk and Laird, 2016], we adapted this

*Contact Author

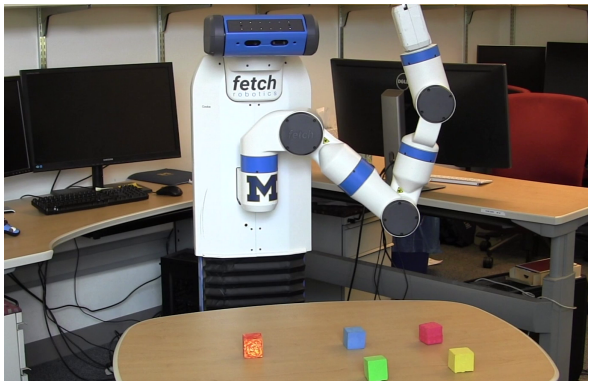

Figure 1: Rosie embodied on a Fetch robot learning a new puzzle.

formulation to enable the learning of hierarchical compositions of task element in the ITL agent Rosie [Mohan *et al.*, 2012]. Rosie, built on the Soar cognitive architecture [Laird, 2012], learns goal-oriented and procedural tasks for physical robotic platforms (Magicbot, Fetch, and a tabletop robot) and simulated domains, through instruction and demonstration. Figure 1 shows Rosie embodied on the Fetch robot learning a version of Five-Puzzle with blocks. Rosie is an end-to-end ITL agent that learns from basic primitives, with interfaces to those environments (vision systems, manipulator controllers, speech and text I/O, ...). It has a custom natural language parser for the interpretation of a restricted subset of English, and planning and reasoning capabilities for solving tasks once their definitions are learned, all implemented within Soar [Mininger and Laird, 2018].

We extend that work by enabling 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 efficiently 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 hierarchical task elements and the breadth of terms and games that the agent can learn.

Below we present our approach, which correctly learns 40 common games and puzzles. These tasks provide a large and varied set of problems, with goals and actions that use many different types of concepts and capabilities. We empirically evaluate our approach's ability to transfer knowledge across tasks by teaching all of these tasks in random sequences. We then 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.

2 Related Work

Previous agents have demonstrated learning of new concepts for tasks, including agents that learn games from observation [Kaiser, 2012; Barbu *et al.*, 2010] and ones that learn from language [Cantrell *et al.*, 2012; Thomason *et al.*, 2015; Chai *et al.*, 2018; Scheutz *et al.*, 2018]. These agents, most of which learn policy knowledge, are limited in their ability to learn new concepts, transfer knowledge to new tasks, and interact naturally. Many agents [Hinrichs and Forbus, 2014] learn intermediate representations to be interpreted, such as GDL [Love *et al.*, 2008], rather than native representations and do not have a theory for task knowledge transfer.

Research on learning the groundings of words in situated domains has focused on learning new symbols grounded directly in a robot's subsymbolic sensorimotor representations. Agents that do learn groundings from existing symbolic concepts only learn synonyms; they assume that for any used term there is a single matching concept with the identical meaning [Goldwasser and Roth, 2014]. Approaches that require large numbers of training examples, using machine learning, have been successful at learning new adjective, prepositions, and nouns.[Roy, 2002; Bhargava *et al.*, 2017; Chauhan and Lopes, 2011; Dindo and Zambuto, 2010; Orhan *et al.*, 2013; Matuszek *et al.*, 2012]

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 may be an offline batch process. With Rosie, 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.

3 Task Element Learning

In this section, we define the process by which Rosie learns to recognize and apply task elements: actions, goals, failure conditions, and words with task-specific meaning, such as "occupied." A task element can be thought of having a *condition,* which is a description of the context in which the task element is appropriate to apply. Therefore, learning a task element involves learning to associate that condition with the appropriate act: 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 exists.

To enable learning (and grounding) a task element, Rosie asks the user to create 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. Rosie's perceptual systems extract objects and primitive features and relations, including

Types	Primitives
Object types	location, block
Colors*	red, green, purple, yellow, orange, black, white, blue, brown
Labels	garbage, destination, card, bank, pawn, king, knight, rook, queen
Shapes*	cube, sphere, cylinder, rectangle
Sizes*	tiny, small, medium, large
Comparitors	less than, more than, equal to, x–less/more than, same
Functions	subset, product, numeric between, attribute of, count, sum
Spatial relations*	on, below, near, adjacent, diagonal, left, right, above, under, between

Table 1: Primitive knowledge encoded in the agent. Previous implementations have learned classifiers for the "*" concepts.

colors (red, $green$), sizes ($large$, $small$), spatial relations ($next\ to$, $below$), object types ($location$, $block$), and labels ($bank$, $destination$) in order to create an internal representation of the state. Rosie also knows a set of primitive functions ($attribute\text{-}of$, sum). Rosie's primitive knowledge is shown in Table 1, organized by type, including those that are used to learn the games taught in the evaluation.

Figure 2 shows the external state (left side) and derived internal symbolic representation (right side) for an instance of Tic-Tac-Toe where red and blue blocks are used as pieces. The state can be used to teach the goal for the red player. The objects and locations in the figure are indexed with numbers 1-16, so that the red block 12 is on location 7.

In our approach, determining whether a task element can be recognized and applied requires an internal relational symbolic representation of the agent's perception of the world in order to ground learning in the current context. The agent asks for a description of the conditions of the task element, and converts the natural language into a relational representation. The agent learns to recognize the task element by recursively learning all the supporting terms (other task elements) needed to ground the structure to the world state. Below, we explain how the task elements are applied, and how ambiguous scenarios and knowledge interference are handled through the creation of multiple versions of recognition structures for each possible interpretation.

Objects 1-9: object type=location
Objects 10-13: color=red, object type=block
Objects 14-16: color=blue, object type=block
On : (10,1); (11,4); (12,7); (13,9); (14,6); (15,5); (16,2)
Linear : (1,2,3); (1,4,7); (2,5,8); (3,6,9); (1,5,9); (3,5,7)

Figure 2: Internal state representations created for Tic-Tac-Toe.

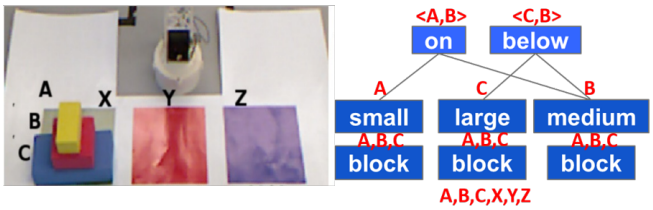

Figure 3: External and internal representation of Tower of Hanoi. On the right is a graphical representation of the internal relational structure Rosie created for the goal.

3.1 Recognition Structure Learning

As mentioned above, Rosie processes natural language using a custom parser to create an internal, relational representation of the task element. From that representation, Rosie constructs a conjunction of predicates that represents the conditions of the task element. For example, from the description "The goal is that a small block is on a medium block and a large block is below the medium block," 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 defined as the unification over n predicates $f_i()$ and m objects x_j as show in Equation 1.

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

The conjunction of predicates, $p(x, ..)$, is defined over a set of objects, x, which can be objects in the environment, strings, numeric values, or sets of x. Predicates represent binary values over unary features (red, $large$), n-ary relationships (on, $behind$), set operations ($subset$), and functions ($count$). Predicates in the conjunction can be negated ($\neg below$).

In this representation, set operations and functions $y = f(x...)$ are reified to the truth test $p(y, x..)$. For example, the predicate 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 function $count(y, x)$. 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 a tree structure to help support composition of task elements and partial knowledge transfer. 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.

An example of the tree structure created for a goal example is graphically depicted in Figure 3. The agent constructs the structure with the unary predicates at the bottom (block) and the binary predicates at the top. Once that structure is built, Rosie links the learned condition structure to the linguistic term. The red letters A-Z (on the right) are indexes for the objects, unique identifiers generated by the agent for objects in the environment (on the left). The structure is evaluated against the world, bottom up, with the results of each

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
```

predicate test labeled in red directly above. Logically, the constructed tree structures are equivalent to the conjunction of a set of predicates represented in Equation 1. To construct this predicate structure, Rosie adds the predicates to the structure in an order based on the predicate arity and dependency information extracted during parsing. First, unary predicates ($block$, $small$) are added, then any predicates created from dependent clauses (such as "the block that is on a location"), then binary predicates (on, $below$), and finally n-ary predicates (such as $between$). This ordering decreases the number of objects tested by each predicate.

The final step is to attempt to ground the structure to the agent's internal state, using interpretation knowledge about functions, primitive predicates, and learned predicates. 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. This allows Rosie to discover if there are any unknown or unsatisfied terms or if the structure cannot be satisfied given its current knowledge.

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.

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 are simple heuristics that leverage the tree structure and hierarchy, such as preferring lower predicates, predicates with satisfied predicates beneath them, and predicates lower in the hierarchy ($RGF[f_i()] > RGF[f()]$). This learning function terminates when it has learned to recognize and apply the task element $f(x)$ and all supporting task-element $f_i(x)$ used to describe it.

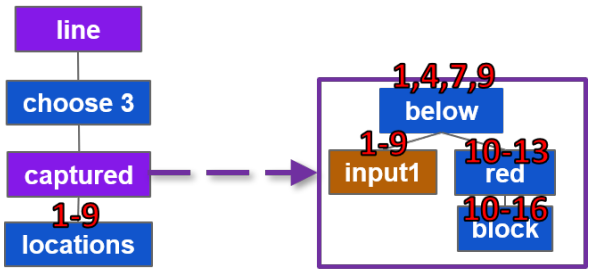

Figure 4: Learned recognition structures for Tic-Tac-Toe goal.

Rosie (its interactions shown in **bold**) learns the goal of Tic-Tac-Toe (Figure 2) from the teacher in the example below.

> The name of the goal is three-in-a-row.
> **Please setup the goal.**
> Ok.
> **Please describe the conditions of the goal.**
> The goal is that three of the captured locations are in a line.
> **Can you define 'captured?'**
> If a location is below a red block then it is captured.
> **Can you define 'in a line?'**
> If the locations are linear then they are in a line.

From this goal description (and state example) Rosie learns the new predicate conjunction $three\text{-}in\text{-}a\text{-}row(x_1, x_2) = locations(x_2) \wedge captured(x_2) \wedge choose\text{-}3(x_1, x_2) \wedge line(x_1)$. For primitive functions, such as the subset function $choose\text{-}X$, the agent innately knows how the function is referred to in the parsed relational representation. In this case, when Rosie sees "X of Y", where X is a number ($three$) and Y is a set of objects ($locations$), it creates the predicate $choose - X(Z, Y)$, where Z represents output of the function: the created subsets.

Figure 4 shows a representation of the recognition tree Rosie creates. The primitive terms known a priori to the agent are labeled in blue. Predicates for $captured$, $choose$-3, and $line$ are not satisfied, and $captured$ is lowest in the tree, so $RGF[captured(x)]$ is called first. The agent doesn't know $captured$, so it asks for a definition. In Figure 4, an arrow points from $captured$ to the structure created from the teacher's response: "If a location is below a red block then it is captured." Rosie satisfies this definition by matching locations 1, 4, 7, and 9 from the state in Figure 2.

Figure 5 shows that that $captured$ can be satisfied by locations $[1, 4, 7, 9]$. These feed into the primitive function

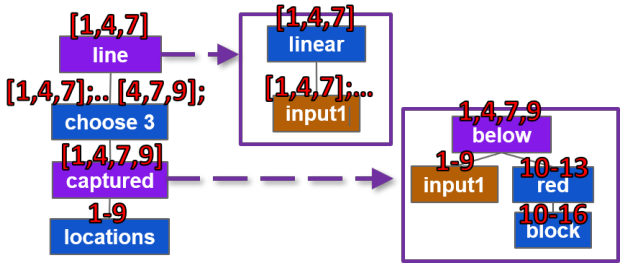

Figure 5: Learned recognition structures for Tic-Tac-Toe goal.

$choose$-3 which generates all subsets of size 3. However $line$ is still unsatisfied and so Rosie calls $RGF[line(x)]$. It does not have any definition, so Rosie asks the teacher for one: "Can you define in a line?" The learned structure for line allows the agent to detect the concept, and finally Rosie can satisfy entire goal $three\text{-}in\text{-}a\text{-}row(x_1, x_2)$ with $[1, 4, 7]$.

3.2 Task Element Application

The agent uses the learned task elements to attempt to perform the task. The details of task element application depend on the type of task element. An action is proposed when its conditions match the current state. For goals, the agent detects that it solved the puzzle or won the game when the conditions match, and for failure conditions, the agent detects that it has found a bad solution path or lost the game. Task-specific terms are evaluated as needed to determine if other tasks elements are satisfied. For example, if Rosie is attempting to determine if the goal of Tic-Tac-Toe is satisfied, it will eventually evaluate $captured$ in the current game state.

3.3 Creating Multiple Interpretations of Task Elements

During the task element learning process, many 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 and meanings across tasks, the agent can have multiple meanings for the same word.
- **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.

To ensure that it correctly interprets an ambiguous situation, Rosie generates *all* possible recognition structures, $f(x)$ (Equation 1), for each known meaning of the defining terms, $f_i(x)$. 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, 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 of this analysis Rosie, uses one of three different strategies to try 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 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.

4 Task Transfer Without Interference

To evaluate transfer when there is no ambiguity, we created instructions for 40 common games and puzzles so that no task elements were overloaded with multiple meanings, and so that there was maximum similarity in game descriptions and simulated environments. For example, the states for Tic-Tac-Toe and Three Men's Morris always contain red blocks for the agent and blue blocks for the opponent, and captured is defined as "below a red block" for both tasks. To study transfer, we taught the agent sequences of the 40 games in 1000 randomly generated permutations. In each permutation, each game is taught, one after another, using scripts that simulate a teacher. The scripts ensure that if a concept had been previously learned the "teacher" will not try to teach it again.

These 40 games, which include variants indicated by (total number) or (names), are Tower of Hanoi (3), N-Puzzle (4), Sudoku, Killer Sudoku, Jigsawdoku, KenKen (2), Logi-5, Map 4-Coloring, Chess puzzles (N-Queens, N-Kings, N-Rooks), Peg solitaires (2), Card solitaires (Golf, Pyramid, Tri Peaks), Fox River crossing puzzle, Jealous Husband (2), Traveling Salesman in a grid, 3x3 stone games (Tic-Tac-Toe, Three Men's Morris, Picaria, Nine Holes), Frogs and Toads, Stacking Frogs (3), Blocks World (2), Mazes (simple, block pushing), Mahjong puzzle, and a sorting puzzle. Teaching scripts and state representations for these games, as well as a video of Rosie learning, are available online.[1]

All 40 games are learned correctly in each permutation. Figure 6 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 39 all other games have been taught. As more games are taught, the number of words required to teach a game decreases if

[1] www.umich.edu/~jrkirk/ijcai2019.html

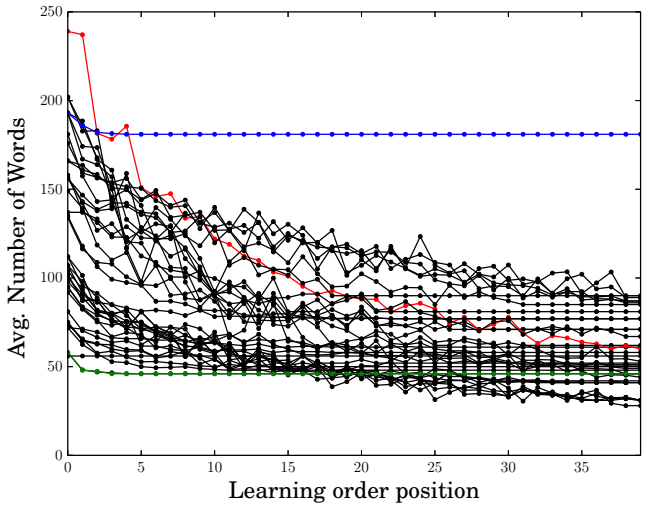

Figure 6: The number of words required to teach each of 40 games by teaching order. Results are averages of 1000 permutations.

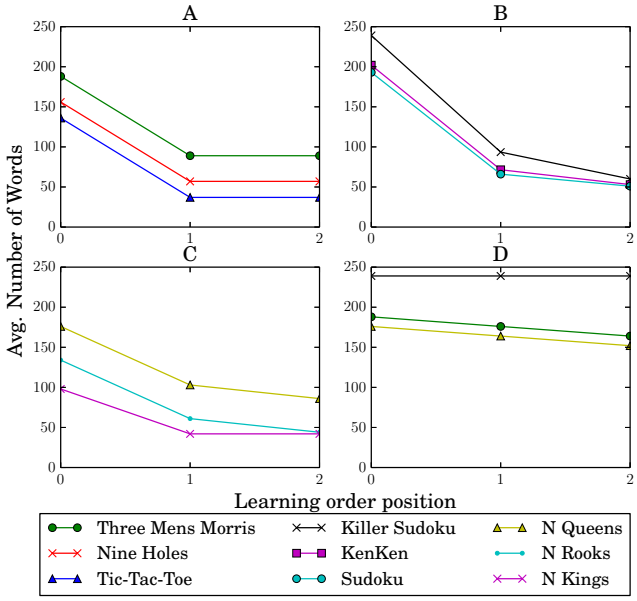

Figure 7: Number of words required to teach clusters of tasks A-D.

there is across-task transfer of the task elements shared between games. Games that have substantial conceptual overlap, such as Five-Puzzle and Eight-Puzzle, which share actions (slide) and learned predicates (clear, matched, adjacent), can be defined using very few words (only 31) at the end. The gradual decrease in the number of words is a reflection of the gradual increase in the probability that a related game is previously taught.

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 high 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.

This experiment was then repeated with small clusters of games to further analyze knowledge transfer. These task clusters contain tasks that have a large conceptual overlap: Tic-Tac-Toe, Three Men's Morris, Nine Holes; Killer Sudoku, KenKen, Sudoku; and N Queens, N Rooks, and N Kings. The final cluster contains tasks with little overlap, with a single task selected from each of the other clusters. Figure 7 shows the results, again showing the number of words required to teach the task based on the position in the teaching order. Plots A-C show the dramatic effects of transfer in clusters of similar tasks, while Plot D shows almost no transfer between the unrelated tasks. This result is expected, but other task learning approaches that learn directly from subsymbolic representation have failed to replicate this type of task transfer that leads to dramatic learning speed up.

5 Task Transfer with Interference

For the cases where there is interference, a scripted-based evaluation as in Section 3 is not possible because it is extremely difficult to predict a priori what questions the agent will ask in ambiguous situations. Instead, we use a series of case studies to illustrate the agent's ability to handle interference using the process described in Section 2.3: 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.

5.1 Case 1: Ambiguous Word Meaning

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 8 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 9, one for each interpretation of *clear*.

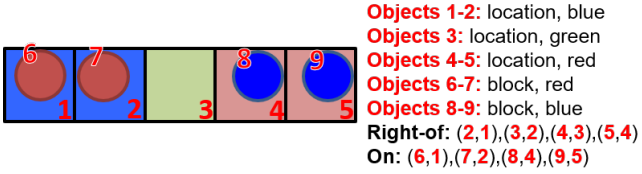

Figure 8: The internal state generated by the agent for the Frogs puzzle, with objects identified by red indexes. On the right are unary features and binary relations that the agent extracts from the state.

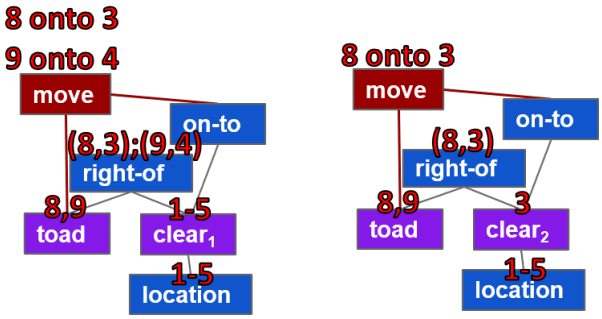

Figure 9: Recognition structures created for two interpretations of an action. Red values indicate the indexes of objects in the environment that results from grounding the structure to the external state.

> If a toad is to the right of a clear location then you can move the toad onto the location.
> **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 interpretation by asking a simple question. 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.

5.2 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 10 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.
> **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 disambuguating question. When the teacher responds indicating there is only one clear location, Rosie selects the correct interpretation structure to learn.

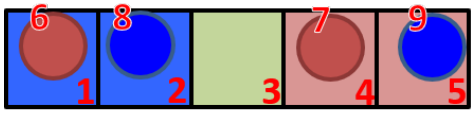

Figure 10: A representation of the internal state generated by the agent for describing jumping in the Frogs and Toads puzzle.

5.3 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, as described in Section 2.1, 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 as in Figure 2, 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).

6 Conclusion

The ability to create, analyze, and debug *multiple* hierarchical symbolic recognition structures extends Rosie so that it can learn to correctly recognize and apply the element of a tasks when the conditions of learning are ambiguous and many interpretation are possible. Furthermore it enables Rosie to efficiently communicate to resolve the ambiguity and select interpretations. Using this strategy, Rosie can learn a large number of tasks (40), and transfer knowledge between tasks even when there is knowledge interference. We plan to explore additional disambiguation strategies, such as asking questions directly about the learned representations, such as: "Does captured mean the it is below a red object?" Future plans include expanding the agent to learn heuristics and reward functions, and expanding to other types of tasks.

A current limitation is that there is no perceptual attention mechanisms, so that it is computational expensive to detect task elements for states with large number of objects and relations (such as in Chess). A second limitation is that the language Roise understands, although sufficient for these games, is more rigid than natural language.

Acknowledgments

This work was supported by the AFOSR under Grant Number FA9550-18-1-0168. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the AFOSR or the U.S. Government.

References

[Azaria *et al.*, 2016] Amos Azaria, Jayant Krishnamurthy, and Tom M. Mitchell. Instructable intelligent personal agent. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI-16, pages 2681–2689. AAAI Press, 2016.

[Barbu *et al.*, 2010] Andrei Barbu, Siddharth Narayanaswamy, and Jeffrey Mark Siskind. Learning physically-instantiated game play through visual observation. In *2010 IEEE International Conference on Robotics and Automation*, pages 1879–1886. IEEE, 2010.

[Bhargava *et al.*, 2017] Dev Bhargava, Gabriel Vega, and Blue Sheffer. Grounded learning of color semantics with autoencoders. *Online source*, 2017.

[Cantrell *et al.*, 2012] Rehj Cantrell, J. Benton, Kartik Talamadupula, Subbarao Kambhampati, Paul Schermerhorn, and Matthias Scheutz. Tell me when and why to do it! run-time planner model updates via natural language instruction. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 471–478. IEEE, 2012.

[Chai *et al.*, 2018] Joyce Y. Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. Language to action: Towards interactive task learning with physical agents. In *IJCAI*, pages 2–9, 2018.

[Chauhan and Lopes, 2011] Aneesh Chauhan and Luís Seabra Lopes. Using spoken words to guide open-ended category formation. *Cognitive processing*, 12(4):341, 2011.

[Dindo and Zambuto, 2010] Haris Dindo and Daniele Zambuto. A probabilistic approach to learning a visually grounded language model through human-robot interaction. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 790–796. IEEE, 2010.

[Goldwasser and Roth, 2014] Dan Goldwasser and Dan Roth. Learning from natural instructions. *Machine learning*, 94(2):205–232, 2014.

[Hinrichs and Forbus, 2014] Thomas R. Hinrichs and Kenneth D. Forbus. X goes first: Teaching simple games through multimodal interaction. *Advances in Cognitive Systems*, 3:31–46, 2014.

[Kaiser, 2012] Lukasz Kaiser. Learning games from videos guided by descriptive complexity. In *AAAI*, 2012.

[Kirk and Laird, 2016] James R. Kirk and John E. Laird. Learning general and efficient representations of novel games through interactive instruction. *Advances in Cognitive Systems*, 4, 2016.

[Laird *et al.*, 2017] John Laird, Kevin Gluck, John Anderson, Kenneth Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, Robert Wray, Shiwali Mohan, and James Kirk. Interactive task learning. *IEEE Intelligent Systems*, 32(4):6–21, 2017.

[Laird, 2012] John E. Laird. *The Soar cognitive architecture*. 2012.

[Love *et al.*, 2008] Nathaniel Love, Timothy Hinrichs, David Haley, Eric Schkufza, and Michael Genesereth. General game playing: Game description language specification. Technical report, 2008.

[Matuszek *et al.*, 2012] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. *arXiv preprint arXiv:1206.6423*, 2012.

[Mininger and Laird, 2016] Aaron Mininger and John E. Laird. Interactively learning strategies for handling references to unseen or unknown objects. *Advances in Cognitive Systems*, 5, 2016.

[Mininger and Laird, 2018] Aaron Mininger and John E. Laird. Interactively learning a blend of goal-based and procedural tasks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Mohan *et al.*, 2012] Shiwali Mohan, Aaron H. Mininger, James R. Kirk, and John E. Laird. Acquiring grounded representations of words with situated interactive instruction. *Advances in Cognitive Systems*, 2:113–130, 2012.

[Newell, 1993] Allen Newell. *Reasoning, problem solving, and decision processes: The problem space as a fundamental category*. MIT Press, 1993.

[Orhan *et al.*, 2013] Güner Orhan, Sertaç Olgunsoylu, Erol Şahin, and Sinan Kalkan. Co-learning nouns and adjectives. In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6. IEEE, 2013.

[Roy, 2002] Deb K. Roy. Learning visually grounded words and syntax for a scene description task. *Computer speech & language*, 16(3-4):353–385, 2002.

[Scheutz *et al.*, 2018] Matthias Scheutz, Evan Krause, Bradley Oosterveld, Tyler Frasca, and Robert Platt. Recursive spoken instruction-based one-shot object and action learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5354–5358. International Joint Conferences on Artificial Intelligence Organization, 2018.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[Thomason *et al.*, 2015] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1923–1929. AAAI Press, 2015.