Advances in Cognitive Systems 3 (2014) 13-30 Submitted 9/2013; published 7/2014

Interactive Task Learning for Simple Games

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

Abstract

We present an agent, called Rosie$_{TAG}$, that is implemented in Soar and interacts with an external robotic environment. Rosie learns new games through interactive instruction with a human via restricted natural language. Instead of learning policy or strategy information, as is common in other game learners, Rosie learns multiple game formulations (the objects, players, and rules of a game) and then uses its own general strategies to solve them. We describe the structure and functionality of Rosie, and evaluate its competence, generality, communication efficiency, communication accessibility, and ability to continuously learn and accumulate new tasks and new task knowledge.

1. Introduction

Although there have been many advances in the past decades in the problem-solving ability of machines, there have been few advances in the ability of machines to acquire representations of novel problems and tasks from natural language. The predominant method of communicating the problem space of tasks is through standard programming languages, such as Java, Lisp, or C++. Cognitive architectures, such as Soar (Laird, 2012), Icarus (Langley et al., 2004), FORR (Epstein, 2001) or ACT-R (Anderson, 2007) provide higher levels of abstraction; however, the user must still program the tasks at the symbol level, using symbol structures that are specific to the given architecture and must include detailed instructions for coordinating the architecture's processing components and memories as opposed to only describing the structure of the task.

Our long-term goal is to create *taskable* agents (Langley, Laird, & Rogers, 2009) – independent intelligent agents that can perform a wide variety of tasks and can *learn* new tasks while they are behaving in the world, no longer a prisoner to their original programming. The more immediate goal is to take an important step toward general taskable agents by developing agents that learn from interactive instruction with humans via constrained natural language.

General taskability, especially in agents embodied in the real world, is extremely challenging. It requires the integration of many areas of AI, including natural language processing, dialog management, object recognition and perception, actuation, language and concept grounding, spatial reasoning, knowledge representation, and general problem solving. Moreover, it requires that an agent operationalize an external description of a task, converting it into a representation where it can attempt to solve it. Recent progress in AI, and more specifically in cognitive architectures, makes us optimistic that we now have sufficient understanding of these capabilities and their integration so it is possible to create robotic agents that can learn to understand and solve novel tasks through interactive instruction.

© 2014 Cognitive Systems Foundation. All rights reserved.

In this paper, we describe an agent that we have named Rosie (RObotic Soar Instructable Entity). Rosie can learn the formulation of a novel task, not a specific policy for controlling behavior in a task, but the definition of the task itself. Thus, Rosie learns a description of the task actions, as well as descriptions of goal and failure states, and then uses those descriptions to perform the task. In other prior work on taskable agents, such as by Langley et al. (2010) and Allen et al. (2007), a human teaches a specific policy for solving a task, such as giving directions to a location. These approaches teach a solution to a problem, rather than teaching the problem specifications, which is sufficient for tasks where there is a single, fixed solution for all problem instances. However, it is not applicable to tasks where the solution depends on the initial state of the problem or an opponent's actions, and where problem solving and search are necessary, such as in many puzzles and games.

Not surprisingly, we initially focus on developing an agent that can learn simple spatial games and puzzles, such as Tower of Hanoi, Tic-Tac-Toe, Connect-3, and the Knight's Tour. There are many advantages to starting with games. They include complex spatial relations with many different constraints, objects, and goals that are reflective of many real world tasks. Games are well structured, have clearly defined problem spaces and goals, and require problem solving to play or solve. For the remainder of this paper, we often refer to games and puzzles as *tasks*.

In contrast to learning task specifications, the majority of existing work on game learning has focused on learning strategy. Early examples include systems that learn to play Checkers (Sammuel, 1959), Backgammon (Tesaauro et al., 1989), and Othello (Buro, 1999). These systems are preprogrammed with knowledge of the game's state descriptions, legal actions, and goals; rather than learning *how* to play, they learn to play *well*. We describe an agent that learns such simple tasks through interactive instruction. Following the acquisition of the task, the agent uses its innate problem-solving capabilities to generate appropriate behavior to perform the tasks.

We have identified five major desiderata for instructable, taskable agents that serve as the basis of the claims of our research:

D1. Competent.

The agent has the capability to learn the knowledge required to attempt a task and the capabilities necessary for making progress on the task. We define competence as the ability to comprehend and attempt the task independent of any strategic competence. We make the additional constraint that the tasks require interaction with an external environment. We claim that our approach is sufficient for our agent to be competent in the tasks we teach it. We evaluate this claim by demonstrating that it can solve the tasks in a mixture of a simulated and a real-world robotic environment.

D2. General.

The agent can learn a diverse set of tasks, which require a diverse set of concepts (objects, relations, actions). Meeting this challenge requires avoiding task specific representations or processing mechanisms. We claim that our approach is sufficient for our agent to learn a wide variety of tasks. We evaluate this claim by demonstrating that we can instruct our agent on 11 tasks: Blocks World, Connect-3, Frogs and Toads, Knight's Tour, Tic-Tac-Toe, Tower of Hanoi, Peg Solitaire, Sokoban, Four Queens, River Crossing (Fox, Goose, and Beans), and the Five Puzzle.

D3. Efficient Communication.

The communication of the task should be concise and efficient. We claim that the amount of information required to instruct our agent is much less than that required in other common

approaches and it approaches the efficiency of natural language. We evaluate this claim by comparing the number of symbols used in our approach for a subset of tasks to those required in Soar and the Game Description Language, as well as to published natural language descriptions of those tasks.

D4. Accessible Communication.

The language for communication should require minimal knowledge of the underlying computational architecture and should require minimal training to use. We claim that our approach is more accessible than existing methods for task acquisition. Our evaluation of this claim is based on inspection of the specifications for tasks in our approach. As we discuss later, this area requires more research to strengthen our results and develop better empirical evaluations.

D5. Continuous, Accumulative Learning.

Instructable agents should not only accumulate their learned knowledge, but also transfer knowledge learned from a previous task to a new task, thereby reducing interactions with a human mentor and speeding task acquisition. We claim that, in our approach, the learning process allows for refinements and requests for additional knowledge. We evaluate this claim by demonstrating that our agent transfers knowledge learned in one task to later tasks, reducing substantially the amount of instruction required in the later tasks.

We will return to these five desiderata in Section 3 when we evaluate our claims, but first we describe the system that supports them.

2. Task Acquisition System

Our approach builds upon previous research on learning through situated interactive instruction (Mohan et al., 2012) in an agent we now call Rosie (RObotic Soar Instructable Entity). This agent is embodied in a robot arm and Kinect sensor in a table-top world, where the arm can manipulate foam blocks. Rosie uses the Kinect sensor to detect the color, shape, size, location, and orientation of the blocks. A chat window lets it communicate with a mentor using restricted natural language. A simulator, developed by the APRIL lab, provides an accurate replication of the environment and has been used in many of our experiments.

A human teacher and Rosie use mixed-initiative (communication is bidirectional) and situated (embedded in a shared real-world environment) interaction. The agent learns the meaning of new nouns (*square, rectangle, triangle, …),* adjectives (*red, large, …),* prepositions (*in, to the right of, on, …),* and verbs (*move, store, cook,* …) that refer to objects, properties, spatial relations, and actions in its environment. When learning new words, Rosie learns procedural knowledge (actions and control knowledge for verbs), object descriptions (such as feature detectors for the colors, shapes and sizes), and spatial relations. After Rosie learns a new word, the human can use it freely in future instructions (*move the green block to the pantry).* The agent can use learned words when responding to questions (*that is a large green block*).

Rosie has only limited initial domain knowledge: action knowledge for performing its primitive actions (picking up a block and putting down a block), feature-space knowledge of object attributes (the fact that objects have a color, a size, and a shape), and primitive spatial relations about the alignment of objects. The agent has additional task-independent processing knowledge for parsing sentences, managing interactions with the human, and learning new words. Spatial reasoning is done within Soar's Spatial Visual System (Laird, 2012), which encodes a

three-dimensional model of the world as a scene graph and allows for mental projection of imagined objects and relations. Mohan et. al (2012) describe the details of how Rosie learns new words from this initial knowledge.

Rosie is implemented in Soar (Laird, 2012), which has been in development and use for over 30 years and has been applied to a wide variety of domains and tasks, including prior research on instruction, natural language understanding, and robot control. Recent advances in the architecture, including a spatial visual system, episodic memory, and semantic memory, have broadened the types of knowledge it can represent, use, and learn. One of our hypotheses is that the collective compatibilities of Soar are now sufficient for creating a general taskable agent.

This research introduces a new agent, Rosie$_{TAG}$ (Task Acquisition for Games). The task acquisition knowledge is encoded in Soar as an extension to Rosie and includes knowledge to support an interactive dialogue where the human introduces a new task, then incrementally and interactively teaches it the available actions, goals, and constraints. For convenience, we will refer to the extended agent as Rosie.

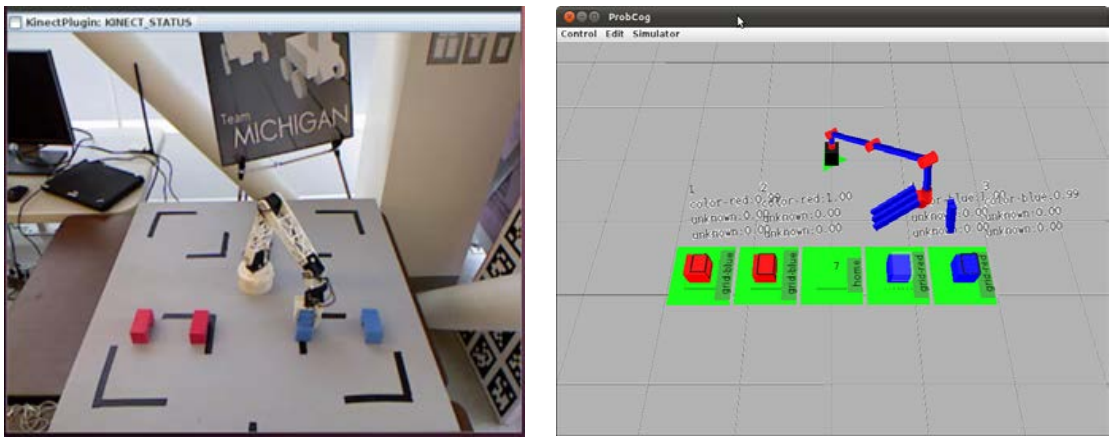

Figure 1. Images of Rosie solving the Frogs and Toads puzzle in the real world (on the left) and in simulation (on the right).

2.1 Task Characterization

As mentioned earlier, this work focuses on games, which can be single or multiplayer, with single player games more commonly known as puzzles. Game formulations correspond to a definition of a problem space: the legal actions in the game, their precondition and effects, and descriptions of goal or failure states. Some game formulations may also include information, such as the initial game state, which is necessary for agents not grounded in a representation of the game. Thus, acquiring a task also requires learning object, action, and spatial concepts.

We take advantage of the fact that many games can be fully described by spatial relationships between different types of objects and locations. For example, in Tic-Toe-Toe, the objects are X's and O's that are placed on a three by three grid. The important spatial relation is whether three objects of the same type are in a line. Actions are defined by spatial preconditions and spatial effects, and goals are described by a set of desired spatial relationships. For example, in Tic-Toe-Toe, the action is to *place X in a grid location*, with the precondition that the location is clear.

This tight coupling of games and spatial relations/reasoning requires the ability to learn representations of new spatial relationships. New spatial relations must be operationalized so that they can be used for both recognition and object placement.

Although a goal of this research is generality, there are many limits to the types of games that it can learn. A game must be fully observable, turn-based, deterministic, describable by spatial or descriptive constraints, and playable with only discrete actions, or have an isomorphism to a problem with these constraints. Currently, the game rules cannot require access to history or temporal knowledge of the game play. For example, the agent could not learn the rule for castling in Chess, which requires knowing whether the king has moved.

The system is also limited to learning games with objects of two different classes: blocks and locations. Blocks can vary in size, shape, and color and those variances can be relevant to game play. Locations are virtual objects defined in the world simulation that can have fixed names such as final-pillar or grid-red. The types of games that can be described with these restrictions are quite large and include many board games, basic puzzles, and some three-dimensional problems. Some tasks that do not conform to these constraints can be described as an isomorphism that does. Most of the games studied involve a small number of objects and some are smaller versions of well-known games. For example, the Five Puzzle is a smaller variant of the Eight Puzzle, and Connect-3 is a smaller variant of Connect-4. The restriction on the number of objects is not a fundamental limitation, but it leads to smaller problem spaces and lets the agent quickly solve problems using simple brute-force search techniques, which are not currently a focus of this research.

2.2 The Acquisition and Operationalization Process

The task acquisition processing and operationalization has three phases.

1. The agent acquires the rules of a new task through situated interactive instruction and translates the information into a Task Concept Network (TCN), which is stored in Soar's semantic memory. The TCN specifies all of the action knowledge, failure conditions, and goal states that are needed to determine legal play and find a solution. Through this process the agent also interactively acquires all of the necessary spatial relationships, object descriptions, and verb knowledge, as described in the beginning of Section 2.
2. When given a task, the agent uses precoded procedural knowledge to retrieve relevant TCN information from semantic memory, extract the needed information from the world, and then interpret the TCN to detect conditions, and find complete matches for legal actions, and goal and failure states.
3. During play, the agent chooses its next action based on an internal lookahead search that uses the action knowledge to simulate actions on an internal model of the environment, searching for goal states while avoiding terminal failure states. Once an action is chosen, the agent executes it in the external environment using its learned verb knowledge.

We describe these phases in more depth in the remainder of this section.

2.2.1 Language Interaction

The agent's main learning focus is to acquire descriptions of the relevant objects, their properties, and the spatial and descriptive constraints between those objects. These are subsequently used to

describe the preconditions for legal actions and the conditions for failure and goal states. Each descriptive set contains a number of parameters described by the mentor in response to agent queries for information. Following our goal for task acquisition to be incremental and continuous, the agent asks for additional training information whenever it encounters novel or unknown concepts, such as the verb *move* or the preposition *left of*. For task actions, the human also must specify which verbs need to be performed to carry out the action.

The acquisition of a task begins with the mentor specifying the name of a new game: "the game is tic-tac-toe." If the agent already knows the game, it prepares to play/solve it when it is informed that its turn has come. Otherwise the agent initiates a structured dialog with the mentor. Future work will address this restrictive structured ordering by making the process more mixed-initiative, such as allowing the mentor to decide what to teach first, goals or actions. The intent of this dialog is to learn legal actions in the game and their conditions, and the goal and any failure conditions of the game. An example of this dialog between $\text{Rosie}_{\text{TAG}}$ (R) and mentor (M) for Tic-Tac-Toe is displayed in Table 1.

Table 1. Dialog between Rosie and Mentor during acquisition of "place" for Tic-Tac-Toe.

	Dialog	**Line**
M:	The game is tic-tac-toe.	1
R:	I don't know that game, how many players are there?	2
M:	Two.	3
R:	Please start by teaching me the name of a legal action in the game.	4
M:	Place	5
R:	What are the verb and parameter arguments associated with this action?	6
M:	move 1 to 2	7
R:	Ok I know that verb, what kind of object is the first parameter? (block, location, or either)	8
M:	M: block	9
R:	Please list all constraints for this parameter (such as 'it is red' or 'it is on [parameter] 2') and then finished	10
M:	it is red	11
M:	it is not on a location	12
M:	Finished	13
R:	What is the type of the next parameter? (or finished if no more)	14
M:	Location	15
R:	What are the constraints?	16
M:	it is not under a block	17

This interaction specifies the complete precondition knowledge for a move in Tic-Tac-Toe, as well as the action knowledge for what parameters to manipulate and how to manipulate them (learned previously from teaching examples of *move*.) If Rosie had not already learned *move*, it would query for teaching demonstrations of the unknown concept. More generally, two types of parameters can be taught: blocks and locations. In addition, there are two types of constraints that can be taught: descriptive and spatial. Descriptive constraints cover colors (*red*, *blue*), shapes (*cross*, *rectangle*, *cylinder*), sizes (*small*, *large*), and names (*grid-green*) of objects. Spatial constraints cover positive and negative relationships between two or three objects, such as *it is*

[not] above a location. The spatial descriptions can refer to a general type of object, such as *a* location, or refer specifically to another parameter. This is done by directly referring to the number of that parameter: *it [the block] is on 3*. All of these concepts are learned through interaction, with two exceptions. S*maller than* and *linear with* are preencoded in the agent's spatial visual system. *Linear with* is the only three-object spatial constraint currently accepted. These concepts are grounded in representations in the spatial system, and they make direct connections to perceptual input from the task environment.

After finishing all actions, the mentor is asked to name a goal and provide the parameters and related constraints in the same manner as demonstrated above. This is also done for any failure states (such as having a large disk on a smaller disk in Tower of Hanoi). There can be any number of parameters and constraints, which provides flexibility in describing complex relations that require a large number of parameters or spatial relationships.

At times, the dialog can be verbose and involve not only a large number of interactions, but also require the mentor to remember the specific ordering of the parameters already specified, and possibly ones that have not yet been specified. In particular, when teaching multiple tasks, it is frustrating to be forced to specify the same types of constraint concepts, whereas learned knowledge about prepositions, adjectives, and verbs will transfer to new tasks. To combat this issue, and be more faithful to the idea of incremental learning, Rosie can learn names for concepts, like the action *place*, as seen in the dialog above, or the goal *three-in-a-row*. This requires that the names of distinct concepts for actions, goals, and failure states be unique.

2.2.2 Task Concept Network

The task information that is learned through instruction is stored in Rosie's semantic memory. We call this structure the Task Concept Network (TCN). The portion of the TCN created from the dialog for Tic-Tac-Toe in Table 1 is displayed in Figure 2 (a). The figure includes only the task name and the action that is learned. It does not show the additional structure that is learned for the goal or failure states. In the game, the only action is A1, which is the *place* action in Tic-Tac-Toe. *Place* is implemented with the *move* verb, which Rosie learned from a previous task. The preconditions of a task action are encoded as parameters and their associated constraints. In this example, the *place* action has two parameters, P_1 (a block) and P_2 (a location). The parameter constraints were learned in dialog lines 10-17 in Table 1, and are associated with C1. Thus, P_1 (the block being *placed*) has an attribute constraint (it must be red) and a spatial constraint (it must not already be on a board location), which is represented by the structure under S1. P_2 (the destination location for *place*) has a single spatial constraint (it must not be under a block, i.e., it is clear), which is represented by the structure under S2. In general, tasks can have multiple types of actions, and the actions can have any number of parameters and associated constraints.

2.2.3 Knowledge Operationalization

Once Rosie has learned the Task Concept Network for a new task, it must be able to convert that knowledge into action when it encounters the task in the future. Thus, when the agent encounters a learned task, it retrieves the relevant Task Concept Network from semantic memory (looked up by the task name) and uses this information to determine valid instantiations of the action from the current visual scene. Rosie first indexes potential objects for each parameter in the action

conditions (P_1 and P_2 in our example) by querying the current visual scene for objects with matching descriptions.

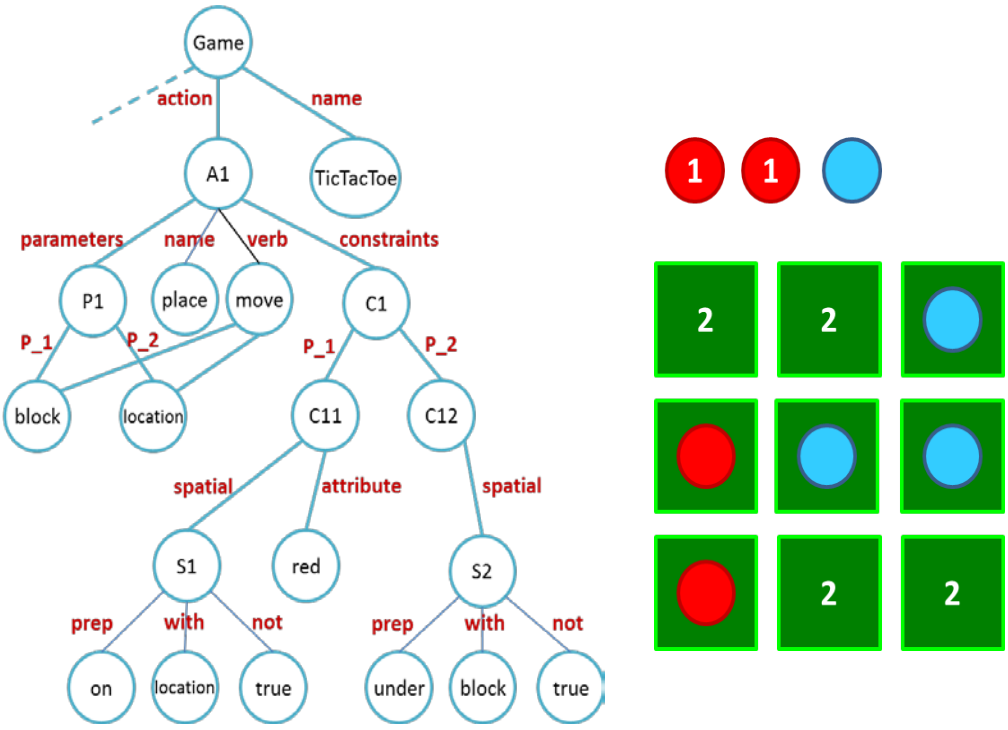

Figure 2. (a) Partial TCN graphical representation of an action in Tic-Tac-Toe. (b) The internal representation of the Tic-Tac-Toe board and the associated parameters.

Figure 2 (b) shows a representation of an intermediate state of a Tic-Tac-Toe game in simulation. The board is defined as a 3×3 grid of locations, with blue and red blocks representing X and O. In this game, the agent is playing O (red), and the opponent is playing X (blue). In this state, P_1 initially has potential matches to all the red blocks and P_2 has potential matches to all the locations. Rosie then prunes the potential matches using the parameters' spatial relationship constraints (S1 for the block and S2 for the location). The successful matches for P_1 are blocks labeled **1,** and the successful matches for P_2 are locations labeled **2**. The agent collects those objects that satisfy the constraints for each parameter and uses them to create the set of possible legal actions for the current state. In this case, there are eight possible actions, as each of the two red blocks can be moved to each of the four open positions. To avoid this needless duplication, we usually stack the red blocks, so that at any time there is only one block of each color available for movement. Rosie uses this same process to retrieve the task goal and failure state descriptions from semantic memory and then test to see if they match the current state.

If only a single task action can be applied to the current state, it is selected, and the associated verb is executed using the instantiated parameters. However, if there are multiple legal actions, Rosie reaches a *tie impasse*, which indicates that there is a lack of knowledge for

selecting the appropriate action. For single-player puzzles, it uses iterative deepening, implemented as recursive substates in Soar (Laird, 2012), to search for the goal. The search takes advantage of the Spatial Visual System's mental imagery capability to simulate potential actions using the learned verb actuation knowledge. For each state in the search, Rosie determines whether a goal or failure state is present, and if not, iteratively extends the search until the current depth limit is reached by generating the legal actions for the new state. 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.

The internal search is possible because even though the full action model of an action is not specified by the instruction, the agent has the action knowledge of the associated verb. 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 grounded knowledge of the verb, it can 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 much better than random behavior.

3. Evaluation

Although Rosie can interact with the real world, our primary mode of development and evaluation has been to use a simulation of the robot environment. The real world has additional problems unconnected to task learning, such as precision of actuation, object segmentation for touching blocks, and tracking the existence of objects during manipulation. These all add challenges, especially when the task requires stacking blocks. To date, Rosie has successfully executed four of these tasks (Tic-Tac-Toe, Tower of Hanoi, Connect-3, and the Frog and Toad puzzle) in the real world and the remaining in simulation. In this section, we present evaluations organized according to the desiderata presented in the introduction. This research is just the first steps in a long journey, so we use this opportunity to describe plans for future research related to each desideratum.

3.1 Competence

Rosie acquires task descriptions for eleven different tasks, listed in Table 2, in real time and can either solve them or, in the case of Tic-Tac-Toe and Connect-3, play a competent game. To facilitate quicker evaluations, in some cases, we created simplifications of existing tasks: the Five Puzzle instead of the Eight Puzzle; Connect-3 instead of Connect 4; Four Queens instead of Eight

Queens; and simple Sokoban puzzles. These simplifications were necessary to restrict the agent's search space, as well as to simplify the state representation that the agent needed to maintain – large state representations of similar objects can lead to combinatorics in matching rules, which slows execution. These simplifications are not fundamental limitations, as descriptions of games can include any number of objects, prepositions, colors, shapes, actions, or goals. However, the system does not currently have the necessary mechanisms for avoiding the computational performance issues that arise with large numbers of pieces or locations.

Our future goals include improving Rosie's ability to handle more complex tasks that contain more objects and constraints. To maintain efficient execution as the number of objects increase, we plan to introduce an attention mechanism that would let the agent focus on only a subset of the game state. We also plan on adding a mechanism to learn game-specific action models so that the agent can avoid simulations of its primitive actions. We also plan to improve Rosie's sensing and robotic capabilities so that it can solve all tasks in the real world. A significant milestone would be for Rosie to learn to play a competent game of real-world chess through instruction.

3.2 Generality

To evaluate the generality of the system, we instructed Rosie in eleven different tasks and then examined the diversity of concepts, actions, and goals required in them. These games range from simple board/grid type games like Tic-Tac-Toe to three-dimensional puzzles like Tower of Hanoi and Blocks World to more complex transport puzzles such as Sokoban. Table 2 lists the games that were taught, as well as the concepts required for each game.

Table 2. Description of the games learned and the concepts taught for each game.

Game	**Spatial Concepts**	**Actions**	**Goal**	**Failure**
Tic-Tac-Toe	on, under, linear	place	3-in-a-row	
Connect-3	on, under, linear	stack-place	3-in-a-row	
Tower of Hanoi	on, under, smaller	smaller-stack	stacked	
Five Puzzle	on, under, near, diagonal	slide	matching-location	
Frogs and Toads	left, right, on, under	slide-l, slide-r, jump-l, jump-r	side-swap	
Four Queens	on, under, linear	place	all-placed	no-attack
Blocks World	on, under	stack	order-stacked	
Sokoban	on, under, linear, diagonal	push, slide	blocks-in	
Peg Solitaire	on, under, linear	jump-remove	one-left	
Knight's Tour	on, under, aligned-vert, aligned-horiz	knight-a, knight-b	all-placed	
River Crossing	left, right, aligned	move-l, move-r, carry-l, carry-r	right-bank	fox-goose, goose-beans

The diversity of these tasks demonstrates that Rosie is general – it does not include task-specific knowledge, and it can learn a variety of concepts, actions, failure states, and goal states. Moreover, once the agent learns these tasks, it can solve any of them – it is not a one-trick robot. However, as described in Section 2.1, there are still restrictions on the types of tasks that Rosie can learn. The states of all these tasks are defined by objects and locations that are visible to the agent. The tasks have simple, spatially defined goals, and the actions can be defined by discrete movements of objects. A goal for future research is to expand Rosie's capabilities so it can learn more types of tasks. As the number of tasks increases, we will need to remove the assumption that the names for concepts are unique. The agent will need to determine from context which meaning is being used. If it cannot determine the correct meaning, it will need to query the mentor – *Is this the same as the action place from Tic-Tac-Toe?*

Within the context of research on AI, one interesting point of comparison is to the General Problem Solver (Ernst & Newell, 1969). This was the first system that had a general method for problem solving (means-ends analysis) and it could solve the 13 tasks that were encoded in it. Cognitive architectures have taken this type of generality much further, so that there have been at least a hundred different tasks encoded in ACT-R and Soar. With Rosie, the system *learns* the multiple tasks through instruction instead of having to rely on manual programming.

3.3 Efficiency of Communication

The goal of efficient communication is to minimize the effort required to instruct the agent in a new task. We take inspiration from evaluations of other forms of task specification in order to get some measure, however crude, of the amount of information that must be communicated to the agent. In programming languages, the number of lines of code is often used as a measure of the complexity of software. We propose a finer-grained measure, which is the number of symbol tokens required to communicate a task. For natural language, that includes the number of words and punctuation symbols, such as commas and periods. For computer languages, we include tokens, such as variable names and constants, reserved words, such as "if" or "while", and punctuation, such as parentheses and equal signs.

As we know of no existing benchmarks or methods of comparison, it is unclear how to appropriately compare Rosie against alternatives. We chose three tasks (Tower of Hanoi, Tic-Tac-Toe, and the Eight Puzzle) for the sake of diversity, and examined the number of symbol tokens required to specify these tasks using four different approaches that were determined in part by availability of data. Unfortunately there are no comparable implementations available in other task specification systems such as HERBAL (Cohen, 2008).

- *Instructions to Rosie*. These are the instructions that the mentor communicated to Rosie in order to learn the task. We examined two cases, one in which many common concepts that are shared among many tasks have already been learned (Rosie+) and another in which all concepts must be taught from scratch for each task (Rosie).
- *Natural language descriptions of the tasks*. Our data on natural language descriptions was derived from existing descriptions of the tasks "in the wild," such as in Wikipedia or AI textbooks. The values are averages of three independent descriptions. We expect that natural language will be very efficient for communicating these tasks.
- *Soar rules*. Soar is a state-of-the-art cognitive architecture and the amount of knowledge required should be representative of what is required to program other architectures. The

Tower of Hanoi and Eight Puzzle agents were developed many years ago, whereas the Tic-Tac-Toe system was developed by the authors explicitly for this evaluation.

- *The Game Description Language (GDL).* As described in the related work section, GDL is a language designed for specifying games for the General Game Playing competition. The GDL descriptions were found online and not developed by the authors.

In making these comparisons, we excluded the specifications of the initial state (which are found in the Soar and GDL descriptions) because our agent has access to the initial state through perception and does not need to learn it. In the future, we plan to extend Rosie so it can learn those representations, potentially from a visual demonstration. For our system, we have two different entries that correspond to the number of tokens needed to describe the game, with and without learned knowledge about common verbs, colors, and prepositions. The assumption is that, in many cases, Rosie will have already learned those concepts from prior instruction. Figure 3 shows the number of tokens required for each approach across the three tasks.

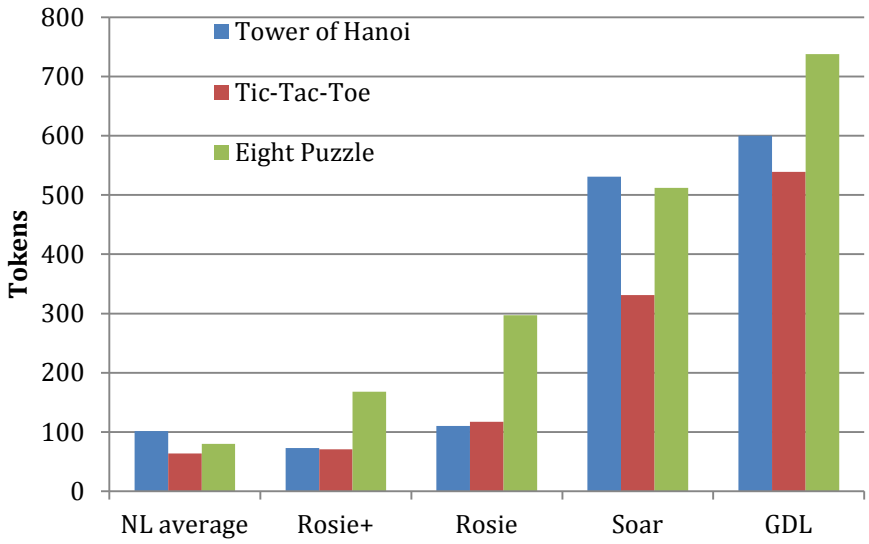

Figure 3. Comparison of the number of tokens required in different task specification approaches for three different tasks.

Somewhat surprisingly, Rosie is comparable in token communication efficiency to natural language, except for the Eight Puzzle. As illustrated by the data for this task, one failing of efficient communication in Rosie is in the specification of goal states. The goal specification includes 16 arguments, eight objects (identified by us with colors) that must be in eight locations (identified by their names). One of our future goals is to improve the instructional system to allow more natural, less pedantic, and more concise specifications such as 'all blocks are in a location of the same color.' In addition to learning the initial state by demonstration, the goal configurations could also potentially be learned by visual demonstration. These additional capabilities would reduce the number of interactions.

Both natural language and Rosie are significantly more efficient than Soar or GDL. Although it is encouraging to see that the number of words in our specification is close to natural language, and substantially better than Soar and GDL, we are hesitant to make any strong claims in terms of communication efficiency. The natural language descriptions are unstructured and assume

significant background knowledge and language expertise, whereas Rosie requires structured and grammatically limited instructions. Moreover, in contrast to Soar and GDL, it makes strong assumptions about the nature of spatial problems and games, and it has strong expectations as to what information it is expecting from the human mentor. It is possible that there is an underlying tradeoff between generality and efficiency that will compromise Rosie's advantages as we extend it to more tasks. However, there are indications that this communication efficiency is real. Rosie's ability to ground referents in the shared environment makes it possible for the teacher to use very short instructions. The inclusion of previously learned concepts (Rosie+ vs. Rosie) further reduces the information that must be communicated, which should continue to improve efficiency as the agent learns more and more tasks, as we discuss in Section 3.5.

3.4 Accessibility of Communication

We have not formally or empirically evaluated the accessibility of Rosie's method of communication. However, based on inspection, Rosie appears to be more accessible than a cognitive architecture, such as Soar or ACT-R, or a programming language, such as LISP, Java, or C++. Those approaches require that the mentor learn a specialized syntax and the structure of the architecture, including the memories systems and decision mechanisms.

However, there are still challenges in using Rosie, as it requires that the mentor understand Rosie's structuring of tasks (the concepts of parameters, constraints, and actions), and the mentor must be able communicate that structure using the limited instructional syntax prompted by agent-driven requests for the actions, failures, and goals. However, the interactive nature of Rosie has the advantage that the mentor does not need to know all of the concepts that Rosie has learned and also does not need knowledge of the underlying Soar rules or how the Soar architecture works. If an existing concept is used by the teacher, Rosie can use it. If a novel concept is introduced, the agent initiates a dialog to acquire it, leading the mentor through the interaction.

One of our future goals is to extend Rosie's language abilities by increasing its background knowledge of common concepts and by enhancing its ability to process a broader range of natural language. We also plan to find ways to move away from the structured dialogs so that the human mentor has more flexibility in specifying the task knowledge. Instead of referencing parameters with specific numbers, parameters for actions would be introduced in more concise, natural statements, such as *Move a free red block on to a clear location*. These changes should also help to reduce the number of interactions. The learning process is currently not robust to errors, so that teaching mistakes often require reinitializing the agent or reteaching an entire task. Future work should investigate capabilities to correct mistakes through interactive instruction. We will also explore other modalities for interaction, including using more gestural interactions and potentially sketching (Hinrichs & Forbus, 2013). Finally, we plan to develop improved methodologies for comparing and evaluating the accessibility of communication, drawing on those used by others (Cohen, 2008).

3.5 Interactive Continuous Learning

Our final evaluation focuses on Rosie's ability to learn interactively and continuously. One positive indication is that learning is cumulative. Rosie can learn more than one task at a time, building up a repertoire of games incrementally through instruction. More interesting is that not only are the tasks available, but the concepts learned in earlier tasks are available when a new one

is learned. This includes not only common concepts, such as the verb *move* and the preposition *on*, but also higher-level concepts that define the actions or goals in a task. Some benefits of this were evident in the comparison of Rosie to Rosie+ in Table 2. If an agent attempts to learn a game with no initial knowledge of spatial prepositions, adjectives, nouns, or verbs, the interaction is substantially longer than if it already knows those concepts. As the agent acquires knowledge

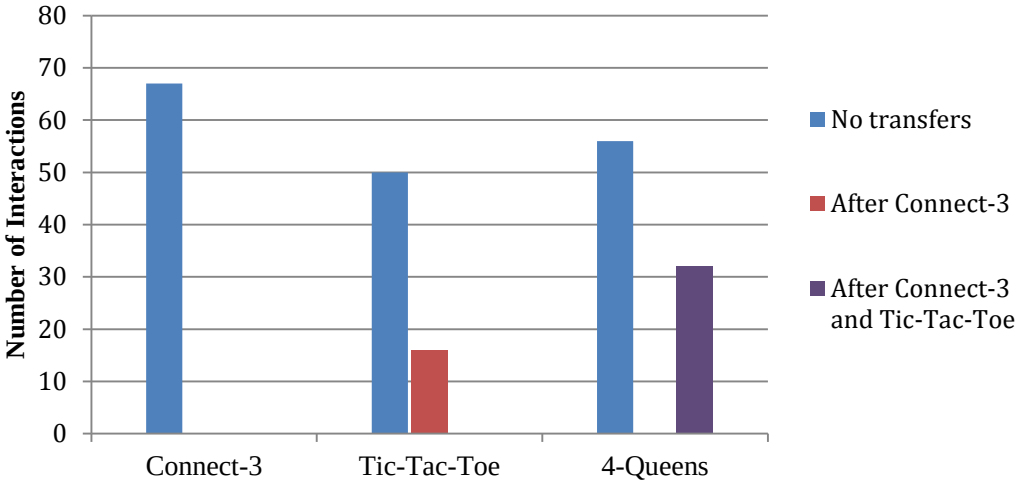

Figure 4. Transfer of knowledge of concepts evaluated by number of interactions needed to describe games separately and sequentially.

of concepts, fewer interactions are required, and concepts transfer to new games. The ability to learn and transfer common goal, action, and failure concepts further reduces interactions. Another goal of future work is to increase the number and types of concepts that can be taught to aid in transfer between games. For example, instead of describing a location as *not below a block*, it could be described simply as *clear*.

We evaluated the transfer of knowledge between interactions in an experiment.in which we taught Rosie three games, Connect-3, Tic-Tac-Toe, and the Four Queens puzzle, separately with no transfer and then in succession. Figure 4 shows the number of interactions, defined as number of separate mentor commands or responses, for each of these cases. The leftmost bar for each task shows the number of interactions required before any previous task was learned. For Tic-Tac-Toe, the second bar shows that the number of interactions drops substantially when Connect-3 is taught before Tic-Tac-Toe. Similarly, the number of interactions for Four Queens drops when Connect-3 and Tic-Tac-Toe are taught first. This improvement is partially due to the transfer of preposition, verb, and adjective knowledge between games, but it also results from the transfer of game concepts, namely the goal *three-in-a-row* (Connect-3 and Tic-Tac-Toe) and the action *place* (Tic-Tac-Toe and Four Queens). As further evidence of transfer, when both Connect-3 and Four Queens are taught before Tic-Tac-Toe the number of interactions drops to six (not shown).

4. Related Work

In this section we review related research, starting from those approaches that support task execution and specification, but not incremental acquisition, to those most relevant to our work that support dynamic task acquisition. We also discuss some related work on game learning that

focuses on strategy learning given a specification, rather than learning that representation. For each approach, we evaluate it according to the desiderata (D1-D5) from Section 1.

4.1 Task Specification Languages

Almost since the inception of cognitive architectures, there have been attempts 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). HLSR was also extended to compile into ACT-R. 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. Although they are more efficient and accessible than Soar, ACT-R, or ICARUS, these formalisms are not as efficient or accessible as natural language. Moreover, they all require independent batch systems that compile the task specifications into the target language, so they are not integrated into an agent that supports dynamic continuous learning (D5) or interactive task specification.

Salvucci (2013) introduced another approach to cognitive skill acquisitions within ACT-R. His work focused on the integration and reuse of previous skill knowledge (D5) and the proceduralization of this knowledge. Although achieving competent behavior (D1) in a diverse set of tasks, the commands are limited to a restrictive syntax (D4) that only specifies policy behavior rather than general task and goal knowledge.

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"). The instruction language is accessible (D4), possibly more so than our instruction language. However, it does not learn new concepts, such as spatial relationships, nor can it learn tasks that involve constraints, failure states, and specific goal conditions.

4.2 Game Description Language

The Game Description Language (GDL) is used in the General Game Playing competition (Genesereth & Love, 2005). GDL is a high-level formalism that lets one specify a large variety of games, although it is not easy for a human to write or interpret (D4). Multiple systems interpret GDL specifications, achieving competence (D1) and generality (D2), but none of them learn incrementally and continuously (D5). There is no goal for communication efficiency (D3), and the degree of efficiency remains an open question (see Section 3).

Thielscher (2011) surveyed the current state of research in general game playing and highlighted areas that are not being investigated. One area is spatial reasoning. GDL specifies relationships like on(x, y), but these are not connected to any spatial meaning and are just a name for a proposition that can be true or false. Spatial reasoning requires not only learning what those terms mean, but also learning how to ground that spatial knowledge in the game environment so that these relationships can be recognized and modified though actions. Another neglected area is natural language processing. Research in this area recognizes that GDL is not a language that most humans would select to specify a game, even those who are technically competent. A third area is game-playing robots, and the more general desire for robots than can handle arbitrary new

tasks. Our work tackles all three of these areas and the issues that arise from their integration, as well as providing a means to incrementally and interactively provide the problem formulation.

4.3 Learning Game Rules

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. This work included human trials studying how different specifications, or isomorphs, of a problem influenced the human's internal task representation and problem solving techniques. The program was limited in generality (D2) to these isomorphs that involved list manipulation actions, such as Transfer X to Y. The input descriptions consisted of natural paragraph form descriptions (D4) that even contained some redundant and irrelevant information. However, much of the translation work was done by hand simulation.

Current research that parallels the work described here, by Hinrichs and Forbus (2013), uses CogSketch to teach an agent developed in the Companions architecture how to play simple games like Tic-Tac-Toe and Hexapawn. In their approach, the system incrementally creates a GDL description of the task from interaction with a user that includes instruction and demonstration. The GDL specification is then interpreted so that a Companions agent can play the game. Their approach focuses more on the naturalness and the accessibility of communication (D3, D4) than on the generality (D2) of the information communicated. They plan on extending the generality to include abilities to learn vocabulary for new concepts, such as spatial relations.

There has been some prior research on learning the rules of games in physical environments through observation of game play. 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. These projects are not focused on accumulative, continuous online learning (D5) or learning representational structures to increase the generality (D2). Although video demonstrations are fairly accessible (D4) and enable competent game play (D1), they can require large numbers of demonstrations, including labeled illegal game play, reducing the efficiency of communication (D3).

5. Conclusions

In this paper we have described Rosie$_{TAG}$, an agent that learns new tasks through interactive situated instruction with a human mentor. The system's strengths include its ability to learn many different tasks, learn different types of knowledge that transfer between tasks (including new spatial relations, object features, actions, and game concepts), and interactively request information when novel concepts are introduced. We evaluated Rosie along a variety of dimensions, comparing its performance to other approaches, while also pointing out its weaknesses and areas for future research.

The major weaknesses of the current system are its restrictive, formulaic instruction syntax, its inability to scale to larger games, and its limited problem-solving capabilities. Addressing

these concerns raises many goals for future work. These include improving the accessibility of communication by expanding the types and flexibility of interactions and extending the natural language processing and grounding system. We also plan to develop mechanisms to reduce the search space and redundant calculations. These could include additional types of mentor interactions, such as advice on task-specific heuristics that reduce the search space. For example, one might state *in Tower of Hanoi never move the same object twice in a row*. Finally, one of the most important areas for future research is to improve Rosie's strategic competence for solving diverse tasks in complex environments.

To date, we have only scratched the surface in terms of the challenges in creating general taskable agents and the potential for them to change the way we develop and interact with AI systems. Closer to home, taskability has the potential to be the next *grand challenge for cognitive systems*. Not only does it require the complete set of cognitive capabilities we associate with intelligent behavior, but it also involves those capabilities that distinguish human-level behavior from other animals. It has the potential of uniting cognitive systems research with common goals and exciting problems.

Acknowledgments

The work described here was supported in part by the Office of Naval Research under Grant Number N00014-13-1-0217. 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 ONR or the U.S. Government.

References

Allen, J., Chambers, N., Ferguson, G., Galescu, L., Jung, H., Swift, M., & Taysom, W. (2007). PLOW: A collaborative task learning agent. *Proceedings of the Twenty-Second Conference on Artificial Intelligence*. Vancouver, Canada: AAAI Press.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Barbu, A., Narayanaswamy, S., & Siskind, J. M. (2010). Learning physically-instantiated game play through visual observation. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1879–1886). Anchorage: IEEE Press.

Björnsson, Y., & Finnsson, H. (2009). Cadiaplayer: A simulation-based general game player. *Proceedings of IEEE Transactions on Computational Intelligence and AI in Games*, *1*, 4–15.

Buro, M. (1999). How machines have learned to play Othello. *IEEE Intelligent Systems Journal*. *14*, 12–14.

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 ACM/IEEE International Conference on Human-Robot Interaction* (pp. 471–478). Boston: ACM Press.

Cohen, M. (2008). *A theory-based environment for creating reusable cognitive models*. Dissertation, College of Information Sciences and Technology, Pennsylvania State University, University Park, PA.

Epstein, S. L. (2001). Learning to play expertly: A tutorial on Hoyle. In J. Furnkranz and M. Kubat (Eds.), *Machines That Learn to Play Games*, 153–178. Huntington, NY: Nova Science.

Ernst, G., & Newell, A. (1969). *GPS: A case study in generality and problem solving*. New York: Academic Press.

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

Hinrich, T., & Forbus, K. (2013). X goes first: Teaching simple games through multimodal interaction. *Proceedings of the Second Conference on Advances in Cognitive Systems*. Baltimore, MD: Cognitive Systems Foundation.

Jones, R. M., Crossman, J. A., Lebiere, C., & Best, B. J. (2006). An abstract language for cognitive modeling. *Proceedings of the Seventh International Conference on Cognitive Modeling* (pp. 160–165). Trieste, Italy.

Kaiser, Ł. (2012). Learning games from videos guided by descriptive complexity. *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence* (pp. 963–970). Toronto: AAAI Press.

Laird, J. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Langley, P., Cummings, K., & Shapiro, D. (2004). Hierarchical skills and cognitive architectures. *Proceedings of the Twenty-sixth Annual Conference of the Cognitive Science Society* (pp. 779–784). Chicago, IL.

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

Langley, P., Trivedi, N., & Banister, M. (2010). A command language for taskable virtual agents. *Proceedings of the Sixth Conference on Artificial Intelligence and Interactive Digital Entertainment*. Stanford: AAAI Press.

Mohan, S., Mininger, A., Kirk, J., & Laird, J. (2012). Acquiring grounded representation of words with situated interactive instruction. *Advances in Cognitive Systems, 2*, 113–130.

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

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal, 3*, 210–229.

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

Tesauro, G., & Sejnowski, T. J. (1989). A parallel network that learns to play backgammon. *Artificial Intelligence, 39*, 357–390.

Thielscher, M. (2011). General game playing in AI research and education. *Proceedings of the Thirty-fourth Annual German Conference on Artificial Intelligence* (pp. 26–37). Berlin, Germany: Springer.

Yost, G. (1993). Acquiring knowledge in Soar. *IEEE Expert, 8*, 26–34.