Advances in Cognitive Systems 4 (2016) Submitted 4/2016; published 6/2016

Learning General and Efficient Representations of Novel Games Through Interactive Instruction

James R. Kirk JRKIRK@UMICH.EDU
John E. Laird LAIRD@UMICH.EDU
Computer Science and Engineering, The University of Michigan, Ann Arbor, MI 48105 USA

Abstract

The goal of our research is to develop agents that can learn new tasks through real-time natural interactions with a human. Previously, we have described Rosie, an agent implemented in Soar that interactively learns new tasks. In this paper, we describe novel extensions to Rosie that allow it to learn complex, hierarchically defined concepts and to dynamically compile the interpretation of the task instructions. We evaluate the generality and efficiency of the knowledge that the agent learns over seventeen simple games and puzzles embedded in a variety of simulated and real world robotic environments. Our results show that learning hierarchical concepts allows Rosie to transfer learned knowledge to new tasks and decrease future instruction, while dynamic compilation decreases the time to process instructions, execute newly learned tasks, and learn future tasks.

1. Introduction

In Interactive Task Learning (Laird, 2014), agents learn tasks through real-time natural interactions with a human instructor. Rosie, an agent implemented in Soar (Laird, 2012), learns new games and puzzles (Kirk & Laird, 2014) and procedural tasks (Mohan, 2015) in a variety of real-world and high-fidelity simulated environments. During learning, Rosie interacts with the human instructor using constrained natural language, asking the instructor to define task goals, failure states, and actions. If a novel word is used in a definition, Rosie requests that the word be defined. Rosie learns new features (nouns and adjectives), relations (prepositions), and composite actions (verbs). As new words and their associated groundings are learned, they are immediately available for use in interactions with Rosie (Mohan, Mininger, Kirk, & Laird, 2012). Although Rosie could learn a variety of tasks and concepts, there were significant limits in the complexity of concepts that could be learned. In addition, newly learned concepts could not be composed to learn new concepts.

In this paper, we describe extensions to Rosie that increase the complexity and generality of the concepts it can learn. Figure 1 illustrates how these extensions have changed the dialog between the instructor and Rosie when teaching an action from Tic-Tac-Toe. The major improvements are due to a more sophisticated parser, enabling natural anaphoric references and complex sentence constructions, and the ability to learn and compose new concepts, such as *free* or *clear*, which transfers from a previously taught task. The previous version was limited to defining actions and goals in terms of primitive concepts, limiting the transferability and complexity of describable concepts.

Previous Rosie:
move 1 on 2
What type of object is parameter 1?
block
Please list all the constraints for this parameter.
it is red
it is not on a location
finished
what type of object is parameter 2?
location
Please list all the constraints for this parameter.
it is not below a location
finished

Extended Version:
You can move a free red block onto a clear location.
I don't know the concept 'free'.
If a block is not on a location then it is free.
Ok, I've learned the concept free.
Does 'clear' mean that it is not below a block?
Yes.

Figure 1. Example dialog used to teach Rosie the action for Tic-Tac-Toe from both the old version (on the left) and the extended version (on the right). Rosie's responses are in italics.

Concepts can now be defined by complex compositions of previously learned concepts, and a learned concept can have multiple task or domain-specific definitions. For example, Rosie can learn a general concept that an opponent's piece is black, which is used in many different games. Rosie can then learn that in one game instance, red blocks are used for the opponent's pieces. Moreover, it can learn that in a different game, red blocks can be used for its own pieces. Thus, the mapping, or grounding, of terms to agent concepts can be many-to-many and context dependent. Furthermore, concepts can now be defined over sets of objects and be constrained by functions, such as in the utterance: *the number of clear blocks is three*, which defines a set of objects (those that satisfy both the conditions *clear* and *block*) and the desired result of a function (*number-of*) of that set.

A second shortcoming of Rosie was that it learned only declarative representations of the game concepts. Those declarative representations had to always be interpreted using a deliberate, sequential process whenever the game was played. The declarative representations provide the generality needed for processing different tasks, but are computationally expensive to interpret. In this paper, we describe how the new concept learning mechanism, in combination with improved representational structures, allow Soar's chunking mechanism to dynamically compile the deliberate interpretation into procedural rules.

Together, these extensions dramatically improve Rosie's generality and efficiency: (1) they expand the diversity and complexity of tasks to seventeen games; (2) they increase the transfer of knowledge between tasks as they are learned; (3) they decrease the processing time required to process instructions, learn new tasks, and execute those tasks; and (4) they allow Rosie to learn many tasks in succession without suffering from any substantial slowdown as knowledge is acquired.

In the remainder of the paper, we describe the new concept learning process and give examples of how it expands the types and complexity of concepts Rosie can learn. We then empirically evaluate it in terms of generality and efficiency across seventeen tasks. We discuss related work, and then conclude with general observations and ideas for future research.

2. Interactive Concept Learning

There are multiple types of task concepts that must be learned to specify a game: actions, goals, failure conditions, and potentially new predicates. All of these involve detecting the states in which the concept 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 predicate is true. Each task concept is defined by a linguistic term ("stack","three-in-a-row","clear"), a conjunction of predicate tests, and the usage knowledge specific to that concept type. For example, in teaching a simple 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 $clear(X) \wedge block(X) \wedge clear(Y) \wedge location(Y)$, and the usage knowledge for an operator has to do with it actions: 'move X onto Y'.

The conjunction of a set of predicates is defined over objects and string and number constants, created from the natural language descriptions, such as *there are eight matched locations*. Predicates, $p(x, ..)$, represent binary values over unary features (red, $large$...), n-ary relationships (on, $behind$...), and the results of functions, where $y = f(x..)$ is represented as truth test $p(y, x...)$. For example, the representation created for *the number of blocks is three* is $number\text{-}of(3, blocks)$. Learning new predicates enables the composition of existing and learned predicates to learn hierarchical concepts.

The concept learning process can be described by two major phases, structure learning and structure interpretation. The learning phase extracts the 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. Once that structure is built, Rosie *links* the learned structure to the linguistic term and its usage as an action, goal, failure, or new predicate.

The second phase 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 *number of*). Rosie tries to interpret the structure immediately after it is learned, both to verify that the learned structure is correct and to enable learning of procedural interpretation code that will be used for task performance. This phase could potentially be delayed until the agent attempts the task, but for now, the agent proactively interprets the structure immediately after learning it.

As a side of effect of the linking and interpretation processes, Soar's chunking mechanism dynamically creates rules that capture the input-output mappings of the processing, so that in the future, the deliberate, sequential processing is replaced by procedural rules.

In the following section, the structure learning and interpretation phases are discussed in detail, followed by a discussion of chunking and presentations of illustrative examples from different tasks.

2.1 Structure Learning Phase

In this section, we describe how Rosie performs linguistic analysis of a natural language sentence to create a semantic representation, extracts the described conjunction of predicates, and converts that structure into an organized declarative predicate structure.

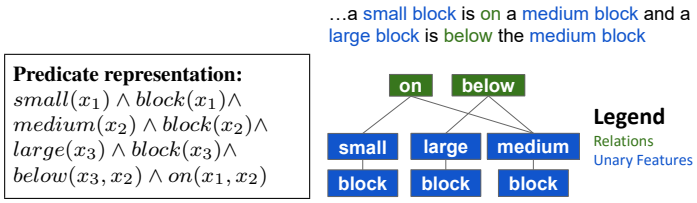


Figure 2. Representations of different intermediate formats Rosie learns from natural language: on the left predicate logic and on the right the learned declarative structure that orders the predicate tests.

2.1.1 Natural Language Processing

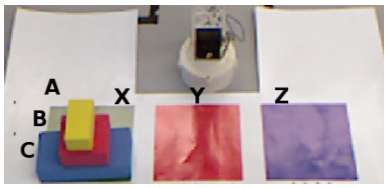
Rosie can parse a restricted form of natural language that is sufficient for natural descriptions of many games and puzzles. The details of its operation are not important for this paper; however some of its features are relevant. It can process many natural forms of anaphoric reference so that multiple references to the same object can be easily made in a sentence. One example is in a sentence we will use going forward: *The goal is that a small block is on a medium block and a large block is below the medium block.*

The parser produces a semantic description of the sentence, as well as a message type. The main message types used for teaching games are *is-a-clause*, *if-then-clause*, and *you-can-action*. Each of these message types can contain multiple subclauses, such as in the example goal sentence. Usually, the parser attempts to ground all object descriptions to entities in the world; however, when describing a game, the concepts are often abstract (such as "a medium block"), so the parser creates an internal representation of a hypothetical object which contains predicates for all the constraints that were present in its description. The conjunction of predicates extracted from the example sentence is shown on left side of Figure 2. The three objects are represented by variables (x_n), which are tested by unary predicates ($small$, $block$, $medium$, $large$) and binary predicates ($below$, on).

2.1.2 Declarative Predicate Structure Construction

From this conjunction of predicates, Rosie constructs a tree structure, organized around the objects in the predicates, as shown on the right side in Figure 2. The predicates are ordered so they can be efficiently evaluated, from bottom to top in the following interpretation phase. The leaf nodes are predicate tests that are evaluated against the agent's perception of the world, and any objects that satisfy that predicate will be passed up to the parent node during interpretation.

To construct this predicate structure, Rosie iterates through each predicate and adds it to the structure, keeping track of the last predicate added to the structure that tested the same object or value. Rosie adds the predicates in an order based on the predicate arity and dependency information extracted during parsing. 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 $between$).



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

When defining a concept for a new predicate, rather than an action, goal, or failure condition, the sentence is always structured as an *if-then-clause*: *if a location is not below a block then the location is free*. 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 *free* is *the location*. Only the predicates contained exclusively in the *if-clause* are added to the predicate structure. The leaf node for an input object is a special input predicate that returns the objects that the new predicate tests. A more complex example is discussed in Section 2.5.

2.2 Interpretation Phase: Structure Matching

The final problem is to interpret the declarative predicate structure using internal knowledge about functions, primitive predicates, and learned predicates, and ground 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.

Because Rosie must have an instance of the described concept visible to perform this interpretation, Rosie will request the user to set up a state that contains at least one instance of the described concept. For example, it will request that the user show it an example of a goal state when learning a goal concept.

Figure 3 displays an example external environment that can be used to interpret the Blocks World goal described above. This interpretive process can be decomposed into two parts: first, the constituent predicates of the structure are evaluated, producing 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.

2.2.1 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 *number of*, Rosie uses the internal *count* operation. This process completes when results have been calculated for every predicate in the learned representation.

For the example Blocks World goal concept and the world state shown in Figure 3, Rosie first evaluates the unary predicate $block$ over all the objects in the world state (A, B, C, X, Y, Z), match-

ing only against A, B, and C. Due to the ordering constraints, when Rosie evaluates the predicates for *small, medium,* and *large,* they only test the blocks A, B, and C. The predicates for on and $below$ are evaluated last.

2.2.2 Joining

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 result will be objects and values in the external world that satisfy all constraints. For the Blocks World goal, in Figure 3, 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.

2.3 Post Interpretation

Once the declarative predicate structure has been successfully interpreted and grounded in the agent's perception of its environment, the task concept's usage can be applied. For actions, the successful interpretation indicates the availability of legal actions, which Rosie should propose to search or to act in the world. For goals, it indicates a detection of the goal state: the agent has won the game. For failures, it indicates a detection of a terminal state: the agent has lost the game. For learned predicates, it indicates a successful match of a predicate that is part of the definition of another concept. This last step is what allows Rosie to evaluate hierarchical compositions of concepts.

2.4 Dynamic Compilation 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. One of the features of Soar is a learning mechanism called *chunking*. 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 (predicate matching and term linking).

During predicate matching (Section 2.2.1), Rosie learns rules to evaluate primitive unary features, spatial relations, and internal functions. Some predicates require multiple rules to be learned, such as those dealing with functions and sets of objects. During the process of linking the linguistic term to the learned structure, Rosie learns rules to retrieve the predicate structures. These rules test the name of the concept used and whether the concept is a goal, failure, action, or new predicate. These rules then add the corresponding structures into 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. These rules prevent Rosie from having to repeat that same interpretive processes to determine matches between the predicate structure and internal and external knowledge.

1. If an object is not below a block then it is clear.
2. You can move a clear block onto a clear location. [Blocks World]
3. If a block is on a location that is adjacent to a clear location then you can move the block onto the clear location. [Eight Puzzle]
4. The goal is that there are eight matched locations. [Eight Puzzle]
5. If the value of a location is the value of the tile that is on the location then the location is matched. [Eight Puzzle]
6. If the color of a location is the color of the block that is on the location then the location is matched. [Eight Puzzle]
7. If a block is red then it is your block.
8. If a location is below your block then it is captured.
9. The goal is that all locations are covered and the number of captured locations is more than the number of occupied locations. [Othello]
10. If a block is on a boat then the block is a passenger of the boat. [Fox Puzzle]
11. You can move a passenger of the boat onto the current bank. [Fox Puzzle]

Figure 4. Example sentences from different tasks.

2.5 Illustrative Examples

The ability to compose hierarchies of concepts enables Rosie to learn many different types of concepts. Essentially Rosie can learn new classes of knowledge based on the types of available primitives. In various domains these primitives have included concepts such as colors (*red, green*), sizes (*large, small*), prepositions (*next-to, smaller-than*), labels (*location, destination*), and functions (*count, attribute-of, comparison*). Through hierarchical composition of these primitives Rosie can learn new prepositions (*adjacent*), labels (*captured, current, current*), 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 concepts, are described further below.

The learned concepts 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 primitive unary attribute used to designate the couples is *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 concept *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.*

Figure 4 shows example sentences for teaching concepts that Rosie learns in different games. Line 1 shows a definition of the task-general predicate *clear*, which can be then used in other tasks as shown in line 2, where *clear* is used to define the Blocks World action. Both *clear* and the learned predicate *adjacent* are used to define the action for Eight Puzzle (Line 3).

Line 4 shows a description of the goal for Eight Puzzle, and lines 5 and 6 show that the new predicate *matched* can be defined in different ways depending on the particular instantiation of Eight Puzzle in the world (either as numbered tiles or colored blocks). Lines 7 and 8 show a description of a context-dependent predicate *your,* which is then used to define another predicate *captured*. Line 9 shows a definition of the goal for Othello using many different learned predicates, including the hierarchically defined predicate *captured* (and *covered* and *occupied*), as well as quantification (*all*) over a set of objects, a function (*number of*), and comparator predicate (*more than*). Finally, Line

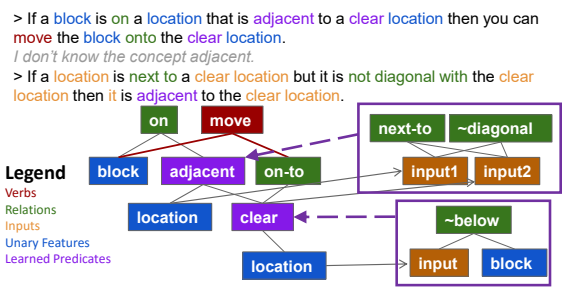


Figure 5. A simplified graphical structure of the representations Rosie learns for the action in Eight puzzle. Included are graphs for the hierarchical concepts used (clear) and taught (adjacent). The words and boxes are highlighted according to their part of speech or type.

10 shows a definition of the function word *passenger-of*, which is then (in line 11) used to define an action in the Fox puzzle.

Figure 5 shows a graphical representation of the learned structure for the action from line 3 in Figure 4 together with additional structure from learning the predicate *adjacent*. This illustrates the internal declarative structures that are built up when learning these concepts. Each predicate is labeled by type: unary features, relations, functions, inputs, and learned predicates. In addition the graph shows the argument attachment of the verb, which is exclusive to action concepts. The subgraphs for the new predicates *clear* and *adjacent* are also displayed, with the accompanying text for teaching *adjacent.* 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. Figure 5 displays the mappings of the inputs and results in the hierarchical structure.

Non-hierarchical, or flat, representations are problematic not only for computation and communication efficiency, 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 5, the sentence would become: *If a block is on a location that is next to a location that is not below a block but it is not diagonal with the location then you can move the block onto the second location.* Determining the correct attachment of the clauses and the object references is difficult: which references are to the same objects? Hierarchical concepts help reduce the number of objects, predicates, and overall complexity.

3. Evaluation

The extensions to Rosie have made it possible to expand the tasks it can learn to include six games (Tic-Tac-Toe, Three Men's Morris, Picaria, Nine Holes, 5x5 Othello, and simplified Risk-like game) and eleven puzzles (Tower of Hanoi with 3 blocks, ToH-4 blocks, Eight Puzzle, Five Puzzle, Missionaries and Cannibals, the Jealous Husbands problem, Frogs and Toads Puzzle, a simple Maze, a Blocks World problem, the Fox puzzle, and simple Mahjong Solitaire). We refer to all these generically as *games.* These games are embodied in a table top robot and a simulated environment.

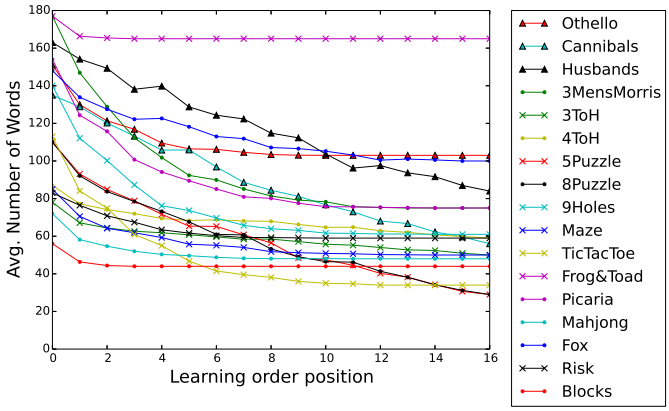


Figure 6. The number of words required to teach each game, as influenced by previously learned games. Results are averages of 3000 permutations of the 17 games.

Games 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 in all cases. For the puzzles, it correctly solves them using iterative deepening, and for the games, it selects legal moves and detects when it wins or loses.

Many of our claims concern the transfer of knowledge between games and the ability of the system to efficiently scale as multiple games are learned. To test these claims, we designed an experiment where we teach 3000 randomly generated permutations of all seventeen 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. To simplify the experiment execution, we created rules in Soar that internally simulate the external environments. For example, when teaching Blocks World, rather than physically setting up the puzzle in the world, as shown on the left in Figure 3, a message (setup-initial-blocks) updates the internal world state to the stored symbolic representation that is shown on the right of the figure. This simulation had no impact on what was learned, but eliminates the time to type instructions and setup game states. All experiments were run on a desktop computer on a single core.

3.1 Communication Efficiency

We first evaluate 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.

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 16 all other games have been taught. Moving from left to right, many games require fewer words, with the largest

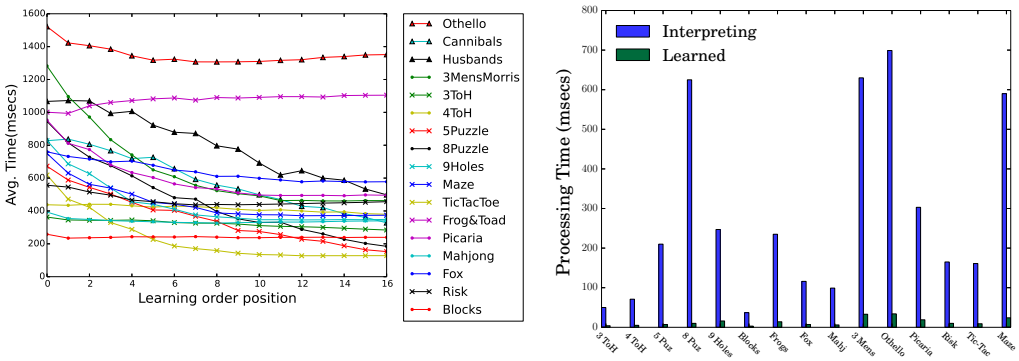


Figure 7. On the left: the total processing time required to learn each game, influenced by previously learned games. Results are averages from 3000 permutations. On the right: the processing time required to interpret (match) all concepts for each game individually compared to the processing required once learned.

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*), require very few words by the end. 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.

3.2 Processing Efficiency

Transfer of learned knowledge, both declarative and procedural, should also decrease the overall processing time required to learn a new task, although the increase of knowledge could also potentially increase processing time. Figure 7, on the left, shows the average processing time required to teach a game based on its position in the teaching order. Note that overall, Rosie is very efficient; 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. The computational cost of added knowledge from previous games, minus the benefit from transfer, can be seen mostly clearly in Frogs and Toads, which contains the fewest transferable concepts. This slowdown is not substantial, even in the worst case for Frogs and Toads, which shows an increase in total average processing time from 1000 to 1104 milliseconds. The average processing time per instruction, essentially Rosie's response time to an input sentence, increases only to ~92 ms from ~83 ms. The average response time over all tasks and orders is ~45 ms. When speaking or typing sentences to Rosie, the communication time dominates the processing time.

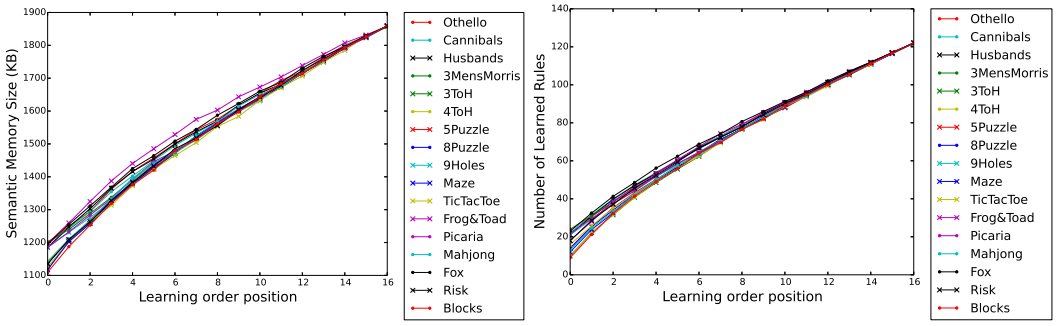


Figure 8. On the left: the cumulative growth in semantic (long-term) memory for all games. On the right: the accompanying growth in procedural memory (number of rules).

Our next experiment focuses on the impact of procedural compilation on performance. Our hypothesis is that the rules learned by chunking will be more efficient than maintaining only the declarative task representations and then interpreting them by deliberately matching them against the game state each time the task is attempted. On the right, Figure 7 shows the processing time to learn each game, omitting the time taken to parse the sentences and construct the declarative predicate structure. 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.

3.3 Memory Efficiency

Learning new tasks involves adding different kinds of knowledge to the agent's memories. Soar, and most cognitive architectures, maintain semantic knowledge in long-term memories while reserving short-term working memory for data relevant to the current task. Soar agents scale well with growth in semantic memory and procedural memory, but less so with growth in working memory.

Figure 8 shows the growth in both semantic and procedural memories. Not surprisingly, as Rosie learns new tasks, knowledge in both semantic (database memory in KB) and procedural (number of rules) memory grows approximately linearly. The different permutations converge to the same value, confirming that the total knowledge learned for all 17 games does not depend on the order in which they are taught.

Figure 9 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) which represent 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,

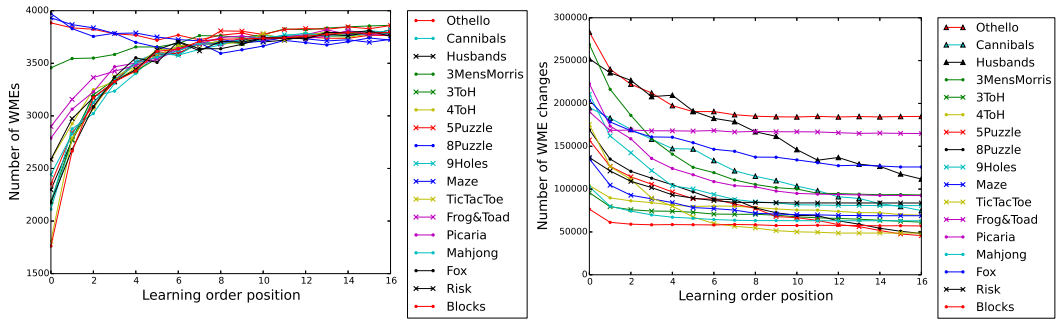


Figure 9. On the left: the growth in maximum working memory, measured in working memory elements, for each game. On the right: the number of changes to working memory to teach each task in the given order.

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.

The number of working memory changes (additions and deletions) 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 a 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.

4. Related Work

Below we focus on systems that learn task structures directly from language. See (Kaiser, 2012) and (Barbu, Narayanaswamy, & Siskind, 2010) for systems that learn single games from observation. Cantrell et al. (2012) describe a mobile robotic system that can be taught individual commands via language by specifying preconditions, action definitions, and post-conditions. It does not learn new predicates, nor can it learn tasks that involve constraints, failure states, and specific goal conditions. Thomason, Zhang, Mooney, and Stone (2015) describe an agent that incrementally learns a semantic parser through interactive natural language in robotic domains. Their system creates a lambda-calculus representation that includes predicates defined over objects, however it cannot learn new tasks or concepts.

Simon and Hayes (1976) proposed UNDERSTAND, whose purpose was to extract operators and goals from natural language specifications of isomorphisms of the Tower of Hanoi puzzle. Unfortunately, the language translation and concept learning processes were only hand simulated, and although inspirational, it is difficult to evaluate its generality or efficiency.

More recently, Hinrichs and Forbus (2014) developed an agent in the Companions architecture that learns to play Tic-Tac-Toe and Hexapawn through a combination of language and sketching. Their system generates a GDL (Game Description Language) specification (Love, Hinrichs, Ha-

ley, Schkufza, & Genesereth, 2008) of the task, which is then interpreted to play the game. Their approach focuses on the naturalness and the accessibility of interaction that is possible with multimodal interaction and does not address efficient execution and learning complex transferable knowledge. Research on transfer of knowledge with agents that use GDL specifications has so far focused only on policy transfer (Banerjee & Stone, 2007). This kind of approach, where some formulation of a game is readable by an agent or convertable into executable code, does not explore how portions of a specification can transfer between games, or the effects of acquiring many successive games.

In general, these other approaches assume that only one task will be learned and that concepts can be directly mapped (one-to-one) to known primitives. They usually divide learning and acting into separate processes where learning is often an off-line batch process. In Rosie, many tasks can be learned and task instruction is fast, interactive, and on-the-fly.

5. Discussion and Future Work

In this paper we have shown how Rosie can learn complex, compositional concepts across a variety of games, and that it can transfer what it learns to similar games, eliminating unnecessary interactions, and speeding the learning of new tasks. We show that by learning efficient procedural representations, Rosie can learn new tasks without significant performance penalties.

One broad observation it that there is a large difference in efficiency between interpreting declarative structures and executing procedural knowledge. We hypothesize that future Interactive Task Learning agents will need to learn representations native to the underlying agent architecture, ideally mirroring the representations and processing of hand-coded solutions. A general ITL agent will need to support the acquisition of many-to-many term groundings for both general and task-specific concepts. We also hypothesize that they will need to support hierarchical composition of concepts to enable efficient knowledge acquisition and transfer. Systems without these capabilities will have a difficult time supporting efficient execution and interaction after learning many different tasks.

In the future, we plan to expand the types of learnable concepts by including more quantifiers, functions, and grammar constructions. These will allow us to study transfer and scaling of knowledge across a wider variety of more complex games. One important shortcoming is Rosie's inability to efficiently reason over large numbers of objects in a single state. This issue is the reason we 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 are needed so that all concepts are not simultaneously computed for every state. 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. Finally, our focus has been on learning the rules of a game. 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.

Acknowledgements

The work described here was supported by the National Science Foundation under Grant Number 1419590 and the AFOSR under Grant FA9550-15-1-0157. 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 NSF, AFOSR, or the U.S. Government.

References

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 on* (pp. 1879–1886). IEEE.

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*.

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

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

Kirk, J. R., & Laird, J. E. (2014). Interactive task learning for simple games. *Advances in Cognitive Systems*, *3*, 13–30.

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

Laird, J. E. (2014). *Report of the NSF-funded Workshop on Taskability Revised Title: Interactive Task Learning*. Technical report, National Science Foundation.

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

Mohan, S. (2015). *From Verbs to Tasks: An Integrated Account of Learning Tasks from Situated Interactive Instruction*. Ph.D. thesis, University of Michigan, Ann Arbor.

Mohan, S., Mininger, A., Kirk, J. R., & Laird, J. E. (2012). Acquiring Grounded Representations of Words with Situated Interactive Instruction. *Advances in Cognitive Systems*, *2*, 113–130.

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

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* (pp. 1923–1929).