Advances in Cognitive Systems 5 (2016) Submitted 4/2016; published 6/2016

Interactively Learning Strategies for Handling References to Unseen or Unknown Objects

Aaron Mininger MININGER@UMICH.EDU
John Laird LAIRD@UMICH.EDU
Computing Science and Engineering, University of Michigan, Ann Arbor, MI 48103 USA

Abstract

In this paper we extend an interactive task learning agent to a mobile robot in a large, multi-room environment. In such an environment, the instructor may make verbal references to objects that are unseen or unknown. We describe a comprehensive approach to this problem that combines reference resolution, object finding, and anchoring. We also demonstrate how some of these capabilities can be extended through interactive instruction. The agent is able to learn and use strategies for finding objects that involve multiple sources of knowledge.

1. Introduction

The goal of research in Interactive Task Learning (ITL) (Laird, 2014) is to allow human instructors to teach intelligent agents new tasks and extend existing tasks in real-time, through interactions such as verbal instructions and demonstrations. In this paper, we extend an existing ITL tabletop agent, Rosie (Mohan & Laird, 2014), to a mobile robot that learns new tasks across a large, multi-room indoor environment. The robot drives around, interacts with simple objects, and communicates with people. It learns new tasks such as *'Deliver the package to the main office,' 'Tell Alice a message,'* and *'Fetch a stapler.'* To teach a task, the instructor describes the goal (*'The goal is that the package is in the main office'*) and, if necessary, actions that the agent should execute to achieve the goal. Once the goal is achieved, the agent learns a policy for the task.

The most significant difference between the tabletop and mobile domains is that in the tabletop domain, all task-relevant objects and locations are immediately perceivable by the agent. This greatly simplifies the problem of making a connection between any linguistic reference (such as *'the stapler'*) and a target object in the environment (the agent's perception of the physical stapler). In contrast, an instructor in the mobile domain can refer to objects that the agent cannot perceive or is not aware of. Often work in interactive task learning ignores this problem by having environments where all the objects are known (Saunders, Syrdal, Koay, Burke, & Dautenhahn, 2016; Mohseni-Kabir, Rich, Chernova, Sidner, & Miller, 2015).

This paper describes a comprehensive approach to connecting linguistic references to objects in a large, partially observable environment where the agent need not have prior knowledge about the referenced object. We decompose this process into three stages. The first stage is *reference resolution*, where the agent resolves the referring expression to an internal representation of the

target object to be used in the task. This stage occurs during the extraction of the semantic meaning of a sentence, using linguistic constructions (such as definite or indefinite determiners, anaphoric references, relative clauses, and prepositional phrases) to inform how a linguistic referent should be resolved. The agent first attempts to resolve it to an existing internal representation of a known object (which may or may not be currently perceived). If no suitable one exists, the agent creates a new internal representation that includes all the constraints implicit in the linguistic structure.

The second stage is *object finding*, which is necessary when the target object is not perceivable. The agent must take actions so that it is physically in a position where it perceives the target object. In some cases, it may not immediately know where to find the object. It may have to search long term memory, ask the instructor, or do an external search in the environment to find it.

The third stage is reference *anchoring*, where the agent connects the internal representation of the object reference to the agent's perception of the object. Note that this stage is unnecessary if the reference was resolved to an existing object representation that was already anchored. This step may be delayed until a suitable object is perceived. If a perceived object matches the internal representation of the reference, they are anchored together.

The object finding stage is the most unconstrained of the three. Some ways of finding objects are useful across a wide variety of tasks. For example, systematically searching a room or going to the last place the object was seen. However, some strategies for finding objects may be domain specific, such as looking in the lost and found or looking for an employee in her office. Similarly, the anchoring stage may fail because an unknown domain-specific concept is used in the reference. For example, when the instructor refers to a new person by name or uses a new term (such as *favorite* in *'Bring Bob his favorite drink'*). This domain specificity makes it difficult to impossible for an ITL agent to be pre-programmed with all the knowledge to find and anchor all objects across all tasks and domains. Our hypothesis is that interactive task learning should include the ability to learn new approaches to object finding and anchoring. Our contribution is implementing these three stages of connecting references to objects in an interactive task learning agent so that it can learn and perform tasks in a large, partially observable environment. A critical component is providing the capability for it to extend its object finding and anchoring capabilities through natural language instruction. As part of supporting object finding and anchoring, we have extended the set of possible instructions to include *internal actions* that manage the agent's knowledge and memory – sometimes proactively remembering information that will be needed later and sometimes deliberately retrieving information stored in its long term memories.

In the remainder of this paper, we describe the embodiment of our ITL agent in a mobile robot (Section 2). We then describe and categorize the different object-finding strategies that the agent can use, and how these can be learned through instruction (3). Finally, we describe how these capabilities extend the tasks our agent can learn (4) and evaluate through simulation the ability of our agent to find objects in several experiments (5).

2. Agent Overview

Our agent is implemented in the Soar cognitive architecture (Laird, 2012). Soar includes a procedural memory, encoded as rules, that holds knowledge for selecting and performing internal actions,

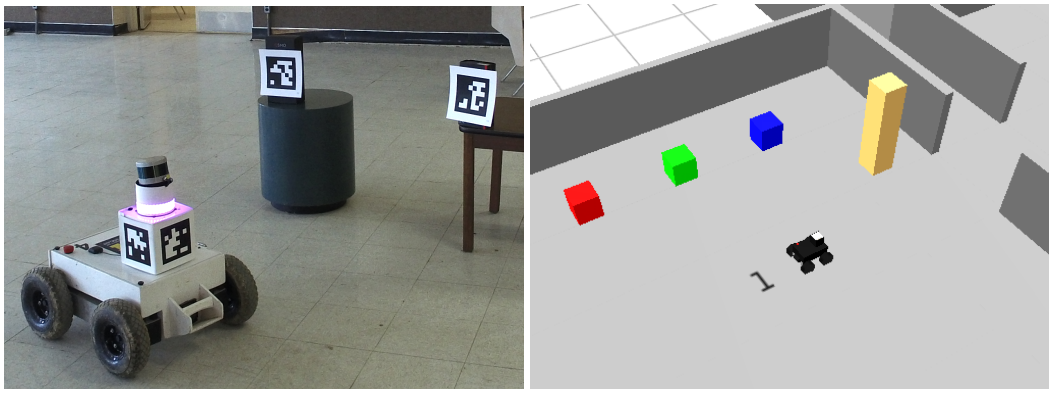

Figure 1. Photo of the mobile robot detecting objects with fiducials (left). Image of the simulated robot environment with objects (right).

external motor actions, and tasks. Procedural memory tests Soar's working memory, which contains representations of the agent's current goals, tasks, and belief about the world. In order for information to influence behavior, it must be in working memory. To maintain efficient execution, working memory only contains task-relevant information. Other long term data are held in Soar long term declarative memories: semantic memory contains facts about the world and episodic memory contains the history of the agent's working memory.

2.1 Perception and Action

The robot platform we use is a four-wheeled robot with a 360 laser range finder and a front-facing camera (Figure 1). It starts with a known metrical grid map overlayed with a set of regions (navigable rooms and hallways, also referred to as locations). The robot's position within the map and current location are reported to the agent. The agent starts with a topological map it can use to navigate between locations via `go-to-waypoint` commands. It can associate a linguistic label with a location, and these associations can be taught through instruction (e.g. *'You are in the kitchen'*). The robot uses visual fiducials to detect and classify objects. These objects are represented as a set of unary predicates (e.g. O_3: {`red(`O_3`), visible(`O_3`), grabbed(`O_3`)`}) and there are also relations defined over those objects (e.g. `in(`O_3`, `O_5`), holding(`O_2`, `O_1`)`). Objects in the agent's current location are maintained in working memory, as are objects relevant to the current task. When the agent leaves a location, all objects not involved in the current task are removed from working memory. Objects with long term importance, such as people and locations, are also permanently stored in semantic memory.

The agent starts with a set of simple actions it knows how to carry out. These include movement actions (`drive-to-location, turn`), manipulation actions (`pick-up, put-down`), and communicative actions (`say, ask`). (Currently we rely on a person to actually move objects on/off the robot). The agent's knowledge of the actions is encoded in Soar's procedural memory as rules. For each of these primitive actions, the agent knows when the action can be performed (its pre-conditions), how to execute the action (it motor actions), and the effects of the action (its action

model). The agent uses the action models during planning to find a sequence of actions that achieve a task's goal.

We also have a simulated environment which we use for development and testing. It allows us to do more extensive and controlled experiments with more trials. The perceptual and motor interfaces to the agent are identical to that in the real world. The simulated robot performs the same low-level motor controls and only perceives objects in its line of sight.

2.2 Task Learning

The agent learns new tasks through *situated interactive instruction* (Mohan & Laird, 2014). The instructor interacts with the agent through a chat interface. The agent parses the typed sentences to create a semantic representation using knowledge encoded in the procedural and semantic memories. The agent assumes it is learning a new task when it receives a command that uses a novel verb, such as *'**Deliver** the package to Alice'*. The agent creates a representation of the new task that includes the semantic structures of the arguments used in the command, and stores that representation in semantic memory. As the agent learns more about the task, it adds more to this structure.

Next, the agent asks for the goal of the task. The goal is specified by a set of predicates over objects that describe a desired state of the world. For example, the agent turns *'The goal is that Alice is holding the package'* into `holding(Alice, package)`. The agent then uses its internal models of its actions to search for a solution. If a solution is found, the agent executes it by selecting and applying the appropriate actions. If the agent is unable to solve the problem because it does not have sufficient knowledge to find a solution, the agent asks the instructor for advice. The instructor then suggests actions that the agent executes until it can solve the problem by itself or the goal is reached. Once the agent recognizes that it has achieved the goal, it learns a policy for selecting each action in appropriate situation so that it will be able to solve the problem in the future. Our agent can learn tasks where the goal is specified as a set of predicates, the known actions are sufficient for achieving the goal, and the internal action models are sufficient for simulating the goal achievement.

3. Connecting References to Objects

During interaction with the agent, the instructor will reference different objects. The agent must eventually connect those references to the perception of a suitable object. To do so, the agent needs to first do *reference resolution* to get an internal representation of the reference. Then, if the object is not being perceived it must *find* the object. Finally, it must *anchor* its perception of the object to the internal representation. These stages are discussed below.

3.1 Reference Resolution

First, the agent must take a linguistic reference and either identify an existing representation or create a new representation for the referenced object. Our approach is inspired by the *Givenness Hierarchy* (Gundel, Hedberg, & Zacharski, 1993), which describes how to associate the form of the reference expression (e.g., pronoun, definite noun phrase) with a cognitive status (e.g. in-focus, uniquely identifiable). We use this cognitive status to determine which candidate objects to consider.

Cognitive Status	**Linguistic form**	**Candidate Set**
In Focus	*it*	DL
Activated	*this, that, this NP*	ACT
Familiar	*that NP*	*not used*
Uniquely Identifiable	*the NP*	DL > VIS > STM > LTM > NEW
Referential	*indef. this NP*	*not used*
Type Identifiable	*a NP*	NEW

Figure 2. Candidate object sets considered during reference resolution based on the linguistic form of the referring expression: previous dialog (DL), activated (ACT), visible (VIS), short term memory (STM), long term memory (LTM), and a newly created representation (NEW).

In our system, we handle four of the cognitive statuses (Figure 2). The *in focus* status is indicated by the word *it* and indicates an object in the current center of attention. The candidate set is the set of objects previously mentioned in the dialog within the current task (DL), with a preference for more recent objects that match usage constraints (for example, if 'it' is used as the direct object of 'move', the referent must be an object and not a location). The *activated* status is indicated by the word *this*, and indicates an object that has some recent salience. Currently, this is only used when the instructor 'points' to an object (clicks on a graphical interface). The *uniquely identifiable* status is indicated by a noun phrase with the word *the*, and is the most common form that our agent encounters. This status is also assigned by default if no form is given. Here the instructor is indicating that he expects that the agent can resolve the reference to a unique object based on the constraints in the reference. Here, the agent looks through the following candidate sets in order until it finds an object that satisfies all the constraints: objects referenced in previous dialog (DL), currently visible objects (VIS), all objects in short term memory (STM), and all objects in long term memory (LTM). If no match is found, it generates a NEW representation containing the constraints in the referring expression. Finally, the *type identifiable* status is indicated by a noun phrase with the word *a*, and indicates that the instructor is referring to a class of objects, not a specific one. In this case the agent generates a NEW representation that includes the constraints in the reference. The end result of this process is that the reference has been resolved to an internal representation, either existing or newly created. If more than one object matches, the agent engages in additional interactions to determine which is intended (e.g. *'Which red block?'*, *'The large one'*).

3.2 Object Finding

Once an object reference has been resolved to an internal representation, the agent can use that representation to continue with the task even if it does not perceive the object. For example, the object representation can be used to generate a goal or to do internal planning. At some later point, the agent may need to actually perform an action involving that object, such as picking it up. To do so, the agent must find the object in the world. This may require taking actions like turning to face the object or searching through multiple rooms. Thus the agent needs different strategies for finding an object that depend on the agent's knowledge about that object.

We categorize the different strategies according to where they get knowledge about the object. Some strategies utilize internal sources of knowledge: *short term memory* and *long term memory*. Others get knowledge from external sources, through *interaction* or *external search*. We describe each of these four categories of strategies and what actions our agent can perform to handle each case. These strategies are implemented within the existing task learning and execution framework (Mohan & Laird, 2014). A significant extension has been adding actions that access knowledge in long term memories (`think` and `recall`) and allowing the instructor to use these actions in instructions.

3.2.1 Short Term Memory

There may already be knowledge in short term memory about the location of the object. This knowledge could come from previous experience with the object, information provided by the instructor, or from a previous retrieval from long term memory. Whatever its source, the agent can immediately act on this information to try and find the object. In our system, the agent may have the object's position stored in its spatial memory. In this case, it can execute a `face` command to turn towards where it thinks the object is. Alternatively, the agent could have knowledge that the object is in a different location. In this case, it can perform a `go-to-location` action to drive there.

3.2.2 Long Term Memory

The agent might have knowledge about where to look for the object stored in its long term memory. For example, the object may have been encountered in the past and subsequently removed from working memory (usually when leaving a room). This knowledge will have been automatically stored in episodic memory, and the agent will need to retrieve it into working memory for it to be useful. To perform this strategy we created the `recall` action. Any arguments of the recall action form a description of a hypothetical state of the world that is then used to access episodic memory. For example, suppose the agent is trying to find Alice. The instructor can say *'Recall Alice in a location.'* The agent then searches episodic memory for an episode with `{Alice(A), location(B), in(A, B)}` in the world state. If such an episode exists, the location of Alice (`B`) is added to working memory along with `in(Alice, B)`. This strategy is equivalent to asking where the agent last saw Alice. The agent can then search in that location.

There might also be knowledge about the object in semantic memory that would be useful when deciding where to look for an object. For example, the agent could know that sodas are stored in the kitchen, or that Bob might be in his office. Although the agent could search semantic memory by generating many different cues that could potentially be relevant, that approach is computationally expensive and would degrades the agent's responsiveness. Moreover, the instructor may know specific cues to use to find the most relevant knowledge in the current situation. To support this type of instruction, we created the `think` action, which is used to retrieve facts from semantic memory. For example, *'Think of the office of Bob.'* Here the agent looks for a predicate `office(Bob, X)` in semantic memory. If such a predicate with object X is found, that object is added to working memory as well as the hypothesis `in(Bob, X)`. The instructor can also add this information to semantic memory by giving instructions such as *'The office of Alice is the main office'*.

3.2.3 Interaction

If the agent has no knowledge about where to find the object, it can ask the instructor for help, for example: *'Where is the red box?'* The instructor then has a number of options to help the agent find the object. The instructor can describe the location of the object (*'The red box is in the conference room'*), tell the agent where to go (*'Go to the main office'* or *'Turn around.'*), go get the object and put it in front of the robot (*'Here it is'*), or decide not to provide any help (*'I don't know'*).

3.2.4 External Search

Finally, the agent can try to find the object through external search. This search can involve searching within the current location (local) or the entire environment (global). For this, we have added two actions: `scan` and `explore`. The scan action involves slowly turning in a complete circle to look around the room. The explore action involves driving to all known locations. These actions need to be terminated when the object is perceived, so the instructor should give the command with an until clause. For example: *'Explore until you see the soda.'*

3.3 Find Object Subtask

In addition to knowing how to find an object, it is important that the agent knows *when* to find an object. The agent is often able to plan and reason using non-anchored representations of objects, and it should not try to find an object until it becomes necessary in order to make progress with the task. For example, when delivering a package to Bob, the agent should pick up the package before trying to find Bob. In order to support a general approach to selectively finding objects in appropriate situations, we created the `find-object` subtask that is proposed for each object in working memory that is not visible. The goal (and action model) of this task is that the object becomes visible. During planning, the agent includes the `find-object` subtask as one of the possible actions it can use for solving a task. Since most actions require that the objects involved to be visible, the agent will discover during planning that it needs to include the find-object action in its plan. For example, in the task *'Deliver the package to Bob'*, once it has picked up the package, the agent must give it to Bob. Since a precondition for `give(package, Bob)` is `visible(Bob)`, if Bob is not visible the agent will include the action `find(Bob)` in the plan.

In the find-object subtask, the agent can then try to execute the different strategies described above. If none are successful, the agent initiates a new interaction to ask for help. It will say something such as *'I can't find the soda. Can you help?'* The instructor can then provide information (e.g. *'The soda is in the kitchen'*) or teach a new strategy in the form of a command (e.g. *'Think of the storage location of sodas'*). For the latter, the agent will then learn a rule to propose that action in the find-object subtask. A benefit of having a separate subtask for finding objects is that strategies learned during one task can be used in other tasks. If the agent has to choose between multiple strategies, it orders them based on the source of knowledge, from most specific and efficient to most general: short term memory, long term memory, interaction, and finally, external search.

3.4 Anchoring

Once an object is perceived, the agent must connect its perception to the internal representation of the object reference. This process is called *anchoring* (Coradeschi & Saffiotti, 2003). New object perceptions are matched against non-anchored objects in the agent's working memory that contain all of its predicates. Thus if the agent is looking for an object and one matching its description is perceived, the anchoring step is done immediately. However, sometimes the object is perceived but not anchored to the internal representation because the reference contains a name or visual attribute that is not familiar to the agent. This is not a case we currently handle, except for allowing the instructor to teach the name of the current location by saying *'You are in the kitchen'*.

In other cases, the agent is unable to anchor the representation because the instructor referred to the object using a new concept the agent does not yet know how to anchor. For example, in the task *'Serve Bob'*, the agent should ask *'What drink would you like?'* and Bob can answer *'The soda'*. The instructor can refer to this answer as the *'desired drink'* (as in *'The goal is that Bob is holding the desired drink'*), but the agent may not know how to anchor this reference because it does not know the concept of `desired`. To have the agent learn this, we created the `remember` action. This action takes the form `remember(A, B)`, where `A` is a known object and `B` is the object the agent can't anchor. The result is that the two objects are merged (all references to `B` are changed to `A`). In the current example, once Bob has answered *'the soda'* the instructor says *'remember the answer as the desired drink.'* The effect of this action is that the known soda object (which is anchored and has the `answer` predicate added by the `ask` action) is merged with the representation of the desired drink. The agent now knows what object in the world corresponds to the phrase *desired drink* and can then pick up the soda and give it to Bob. Note that the agent does not need any previous knowledge about what the concept of *desired drink* means in order to learn the task. Through instruction it learns to associate the person's answer with the concept `desired`.

4. Extending Interactive Task Learning

A major contribution of this work is extending our previous task learning agent to function in partially observable environments and use internal actions that deliberately access and modify the agent's memories. Previous work involved the agent in a table-top environment where the entire workspace was always visible. Extending reference resolution to access objects in semantic memory means that the agent can learn tasks involving objects that are known but not immediately present, and implementing the ability to create and use hypothetical representations of objects means the agent can learn tasks involving objects the agent has not previously encountered. Tasks such as *'Deliver the package to Alice'* or *'Fetch a stapler'* are now possible, even if the objects involved are not visible or even known. Figure 3 shows some examples of the kinds of tasks that our agent can now learn and perform. Our agent is able to learn and execute all the tasks on the real-world robot. Below, we describe three tasks in detail. The interactions for these tasks are shown in Figure 4.

Task Command	**Task Goal**
Discard the package	The package is in the trash
Deliver the package to Alice	Alice is holding the package
Fetch a stapler	The stapler is in the starting location
Fetch a stapler from the office	The stapler is in the starting location
Take the stapler to the main office	The stapler is in the main office
Tell Charlie a message	Charlie heard the message
Serve Bob	Bob is holding the desired drink
Guide Bob	You are in the desired location
Greet Alice	Alice heard the welcome message

Figure 3. Examples of tasks and their goals that our agent can learn through instruction.

Deliver the package to Alice.	**Fetch a stapler.**	**Guide Bob.**
What is the goal? The goal is that Alice is holding the package. (Rosie picks up the package) *I cannot find Alice. Can you help?* Think of an office of Alice. (Rosie goes to her office and gives her the package)	*What is the goal?* The goal is that the stapler is in the starting location. *What do I do next?* Remember the current location as the starting location. *I cannot find the stapler. Can you help?* Recall the stapler in a location. (Rosie goes to the location and brings the stapler back)	*What is the goal?* The goal is that you are in the desired location. *What do I do next?* Ask 'Where do you want to go?' *Where do you want to go?* The kitchen. *What do I do next?* Remember the answer as the desired location. (Rosie goes to the kitchen)

Figure 4. Examples of teaching three tasks. Rosie's responses are in italics

4.1 Deliver

Here we go through a detailed example of how the agent learns to deliver an object to a person. The instructions for this task can be seen in Figure 4. The instructor gives the command *'Deliver the package to Alice'*. We assume that the package is already visible and represented in working memory, while Alice is not. An object will be denoted by a symbol O_i along with the set of unary predicates that define it. For example, the package is O_1`:{object, package, visible}`.

The agent parses the command and identifies two noun phrases to try and resolve: *'the package'* and *'Alice'*. The reference *'the package'* is given the cognitive status *uniquely identifiable*. Since this is the first reference to the package, no match is found in the dialog list. However, the object O_1 is visible and satisfies the `package` constraint, so the reference is resolved to that object. The second reference *'Alice'* is given the cognitive status *uniquely identifiable*, but no match is found in working memory. The agent constructs a query to semantic memory for some object `X` with the predicate `Alice(X)`. It finds a match with object O_2 which it adds with all its unary predicates to

working memory: O_2:{person, Alice}. Once the reference resolution is complete, the agent has a representation of the action:

deliver(O_1, to(O_2)) *'Deliver the package to Alice.'*
O_1:{object, package, visible} O_2:{person, Alice}

The agent then tries to execute the action, but reaches an impasse because it has no goal representation. It asks *'What is the goal?'*. The instructor responds *'The goal is that Alice is holding the package'* Now, during reference resolution both references can be resolved to objects in the previous dialog, and the following representation of the goal is generated: holding(O_2, O_1). Note that the agent stores generalized representations of the action and goal in semantic memory, where specific objects are replaced with variables (e.g. deliver(A, to(B)) goal=holding(B, A)). Now that it knows the goal, it tries to find a sequence of actions that can achieve the goal. In this task, it finds the following sequence of actions:

Action	**Preconditions**	**Postconditions**
pick-up(O_1)	!grabbed(O_1) visible(O_1)	+grabbed(O_1)
find(O_2)	!visible(O_2)	+visible(O_2)
give(O_1, O_2)	visible(O_2) grabbed(O_1)	+holding(O_2, O_1) -grabbed(O_1)

Figure 5. Actions needed to deliver the package (O_1) to Alice (O_2). The goal is holding(O_2, O_1).

The agent picks up the package, then tries to find Alice. It fails and asks for help: *'I cannot find Alice, can you help?'* The instructor says *'Think of an office of Alice'*. During reference resolution, the reference *'Alice'* is connected to O_2. The reference *'an office'* is given the *type identifiable* status and a new representation O_3 is created for it. The final action is:

think(O_3, of(O_2)) *'Think of an office of Alice'*
O_3:{office} (new) O_2:{person, Alice}

The agent carries out the think action by looking for the predicate office(O_2, X) in semantic memory. It finds the predicate office(O_2, O_4) and adds O_4 (Alice's office) along with its predicates to working memory. It also adds the belief in(O_2, O_4) to working memory. This belief causes the agent to drive to her office to look there. It arrives at the office and the new perception of Alice is anchored to O_2. The predicate visible(O_2) is added and the find(O_2) subtask is finished. The agent also uses this instruction to learn a new object finding strategy. It learns a rule that when it is trying to find an object A, it should propose the action think(B, of(A)) with office(B). This strategy of looking in an office can be used to find objects or people in other tasks. Note there is nothing special about the label office. The agent does not need to know ahead of time how it will be used. The instructor can just have easily used the terms

classroom, cubicle, or storage as long as they match a predicate in semantic memory (which can be added through other interactions).

Once the agent has found Alice, it will give her the package and complete the task. Then, the agent does a retrospective analysis (Mohan & Laird, 2014) to learn a policy for selecting each of the operators in the solution in appropriate states. For example, it learns to prefer finding the person once the object being delivered is grabbed.

4.2 Fetch

The agent can also be taught to fetch an object and bring it to the agent's original location (Figure 4). The instructor starts by giving the command *'Fetch a stapler'*. We assume that there is no stapler in the immediate environment, and the robot's current location is the kitchen (O_1). In the command is a single reference *'a stapler'*. It is given the *type identifiable* status, so a new representation is created and added to working memory: O_2:stapler(O_2). The action becomes:

fetch(O_2) *'Fetch a stapler'*
O_2:{stapler} (new)

The goal is given through the statement *'The goal is that the stapler is in the starting location'*. The reference *'the stapler'* is resolved to the object O_2 because it is in the previous dialog. The reference *'the starting location'* fails to match anything, so a new representation is created. The resultant goal is {in(O_2, X), starting(X), location(X)}. The agent cannot find a way to achieve this, because no known actions add the starting predicate. The agent can be taught to mark where it starts with the starting predicate through the remember action:

remember(O_1, as(O_3)) *'Remember the current location as the starting location'*
O_1:{location, kitchen, current-loc, visible}
O_3:{location, starting}

The effect of this action is to merge the objects, replacing all references to O_3 with O_1. This changes the starting predicate to starting(O_1). In addition, a rule is learned to propose this remember action inside the fetch task. Now that there is a starting predicate in working memory, the agent can find a sequence of actions to reach the goal (Figure 6).

Action	Preconditions	Postconditions
find(O_2)	!visible(O_2)	+visible(O_2)
pick-up(O_2)	!grabbed(O_2) visible(O_2)	+grabbed(O_2)
go-to(O_1)	!current(O_1)	+current(O_1)
put-down(O_2, O_1)	grabbed(O_2) current(O_1)	-grabbed(O_1) +in(O_2, O_1)

Figure 6. Actions needed to fetch the stapler (O_2) and bring it back to the kitchen location (O_1). The goal is in(O_2, O_1), starting(O_1), location(O_1)

First, the agent needs to find the stapler. It saw one before in a different location, and that knowledge is in episodic memory. The instructor teaches it to retrieve this information through the following command:

`recall(O2, in(O4))` *'Recall the stapler in a location'*
`O2:{stapler}` `O4:{location}` (new)

The agent carries out the `recall` action by constructing an episodic memory query cue from the predicates in the command: `{stapler(X), location(Y), in(X, Y)}`. It retrieves an episode from episodic memory which contains all the predicates. In it, the lab location (`O5`) matches the variable `Y`, so it and all its predicates are added to working memory as the object `O5:{location, lab}`. It also adds the belief `in(O2, O5)` to working memory. This belief causes the agent to drive to the lab to look for the stapler. Once there, it scans the room and perceives the stapler. This perception is anchored to `O2` and the predicate `visible(O2)` is added. If the stapler was not in the recalled location, the `in` predicate would be removed and the agent would do something else to find the object. This `recall` strategy is learned within the find subtask, so in the future it can be used within different tasks to find different objects. It learns a rule that when it is trying to find an object `A`, it should propose the action `recall(B, in(A))` with `location(A)`. Once the stapler is found, the agent can go on to complete the rest of the task and learn the policy.

4.3 Guide

The `guide` task involves asking a person where he would like to go and then driving to that location. For example, the instructor says *'Guide Bob'*, where *'The goal is that you are in the desired location'*. However, the agent does not know how to anchor the reference *'desired location'* to an actual location in the world. The instructor teaches it to ask the question *'Where do you want to go?'*, to which Bob answers *'The kitchen.'* As part of the ask action, the agent adds an `answer` predicate for the response (in this case, `answer(kitchen)`). It then associates the answered location with the desired location via a remember action (*'remember the answer as the desired location'*). In a future execution of this task, the agent will perform the remember step automatically. This `remember` action allows the agent to learn tasks where some knowledge needs to be deliberately added to working memory to be used later. And the agent can learn a task involving a new concept (like `desired`) whose usage is determined by the instructor.

5. Experiments

Our evaluation focuses on whether the different find-object strategies transfer between different situations, thereby contributing to the agent's overall ability to successfully find objects independent of task. We also want to determine whether those strategies are as effective when taught instead of pre-encoded. We explore these questions by performing several experiments in a simulated environment. The simulator realistically simulates the low-level perception and motor controls of the physical robot in real time. While all of the tasks in the previous section (including find-object) can be done on the real robot, the simulated environment provides more control over reproducibility

and allows us to precisely control the placement and dynamics of objects in the environment. It also allows us to avoid spurious perceptual errors that have little to do with our research on task learning. In the experiments, we measure the ability of the agent to find different objects.

Our testing environment has eight rooms connected by four hallways, with thirty objects and eight people. This environment is large enough so that the agent usually has to search a few rooms during exploration before finding an object, and has enough objects so that each room has multiple objects in it. The agent starts with a complete map of this environment. Half of the objects are assigned to a storage location and half of the people to an office. This information is stored in the agent's semantic memory so that it has semantic information about some of the objects. The environment is dynamic, so that the agent's knowledge about an object is not always correct. Thus after each find-object request, each object and person has a 20% chance of being moved to a new random location.

5.1 Comparing Finding Strategies

Our first experiment examines how the different strategy categories contribute to the ability of the agent to find objects. We want to evaluate the benefit of having multiple strategies – when the agent has specific knowledge about where to look for an object, can it use it to quickly find the object, and if not, can the agent still find it using a more time-consuming, but still effective search? We grouped the strategies into three sets according to their categories. In the Short Term Memory (S) set, the agent knows the location of an object and must either *face* it or *drive* to its location. In the Long Term Memory (L) set, the location is not in short term memory and the agent must use the *think* action to retrieve an object's expected location (either office or storage location) from semantic memory, or the *recall* action to retrieve an object's previous location from episodic memory. The External Search (E) set is for when it does not have knowledge it can use to find the object. In this case, the agent either *scans* the room by doing a complete revolution or *explores* all the rooms until the object is found. We did not use the interaction strategy because the effectiveness of it depends on what the instructor does, not on the agent.

For each combination of strategy sets, we had the agent find 20 random objects. The environment was changed between each find-object request as described above. To reduce variance, the sequence of objects being found and the way the objects were moved was the same in each test. We measured the number of those find-object requests that the agent could satisfy, and measured the average number of rooms searched when it was successful.

For the first comparison, we ran the test only using the most specific strategy (S) first. This would only succeed when the agent already knew about the necessary object. We then added the more general strategies (L and then E), which should improve its ability to find the objects. The results are shown in Figure 7 on the left. The short term memory strategy (S) can utilize very specific and relevant knowledge, so when it had that knowledge, it could use it right away. However, the agent often did not have that knowledge so it failed 90% of the time. Adding in the long term memory strategy (S+L) meant the agent succeeded more often because it could use knowledge from more sources. And when all three kinds of strategies were used (S+L+E) the agent achieved 100% success because when it didn't have knowledge to rely on, it could do external search. However, these less specific strategies meant searching more rooms.

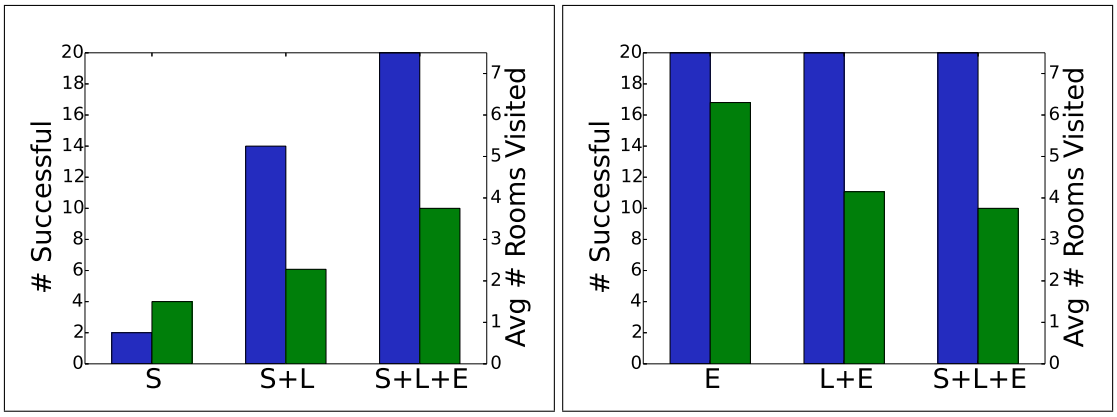

Figure 7. Results of 20 find-object tasks with different combinations of finding strategies. Short term memory (S) includes face and go-to-location. Long term memory (L) includes think and recall. External actions (E) includes scan and explore. Results include the number of successes (blue/left bar) and the average number of rooms searched when successful (green/right bar).

For the second comparison we ran the test on the most general strategy first (E), and then added the more specific strategies of L and then S. The results are shown in Figure 7 on the right. The external search strategy (E) always succeeded, but it required searching many rooms. As strategies that involved more knowledge were added, the agent found the object with much less search.

These results demonstrate how the agent took advantage of these different sources of knowledge to quickly find objects in the environment. With all the strategies (S+L+E), the agent successfully found objects 100% of the time, and did so much more quickly than when using random search by taking advantage of relevant knowledge.

5.2 Strategy Learning

Our second experiment compared the ability of the agent to find objects when the strategies were taught though instruction instead of pre-encoded. Here the agent started only with the ability to ask for help and use knowledge in short term memory. We taught the agent five strategies – three using long term memory (L) and two using external search (E). This happened in five teaching interactions, one for each strategy, where the instructor gave the agent a *find-object* task that required that strategy in order to succeed. We taught the strategies in the following order: `think(office)`, `think(storage)`, `recall(location)`, `scan`, and `explore`. After these five training interactions, we had the agent find the same 20 objects as in the previous experiment with no further teaching. The results were then compared to the performance of the agent with the full set of hand-coded strategies (S+L+E) from the first evaluation. The agent achieved 100% success when the strategies were learned, the same as when they were pre-encoded. This occurred without any additional instruction, showing that the agent successfully transferred strategies from one find-object subtask to another. In addition, the agent visited an average of 3.6 rooms per task with the taught strategies; roughly equivalent to the 3.75 rooms per task for the hand-coded strategies. This shows that the agent used its knowledge to quickly find objects just as effectively with taught strategies.

A major strength of our approach to doing ITL within a cognitive architecture is that the agent learns representations native to the architecture. In our agent, strategies are encoded as proposal rules for the actions in the find-object subtask. When we code these by hand, as in the first experiment, only a single rule is required. When our agent learns these strategies through instruction, it first generates a declarative representation, but then learns via chunking a single rule to propose the action, so that the learned representation is similar to what an expert would encode.

6. Conclusion

In this work we have combined capabilities in *reference resolution*, *object finding*, and *anchoring* in an existing interactive task learning framework to enable task learning and execution in a large, partially observable, open-world environment. These capabilities allow the agent to do tasks with unseen and unknown objects. Furthermore, we have shown how the object finding and anchoring capabilities can be extended through instruction.

A major contribution of this work is the integration of these capabilities in a end-to-end task learning agent. Much of the related work involves a subset of these capabilities or exist in a simpler environment. Williams, Acharya, Schreitter, & Scheutz (2016) implement a more sophisticated reference resolution scheme in an open world environment, but do not worry about how to find unseen objects that were referenced. Work by Lemaignan, Ros, Sisbot, Alami, & Beetz (2012) and Skočaj et al. (2016) also involves connecting references to objects in a situated discourse, but do so in a limited workspace.

Other researchers have explored using knowledge when finding new objects in new environments. Samadi, Kollar, & Veloso (2012) present an agent that can use information from the web to identify the likely locations of new objects in an indoor environment. Aydemir, Pronobis, Gobelbecker, & Jensfelt (2013) make visual search more effective by taking advantage of background knowledge to focus the search on more likely places. Joho, Senk, & Burgard (2011) describe a way to improve its search technique in a new environment by using experience gained previously from similar environments. While these examples can learn within a strategy, they do not involve the ability to learn new strategies. On the other hand, our learned strategies are fairly simple. Those involving long term memory are limited; the agent could not learn a strategy involving a more complicated use of semantic or episodic memory. Also, the instructor requires significant knowledge of the internal representations of the agent and how to give the instructions in order to properly teach it these strategies. However, this work represents a crucial step in extending interactive task learning to a mobile robot in a large, indoor environment. Future work will involve extending the kinds of tasks the agent can learn and improving the ways the agent can use its internal memories during tasks. In addition, there is work to be done on making the language interactions more flexible and natural, with the end goal to have non-expert users be able to teach the agent new tasks with minimal training. Finally, we would like to improve the perceptual capabilities of the agent with real object recognition and handling of perceptual noise and errors.

Acknowledgements

The work described here was supported by the National Science Foundation under Grant 1419590 and the Office of Naval Research under Grant N00014-08-1-0099. 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, ONR, or the U.S. Government.

References

Aydemir, A., Pronobis, A., Gobelbecker, M., & Jensfelt, P. (2013). Active visual object search in unknown environments using uncertain semantics. *Robotics, IEEE Transactions on*, *29*, 986–1002.

Coradeschi, S., & Saffiotti, A. (2003). An introduction to the anchoring problem. *Robotics and Autonomous Systems*, *43*, 85–96.

Gundel, J. K., Hedberg, N., & Zacharski, R. (1993). Cognitive status and the form of referring expressions in discourse. *Language*, (pp. 274–307).

Joho, D., Senk, M., & Burgard, W. (2011). Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems*, *59*, 319–328.

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).

Lemaignan, S., Ros, R., Sisbot, E. A., Alami, R., & Beetz, M. (2012). Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction. *International Journal of Social Robotics*, *4*, 181–199.

Mohan, S., & Laird, J. E. (2014). Learning goal-oriented hierarchical tasks from situated interactive instruction. *AAAI* (pp. 387–394).

Mohseni-Kabir, A., Rich, C., Chernova, S., Sidner, C. L., & Miller, D. (2015). Interactive hierarchical task learning from a single demonstration. *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction* (pp. 205–212). ACM.

Samadi, M., Kollar, T., & Veloso, M. (2012). Using the web to interactively learn to find objects. *AAAI*.

Saunders, J., Syrdal, D. S., Koay, K. L., Burke, N., & Dautenhahn, K. (2016). "teach me-show me"–end-user personalization of a smart home and companion robot. *IEEE Transactions on Human-Machine Systems*, *46*, 27–40.

Skočaj, D., et al. (2016). An integrated system for interactive continuous learning of categorical knowledge. *Journal of Experimental & Theoretical Artificial Intelligence*, (pp. 1–26).

Williams, T., Acharya, S., Schreitter, S., & Scheutz, M. (2016). Situated open world reference resolution for human-robot dialogue. *ACM/IEEE Conference on Human-Robot Interaction*, *11*.