Technical Report, July 2017
Computer Science Department
Worcester Polytechnic Institute

# A Unified Theory of Learning, Doing and Teaching Hierarchical Tasks: The Learn-Do-Teach Challenge for Collaborative Agents

**Charles Rich and Candace L. Sidner**

*An Unfunded Research Proposal*

RI: SMALL:

# A Unified Theory of Learning, Doing and Teaching Hierarchical Tasks: The Learn-Do-Teach Challenge for Collaborative Agents

**Overview:** This research will result in new fundamental computational principles and algorithms enabling autonomous agents and robots to collaborate with humans and each other in a broad range of situations in home, commercial and hazardous environments.

The immediate objective of this project is to develop a unified theory of *learning*, *doing* and *teaching* complex hierarchical tasks, such as preparing a meal, rotating the tires on a car, or shutting down a power system. Hierarchy is important because humans deal with complex tasks by breaking them down into subtasks. A unified theory is needed in order to build collaborative agents and robots that can, like human collaborators, dynamically and seamlessly shift between learning, doing and teaching such tasks—or parts of them—as the situation demands. For example, consider a disaster response situation, in which a human located at a safe distance teaches a robot how to safely shut down a power system. In order to complete its mission, the robot might later need the assistance of people inside the hazardous area and will therefore need to teach them how to perform parts of its assigned task.

In a unified theory, the same representations and processes are used for learning, doing and teaching. We use *hierarchical task networks* (HTNs) as the core representation of complex tasks and apply *reflection* and *theory of mind* to model pedagogical goals and strategies as meta-tasks and meta-plans.

Since humans will interact with our agents as both teachers and learners, the task representation and algorithms in our theory will be informed by current research on human pedagogy. Both computational simulations and human studies will be used to evaluate the theory.

**Keywords:** collaborative agents; intelligent tutoring; learning from demonstration; learning from instructions; hierarchical task networks; theory of mind

**Intellectual Merit:** This work will transform the landscape of research on collaborative agents and robots, which is currently divided into disconnected subfields, such as *learning from demonstration*, *learning from instructions*, and *intelligent tutoring systems*, each of which looks at learning, doing and teaching in isolation. Because current approaches in each of these subfields are mostly incompatible, they cannot easily be combined to achieve the type of flexible collaboration described above. This research will provide the needed unifying theory, starting with HTNs. This work will also contribute to the fundamental computational understanding of how humans learn and teach procedural knowledge.

The investigators are well qualified for this work, as they have made significant contributions to, and thus understand the problems in, all three of the subfields mentioned above. They have also already implemented a proof of concept system for learning, using and teaching HTNs.

**Broader Impacts:** Autonomous agents and robots that interact with humans in diverse environments are the next step in the evolution of automation technology. Although such agents and robots will have some basic capabilities builtin, their task repertoire will also need to be extended "in the field" via collaboration with humans or other agents. Basic research to make this possible will give the U.S. a competitive advantage in the new industries arising from this technology. Also, by developing computational models of human pedagogical theories, this project can contribute to improving human pedagogy.

A substantial part of the research described here will directly contribute to the training of graduate and undergraduate engineering students, including women and underrepresented minorities. The results will be widely disseminated through scientific publication and open-source software.

Visibility and interest in a unified theory of hierarchical tasks within the broader research community will be raised via a novel and engaging research competition, called the *Learn-Do-Teach Challenge*, in which an autonomous software agent learns a task from a human, does the task, and then teaches it to another human and to another copy of itself.

RI: SMALL:

# A Unified Theory of Learning, Doing and Teaching Hierarchical Tasks: The Learn-Do-Teach Challenge for Collaborative Agents

## 1 Introduction

*What does it mean to "know how to do" something, such as preparing a meal, rotating the tires on a car, or shutting down a power system?* For a human, this usually means being able not only to perform the task, but also to teach it to someone else, and often that the task was learned from someone else in the first place. For machines, however, this is generally not true. Machines have long been able to perform complex tasks based on their programming without being able either to learn such tasks from humans or teach them to humans. Advances have been made in machines that learn, e.g., machine learning algorithms, and machines that teach, e.g., intelligent tutoring systems. However, these research subfields have for the most part been isolated from each other and have used incompatible approaches. The overall, long-term goal of this work is to increase the synergy between these separate research threads for both scientific and practical benefits.

The immediate objective of this project is to develop a unified theory of *learning*, *doing* and *teaching* complex hierarchical tasks. Hierarchy is important because humans deal with complex tasks by breaking them down into subtasks. Figure 1 shows what an agent based on a unified theory should be able to do: (a) learn a new task from a human, (b) do the task, and then teach it to (c) another copy of itself (i.e., based on the same theory) and (d) another human. For details on (e) and (f), see Section 4.4.

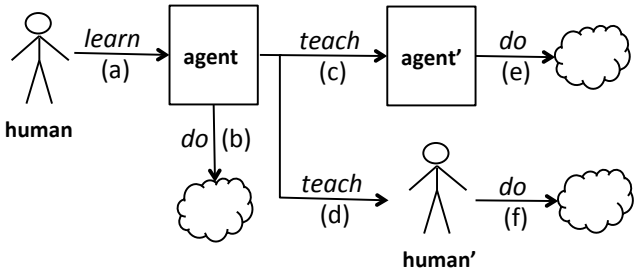

Figure 1: The Learn-Do-Teach Challenge.

The foundation of this unification is to view teaching and learning as kinds of *collaboration*, i.e., to model the teacher and learner as two participants in a collaborative interaction in which the shared goal is increase the learner's abilities.

The practical motivation for a unified theory is in order to build collaborative agents and robots that can, like human collaborators, dynamically and seamlessly shift between learning, doing and teaching such tasks—or parts of them—as the situation demands. For example, consider a disaster response situation, in which an expert human located at a safe distance teaches a robot how to safely shut down a power system. In order to complete its mission, the robot might later need the assistance of workers inside the emergency area and will therefore need to teach them how to perform parts of its assigned task. Or imagine a domestic robot being replaced by a newer model from another company teaching the new robot what it learned from the householder.

Since the focus of this research is on cognitive rather than physical abilities, it applies to both software agents and robots. For simplicity, in the remainder of this document we will use the term *agent* to refer to both autonomous software agents and robots.

Since humans will interact with our agents as both teachers and learners, the task representation and algorithms in our theory will be informed by current research on human pedagogy. Our work will also contribute to the computational understanding of how humans learn and teach procedural knowledge.

We begin below by reviewing work by the PIs and others in three key related subfields: collaborative agents, intelligent tutoring systems, and learning from demonstration and instructions. Current approaches in each of these subfields are mostly incompatible and thus will not together achieve the goal of flexible collaboration discussed above without a new unified theory.

To concretely illustrate what we mean by learning, doing and teaching based on a unified theory, we have implemented a proof of concept system and present a sequence of interaction walkthroughs with this system. To evaluate our proposed work, we will use both computational simulations and human studies, including a novel and engaging research competition, called the *Learn-Do-Teach Challenge* (Section 4.4).

There is no Results from Prior NSF Support section below because neither PI has received NSF funding with a start date in the past five years.

# 2 Related Work

The inspiration for this project grew out of the investigators' research experience in the three closely related subfields discussed below. Each of these subfields contains theories and systems that either do complex hierarchical tasks collaboratively, learn and do them, or do and teach them, *but not all three*. We also discuss two recent research efforts that take a learn-do-teach approach with more limited task representations.

## 2.1 Collaborative Agents

*Collaboration* is a process in which two or more participants coordinate their actions toward achieving shared goals. In this proposal, as in most of our previous research, we focus on two-party collaborations. Starting in the early 1990s, we extended and implemented a computational model of collaborative interaction based on SharedPlans discourse theory [22, 23, 24, 35]. The main contribution of this theory was to elucidate how, in task-oriented collaborations, the structure of the dialogue is governed by the underlying hierarchical task structure—see Figure 5 for an example.

Our research has resulted in two general-purpose tools, Collagen [47] and its open-source successor, Disco [49], which have been used by the investigators and others to build more than a dozen collaborative software agents, including an air travel planning assistant [46], an email assistant [25], a VCR programming assistant [57], a power system operation assistant [52], a gas turbine engine operation tutor [14], a flight path planning assistant [11], a recycling resource allocation assistant, a software design tool assistant, a programmable thermostat helper [16], a mixed-initiative multimodal form filling assistant and a robot hosting system [58]. The proof of concept system described in Section 4.1 was implemented using Disco (see Section 3).

Ferguson and Allen [1, 17, 18] and Bohus and Rudnicky [5] have also had long-running research projects focused on building collaborative software agents using hierarchical task models. Similar concepts have been applied to human-robot collaboration by Breazeal [27] and the investigators [43, 48].

In the next two sections, we describe how we applied our foundational research on collaborative agents first to intelligent tutoring systems (where the agent teaches the human) and then to learning from demonstration and instructions (where the human teaches the agent).

## 2.2 Intelligent Tutoring Systems

The intelligent tutoring systems most related to this project are those concerned with teaching procedural knowledge. The archetype of such systems is STEVE [50], shown in Figure 2, an animated 3D software agent for teaching the complex operational and maintenance procedures associated with gas turbine engines on naval vessels. STEVE represented these procedures hierarchically and taught them in a shared virtual environment where both STEVE and the human student could perform and observe all actions. There was also communication between STEVE and the student to guide the learning process. However, this communication was not based on a general model of collaborative dialogue.

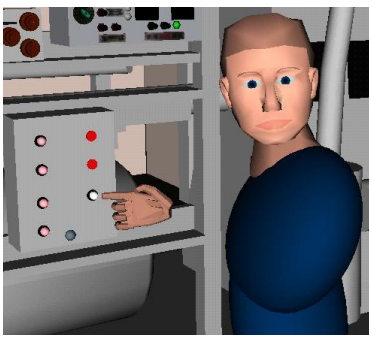

Figure 2: STEVE.

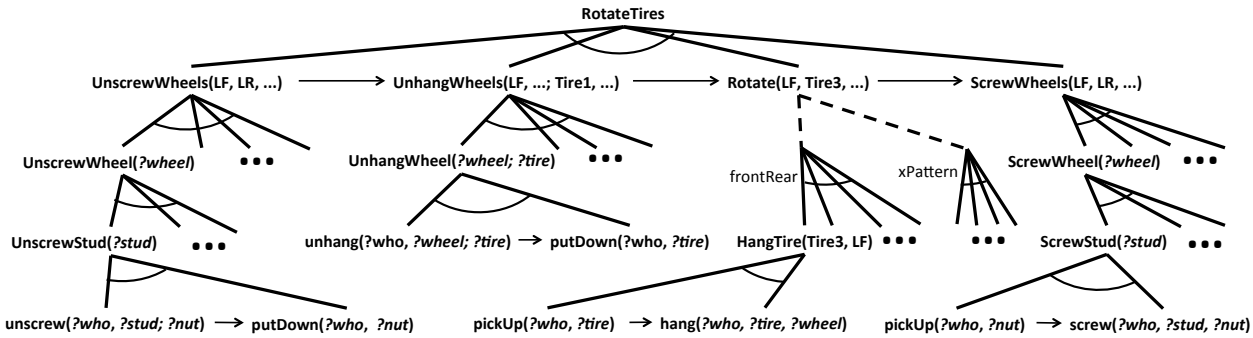


Figure 3: Hierarchical task model for tire rotation (to reduce clutter, most of the recipe names have been omitted and repetitive parts of the model have been replaced by ellipses).

Around 2000, the investigators began working with the main architect of STEVE, Jeff Rickel, with the goal of "Building a Bridge between Intelligent Tutoring and Collaborative Dialogue Systems" [51]. This led to a reimplementation of the cognitive parts of STEVE using Collagen [53] and to the central concept in the unified theory of this proposal, which is to reflectively model learning as a collaborative meta-task (as in Section 4.1.1). Other intelligent tutoring systems following in STEVE's footsteps include Lester et al.'s explanatory lifelike avatars [33] and the virtual training agents of Mendez and de Antonio [15].

## 2.3 Learning from Demonstration and Instructions

Historically, learning from demonstration [4] and learning from instructions [28, 36] developed separately, primarily as subfields of robotics and natural language processing, respectively. However, we group these two techniques together here because in most situations, interactive learning of complex procedural knowledge is best achieved using a combination of demonstration (actions) and instructions (utterances), and our unified theory supports this combination.

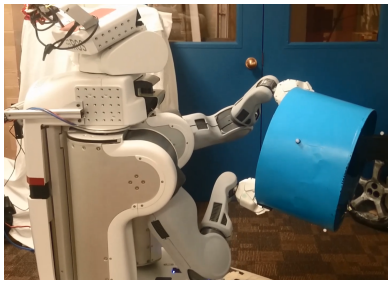

Figure 4: Tire rotation.

In a current research project [41], the investigators are developing algorithms and user interfaces for the PR2 robot in Figure 4 to learn the hierarchical structure of complex tasks, such as the tire rotation procedure in Figure 3, from a human teacher via a combination of demonstration and instructions. We are using Disco as the task representation in this current project. However, once the robot has learned a task, it is *not* able to teach it to another robot or human. Simply put, the goal of this new research is make it possible for the robot to teach what it has learned.

Most other work on learning from demonstration and/or instructions has not used hierarchical task representations. Two notable exceptions are [38], which is based on the Soar architecture, and [54], which combines demonstration and instructions. Neither of these systems both learns and teaches.

## 2.4 Other Learn-Do-Teach Research

Several other researchers have recently also come to the conclusion that learning, doing and teaching procedural knowledge need to be unified, but have applied it to simpler task representations.

Working within the reinforcement learning paradigm, Taylor [59, 62] uses action suggestion as a natural and theoretically unified method for humans or agents to teach agents and for agents to teach humans or agents. However, reinforcement is generally not an efficient approach for humans to learn or teach sequential procedures, because it uses a state-based action policy. This works well for a reactive application, such as playing Pac-Man (which Taylor uses in his experiments), but does not scale to complex hierarchical tasks.

Scorce et al. [56] motivate the need for a robot to first learn a task from one human and then teach it

to another human by the crew turnover problem on the International Space Station. However, their work is limited to simple linear plans. Furthermore, their work focuses on the human-robot interaction mechanisms, such as speech commands, rather than an underlying theoretical unification.

We also need to say something here about generic cognitive architectures, such as Soar [29], ACT-R [3] and ICARUS [30]. Such architectures seek a much more ambitious theoretical unification than we do, i.e., not just learning, doing and teaching, and not just hierarchical tasks, but all of cognition. Cognitive architectures therefore model perception, memory and other phenomena that are abstracted away at our level of theory. Practically speaking, although there have been many software agents built using these cognitive architectures, none to our knowledge can currently pass the Learn-Do-Teach Challenge described in Section 4.4; however, we hope the challenge will motivate the creation of such agents in the future.

### 2.5 Human Pedagogical Theory

Many psychologists and cognitive scientists have studied human teaching and learning as two sides of the same process, and have built empirically-based theoretical models at various levels of detail. For our proposed work, the most relevant branch of this research concerns identifying pedagogical dialogue moves [12, 20] and strategies [7]. Dialogue moves are the lowest level of interaction analysis and include, for example, student question categories, such as: enablement (what action allows another action to occur?), instrumental (what actions accomplish a goal?) and expectational (why did the expected action not occur?). Pedagogical strategies are recurring, larger phases of a teaching session, such as modeling (teacher does all/most of the task), scaffolding (teacher only intervenes when learner needs help) and fading (learner performs task independently).

See Section 4.3.4 for a discussion of how this human pedagogical theory will inform our research. To assist us, we have retained a prominent member of this research community, Dr. Natalie Person, as a consultant to this project (see budget justification).

## 3 Disco: HTNs and Collaborative Dialogue

We propose to use our existing open-source collaboration manager, Disco [49], as the computational platform for our research. This section introduces the related technical terminology and notational conventions necessary to understand the rest of this document.

### 3.1 Hierarchical Task Networks

*Hierarchical task networks* (HTNs) are a common and convenient representation for procedural knowledge about hierarchical tasks. Disco uses the ANSI/CEA-2018 standard [44] for HTNs, whose definition was led by PI Rich. The diagram on the right side of Figure 5 shows the simple example HTN that will be used throughout this proposal. To avoid distracting details, the tasks in this model have abstract names, such as A, B and C. Figure 3 is a more realistic HTN from the car maintenance domain.

An HTN is essentially a set of task decomposition rules, commonly called *recipes*, each of which (e.g., r1 in Figure 5) decomposes a *non-primitive* task (e.g., A) into a sequence of primitive and/or non-primitive tasks (e.g., B and C), which are called the *steps* of the recipe. In this proof of concept, all recipes are totally ordered, as indicated by the arrows between the steps. Additional features of HTNs in ANSI/CEA-2018 will be discussed in Section 4.3.1.

The complete set of recipes that recursively decompose a given non-primitive task, such as A, into *primitive* tasks, such as d, e, f, g, h and i, is conventionally drawn as a tree, such as in Figure 5, with A at the root and the primitives are at the fringe. The dotted lines in Figure 5 indicate that r2 and r3 are two alternative recipes for (ways to decompose) B. This structure is sometimes called an "and-or tree."

As a typographical convention in this document, the names of non-primitive tasks will always start with an uppercase letter, e.g., A or LearnStep, and the names of primitive tasks will start with a lowercase letter, e.g., d or addStep.

```
1  [A by r1]
2     do(human,A)                          "Let's do A."
3     [B by r2]
4        d(agent)
5        e(agent)
6        done(agent,B)                     "We're done doing B."
7     [C by r4]
8        h(human)
9        [i(agent)]
10          do(human,i(agent))             "Please do i."
11          i(agent)
12       done(human,C)                     "We're done doing C."
13    done(human,A)                        "We're done doing A."
```

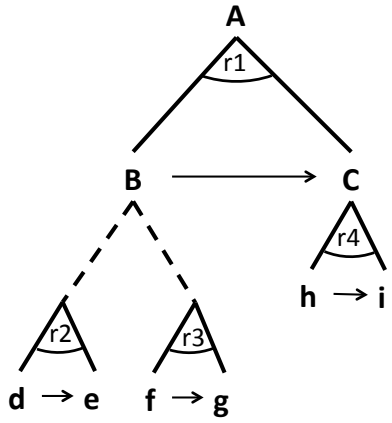


Figure 5: Segmented interaction history for one possible collaborative execution of example HTN.

### 3.2 Collaborative Dialogue

Since collaboration requires communication in some form (verbal and/or nonverbal), collaborative interactions are also dialogues. In this document we will therefore use the terms *dialogue* and *interaction* interchangeably to refer to interactions that, like Figure 5, include both actions and utterances.

The indented lines on the left of Figure 5 serve as an introduction to the notation, called a *segmented interaction history*, that Disco uses to record the structure of a collaborative dialogue. With a few small changes (such as bold fonts) to improve readability, all of the text shown in Figure 5 and similar figures in this proposal, including the utterance glosses in quotes, is automatically generated by Disco as the interaction proceeds.

In this interaction, all the recipes in the HTN in Figure 5 (r1, r2, r3 and r4) are assumed to already be known to both participants, i.e., the agent implemented in Disco and a human. This interaction thus illustrates the *do* component of learn-do-teach.

Notice first that a segmented interaction history is hierarchical (shown by indentation) and that each level of the hierarchy, called a *segment*, is introduced by a line in square brackets (e.g., lines 1, 3, 7, and 9) that indicates the *purpose* (goal) of the following segment. Thus the purpose of the toplevel segment is to achieve task A, with two subsegments to achieve B and C, and so on. In addition, for segments whose purpose is to achieve a non-primitive task, the recipe chosen is also identified (e.g., "by r1"). The fundamental insight of SharedPlans dialogue theory, which we see in this example, is that in task-oriented dialogues, the interaction structure reflects the underlying task structure.[1] Each line in the interaction history other than the bracketed purpose lines corresponds to an *event* in the interaction, which is the occurrence of either a primitive action from the fringe of the HTN, such as `d(agent)` on line 4, or an utterance, such as `do(human,A)` ("Let's do A") on line 2. Primitive actions and utterances are shown in the history as operators, where the first parameter indicates who performed the primitive (e.g., the agent or human) and additional parameters depending on the type of task. This simple operator notation could be further formalized in a logic of action [37], but that is not focus of this work.

The only two utterances used in this example, 'do' and 'done', are generic and thus do not appear in application-specific recipes, such as the HTN in Figure 5. As we will see in further examples, both the human and the agent can generate these utterances.

Furthermore, both of these utterances are *meta* (higher-order) operators, because they take an operator as a parameter. The utterance `do(human,A)`, said by the human on line 2 and glossed as "Let's do A," signals

[1]This does not mean that the interaction structure is always identical to the task structure. It is possible for the focus of attention to shift around in the task (see [32]).

the start of a new segment whose purpose is to achieve A. The utterance `done(human,A)` on line 13, which is glossed as "We're done doing A," signals the end of that segment. When 'do' is used with primitive tasks, the performer of the target task must also be specified,[2] as in `do(human,i(agent))` on line 10, where the human tells the agent to "Please do i," which the agent subsequently does on line 11. The 'done' utterance is optional and is often omitted for segments whose purpose is a primitive task.

The human's utterances and actions in this interaction were selected by the human from a menu. A slightly different collaboration would have resulted if the human made different choices. For example, in line 10 the human could have chosen to perform i herself rather than asking the agent to do it.

All of the agent's utterances and actions were automatically generated by the Disco agent, using its HTN planning and execution algorithm [45]. This algorithm keeps track of which step of the current HTN needs to be done next, when a recipe choice needs to be made, and so on. Disco also includes turn-taking rules and processes for deciding who should perform a particular primitive action.

# 4 Proposed Research: Supervised Interactive HTN Learning

To limit the size of this effort to the resources available, we have made a number of scoping decisions. First, since our theoretical goal fundamentally concerns teaching and learning by humans, we are focusing on *supervised* and *interactive* learning. In contrast with most current machine learning research, which focuses on exploiting large (or huge) amounts of data, we restrict ourselves to approaches that are realistic for human interaction, i.e., involving one or a few demonstrations of a single task and/or dozens of instructions, but not hundreds or thousands.

Second, we decided to start with a unified theory of learning, doing and teaching task *hierarchy* (using HTNs), rather than learning and teaching task primitives[3] or non-deterministic plans. HTNs have been and continue to be successfully applied in many practical applications, including with robots. Non-determinism is problematic for human interaction, because it tends to dramatically increase the amount of data required for teaching or learning. Furthermore, for complex tasks, the need for hierarchy does not go away when non-determinism is added. Section 4.3.2 discusses a limited form of uncertainty that we do allow in task pre- and postconditions and recipe choice conditions. Extending our work to fully non-deterministic plans is a suitable topic for future research.

Finally, since our theory is focused on the semantic level of interaction, we will use various practical approaches, such as menus and specialized languages, to avoid the difficulties of unrestricted natural language understanding and nonverbal communication.

## 4.1 Proof of Concept System

To make the idea of a unified theory more concrete and plausible, we have implemented a small proof of concept system in Disco, which passes the Learn-Do-Teach Challenge for the simple HTN shown in Figure 5. Following several example walkthroughs with this system below, we discuss our further research plans and the Learn-Do-Teach Challenge competition.

### 4.1.1 Pedagogical Strategies as Meta-Recipes

The Disco agent in Figure 5 is capable of collaboratively doing application tasks, such as C, but in order to also learn and teach such tasks, it needs to have *pedagogical* goals and strategies. An example of a pedagogical goal is teaching someone how to do C. Learning by demonstration and learning from instructions are examples of pedagogical strategies.

We introduce pedagogical goals and strategies into our framework using reflection and theory of mind. *Reflection* in computational systems in general is the process of reasoning about oneself, especially using

[2] In SharedPlans theory, all shared non-primitive goals are the joint responsibility of both collaborators, regardless of whether all of the primitives in a recipe happen to be performed by only one collaborator.

[3] We have recently worked in another project on combining hierarchy learning with primitive task learning [39, 31].

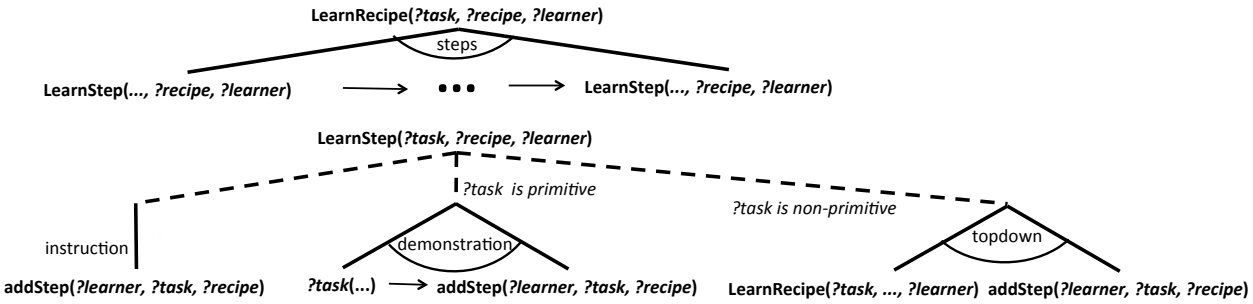


Figure 6: Pedagogical tasks and strategies.

the same representations and algorithms at both the object and meta (reflective) levels.

To start the reflection process, we first introduce a new generic non-primitive task, LearnRecipe, to express the goal of a pedagogical interaction. In general, the desired result of LearnRecipe is for the learner to have a mental representation at least sufficient to perform the recipe being taught. For example, if a human is teaching an agent how to do C by r4, the goal of the interaction is `LearnRecipe(C,r4,agent)`. LearnRecipe is a *meta-task*, because it takes a task as a parameter.

The postcondition (success condition) of this example of LearnRecipe is `knows(agent,C,r4)`, i.e., that the agent has a mental model equivalent to recipe r4 for C. The collection of such knowledge (knows) beliefs on the part of the teacher is a kind of theory of mind. *Theory of mind* in cognitive systems in general refers to the attribution of mental states to the participants in an interaction. The *student model* [21] in intelligent tutoring systems (see Section 2.2) is a special case of theory of mind. The student model in our proof of concept system is rudimentary—it only keeps track of which recipe steps the student knows (see Section 4.3.4).

We also need to introduce generic primitive tasks that modify mental states. In our proof of concept system, we have only one such primitive, namely `addStep(?learner,?task,?recipe)`, which adds the specified task as the next step in the specified recipe. The addStep meta-primitive is used both for updating the student model when an agent is the teacher and updating its *own* mental state when it is the learner.

Pedagogical strategies can now be expressed as recipes for decomposing pedagogical goals. In our proof of concept, there is only one toplevel pedagogical strategy, which is to teach the steps of a recipe in order. This strategy is expressed by the *steps* recipe shown at the top of Figure 6, which has a variable number of steps of type LearnStep, corresponding to the number of steps in the recipe being taught. We call steps a *meta-recipe*, because it decomposes a meta-task (LearnRecipe).[4]

Notice that there are three alternative meta-recipes for LearnStep: instruction, demonstration and top-down, which express the pedagogical strategies of learning from instructions, learning by demonstration, and topdown learning, respectively. Each of these recipes decomposes to instances of addStep. We will see examples of using each of these pedagogical strategies and explain them further in the walkthroughs below. Figure 6 also introduces two additional and important general features of HTNs: constraints and applicability conditions.

*Constraints* are equalities that an HTN enforces/requires between the parameters of constituent tasks. In the notation used here, these equalities are indicated by the appearance of the same variable, e.g., *?task*, in several places in the same recipe.

Optional boolean *applicability conditions* specify when alternative recipes may be used. For example, the instruction recipe has no applicability condition, which means it may always be used to decompose LearnStep. However, the demonstration recipe may only be used when the task parameter of LearnStep is primitive; the topdown recipe may only be used when the task parameter is non-primitive.

[4]This combination of reflection and theory of mind is closely related to meta-planning [60]; but meta-planning has not been applied before to building a learning and teaching agent.

```
[LearnRecipe(C,r4,agent) by steps]
   do(human,LearnRecipe(C,r4,agent))                "Let's learn how to do C."
   [LearnStep(h,r4,agent) by instruction]
      do(human,addStep(agent,h,r4))                 "First step is h."
      addStep(agent,h,r4)
   [LearnStep(i,r4,agent) by instruction]
      do(human,addStep(agent,i,r4))                 "Next step is i."
      addStep(agent,i,r4)
   done(human,LearnRecipe(C,r4,agent))              "We're done learning how to do C."
```

(a) Human teaching agent by instruction how to do C.

```
[C by r4]
   do(human,C)                                      "Let's do C."
   h(agent)
   i(agent)
   done(agent,C)                                    "We're done doing C."
```

(b) Agent doing C.

```
[LearnRecipe(C,r4,agent') by steps]
   do(agent,LearnRecipe(C,r4,agent'))               "Let's learn how to do C."
   [LearnStep(h,r4,agent') by demonstration]
      h(agent)
      addStep(agent',h,r4)
   [LearnStep(i,r4,agent') by demonstration]
      i(agent)
      addStep(agent',i,r4)
   done(agent,LearnRecipe(C,r4,agent'))             "We're done learning how to do C."
```

(c) Agent teaching another copy of itself by demonstration how to do C.

```
[LearnRecipe(C,r4,human') by steps]
   do(agent,LearnRecipe(C,r4,human'))               "Let's learn how to do C."
   [LearnStep(h,r4,human') by demonstration]
      h(agent)
      addStep(human',h,r4)
   [LearnStep(i,r4,human') by instruction]
      do(agent,addStep(human',i,r4))                "Next step is i."
      addStep(human',i,r4)
   done(agent,LearnRecipe(C,r4,human'))             "We're done learning how to do C."
```

(d) Agent teaching another human by mixture of demonstration and instructions how to do C.

Figure 7: Implemented proof of concept system illustrating Learn-Do-Teach Challenge and mixture of learning from demonstration and instructions.

More than one recipe for a task may be applicable in a given situation. For example, if the learned task is primitive, both the instruction and demonstration recipes are applicable. This is a choice that the teacher is free to make and may be informed by application-specific and/or generic heuristics. For example, a generic heuristic might be to always use the demonstration strategy the first time a given primitive is taught, and then to use the instruction strategy thereafter.

Specifically, in our system, because the pedagogical strategies are represented the same way as application-specific recipes, all of the computational mechanisms of collaboration are uniformly applied at both the object (application-specific) and meta (generic pedagogical) levels, making it possible to seamlessly shift between learning, doing and teaching. In Section 4.3.5, we discuss applying reflection one more time to *learn* pedagogical strategies within the same unified theory.

### 4.1.2 Learning from Demonstration and Instructions

Figure 7 shows four implemented walkthroughs with our proof of concept agent in which it performs, for task C, everything required by the Learn-Do-Teach Challenge. In order to focus on the learning semantics, we have somewhat simplified these interactions[5] by omitting grounding [13] behaviors (such as saying "Ok") and negotiation (such as accepting or rejecting a proposed goal), which are implemented in Disco and which we plan to use in our proposed research.

Figure 7(a) illustrates the use of the *instruction* strategy by the human to teach the agent how to do C. The instruction meta-recipe for LearnStep in Figure 6 expresses (in a simple way in this proof of concept) this pedagogical strategy of "telling" someone how to do something. The interaction begins on line 2, where the human introduces the LearnRecipe goal in exactly the same way that the A goal was introduced on line 2 in Figure 5. Notice on line 5 that the agent, following the instruction meta-recipe, performs an addStep, which appropriately updates its mental model of r4.

In Figure 7(b), after being asked by the human, the agent shows that it has learned how to do C by correctly performing all the steps in recipe r4.

In Figure 7(c) and (d), the agent is teaching C using the *same meta-recipes* it used to learn C. In Figure 7(c) it is teaching C to a fresh copy of itself (`agent'`), thus illustrating concretely that we are using the same theory for learning and teaching. In Figure 7(d) it is teaching C to a human. In both cases, the agent is making no assumptions about whether it is teaching an agent or a human.

Figure 7(c) illustrates the use of the *demonstration* strategy by the agent to teach C. The demonstration meta-recipe for LearnStep in Figure 6 expresses a simple, optimistic version of learning from demonstration, wherein after a primitive step is performed, the learner adds the step to its mental model of the current recipe, i.e., learning it the first time.

Figure 7(d) illustrates how the pedagogical meta-recipes allow the intermixing of demonstration and instruction strategies in teaching a single application-specific recipe. In this interaction, the original agent teaches *another human* (`human'`) how to do C, choosing demonstration for the first step and instruction for the second step of r4. Notice on lines 28 and 31 that the agent updates its student model by performing the appropriate addStep's with the bindings specified by the demonstration meta-recipe.

### 4.1.3 Topdown Teaching

Figure 8 shows how, after the agent already knows how to do C, the human teaches it how to do A. This interaction illustrates the *topdown* strategy. The topdown meta-recipe for LearnStep in Figure 6 expresses the idea of starting with the topmost task in a hierarchy and incrementally decomposing it—versus starting with primitives and combining them into bigger and bigger non-primitives, as was done in Figure 7(a). Notice that the topdown meta-recipe has a mutually recursive invocation of LearnRecipe, which allows the topdown strategy to be intermixed with other strategies at different levels in the recipe tree.

[5]The complete log files are available at http://github.com/charlesrich/Disco/tree/master/examples/learn-do-teach/test

```
1  [LearnRecipe(A,r1,agent) by steps]
2     do(human,LearnRecipe(A,r1,agent))                    "Let's learn how to do A."
3     [LearnStep(B,r1,agent) by topdown]
4        [LearnRecipe(B,r2,agent) by steps]
5           do(human,LearnRecipe(B,r2,agent))              "Let's learn how to do B."
6           [LearnStep(d,r2,agent) by demonstration]
7              d(human)
8              addStep(agent,d,r2)
9           [LearnStep(e,r2,agent) by demonstration]
10             e(human)
11             addStep(agent,e,r2)
12          done(human,LearnRecipe(B,r2,agent))  "We're done learning how to do B."
13       addStep(agent,B,r1)
14    [LearnStep(C,r1,agent) by instruction]
15       do(human,addStep(agent,C,r1))                          "Next step is C."
16       addStep(agent,C,r1)
17    done(human,LearnRecipe(A,r1,agent))        "We're done learning how to do A."
```

Figure 8: Human teaching agent topdown how to do A.

Due to space constraints, the last interaction required to finish learning the entire HTN in Figure 5 is omitted here. In that interaction, the human similarly teaches the second (alternative) recipe for B, namely recipe r3.

## 4.2 Architecture of the Unified Theory

We now step back and look at the conceptual architecture of our unified theory, which is comprised of the four layers shown in Figure 9. The bottom two layers are application-specific, while the top two layers are generic. Dotted arrows in the figure indicate the use of reflection. Each layer in the figure shows examples from our proof of concept system. Our further research described below will populate the top two generic layers with additional pedagogical strategies and associated meta-utterances.

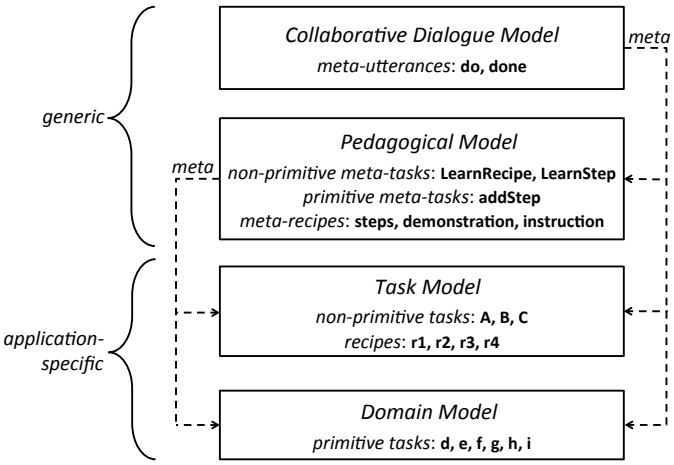


Figure 9: Layers of unified theory with examples from proof of concept system.

The bottommost layer in the architecture is the *domain model*, which specifies fundamental decisions about how to represent the possible states of the application world. We have adopted here a simple and commonly used approach, in which the domain model consists of a set of predicates that describe the state of the world and operators that change that state. In our proof of concept, the operators are the primitive tasks: d, e, f, g, h and i. We did not define any domain predicates in our proof of concept, but in general, domain predicates are needed to express preconditions and postconditions of tasks. It is also possible to have application-specific utterances. Other researchers have worked on learning domain models [19, 61, 63].

The second layer in the architecture, called the *task model*, contains the procedural knowledge that enables a human or agent to perform and communicate about complex application tasks. As discussed above, we have chosen HTNs as a common and convenient representation of this knowledge. Thus our task model contains the definitions of non-primitive application tasks and the recipes for decomposing them, as

for example shown in Figure 5. The procedural knowledge part of the agent's theory of mind (the student model for teaching) is also represented in terms of this task model.

The *pedagogical model* is the heart of our approach. The simple pedagogical model in our proof of concept is shown in Figure 6. Refining and expanding this pedagogical model is one of the major activities of our proposed research. As discussed above, we use reflection to express generic pedagogical strategies as meta-recipes in the same HTN representation as the application task model. Using the same representation and processes at both the object and meta levels is what makes it possible for our theory to unify learning, doing and teaching. The agent also uses the meta-recipe representation for the student model when teaching pedagogical strategies (Section 4.3.5).

Finally, the *collaborative dialogue model* contains generic meta-utterances, such as 'do' and 'done' in our proof of concept system, that are used mediate the collaborative learning, doing and teaching processes, as we saw in the preceding walkthroughs. Notice that these meta-utterances can be applied to either pedagogical or application-specific tasks, i.e., to tasks in the pedagogical model, task model or domain model in Figure 9. Disco contains a number of additional such meta-utterances, for example to repeat or stop working on a specified task [49], which we expect to take advantage of in our further research.

*Reflection and theory of mind have been used in cognitive architectures before–so what's novel and innovative here?* What's new and important here is a theory that unifies learning, doing and teaching. Reflection and theory of mind are powerful and well-known conceptual tools we are using to implement the theory in a computationally concise and effective form.

## 4.3 Extending the Proof of Concept

In the remainder of this section, we describe how we will build on our proof of concept system.

### 4.3.1 Learning and Teaching Other Core HTN Features

The only representational features of HTNs that our proof of concept agent can learn or teach are those shown in the simple task model of Figure 5: (total) ordering of recipe steps and alternative recipes. Representing real-world tasks requires many additional HTN features, including (see ANSI/CEA-2018 standard [44]): task inputs and outputs, task preconditions and postconditions, partially ordered recipe steps, recipe applicability conditions, and recipe constraints. Since our proof of concept agent is implemented in Disco, it is already capable of using all of these features in the context of collaboratively doing tasks. Below, we sketch preliminary ideas and directions for how to support learning and teaching each of these features within a unified theory.

In our proof of concept, the application-specific tasks (A, B, C, d, etc.) did not have parameters (other than to specify who performed primitive tasks), so there was no teaching or learning of task inputs or outputs. The essential challenge for teaching and learning the inputs and outputs of tasks is to generalize appropriately from the particular objects used in a concrete execution. This problem has been well-studied in the programming by example [34] field. We plan to develop pedagogical strategies based on this work, with the option of introducing new meta-utterances to help the teaching/learning process.

Task preconditions and postconditions are important for both execution and planning. For example, preconditions (when they are false) prevent you from trying to execute a task in a situation in which it will not work correctly. Postconditions (when they are false) tell you that a just-executed task has failed. We will build on the work of other researchers on learning and teaching preconditions and postconditions using natural language [10] and observation [2, 42].

The conventional machine learning approach to unordered or partially ordered recipe steps is for the teacher to demonstrate all (or most) possible orders, so that the learner can generalize appropriately. However, this approach often puts an unreasonable and unnatural burden on the teacher. For example, it doesn't matter in what order you unscrew the five lug nuts holding a car tire. To teach this, instead of demonstrating all 120 possible orders, a better collaborative pedagogical strategy might be to demonstrate one step,

i.e., unscrew one nut, and then to ask the learner to take off the other four nuts. This strategy implicitly suggests that the order doesn't matter. Furthermore, in the spirit of mixed-initiative (see Section 4.3.3) and active learning [8], a learning agent could ask the teacher a question to confirm that the steps are unordered. Also, see [40] for our related work on learning partial ordering based on a better understanding of frames of reference.

Recipe applicability conditions express how to choose between alternative recipes for the same task, such as r2 and r3 in Figure 5. We will build on other work on learning applicability conditions [63] and conditional plans [55] to developing pedagogical strategies for learning and teaching recipe applicability conditions.

The most common form of recipe constraint in HTNs is *data flow*, which is formally equality between the inputs and outputs of steps. For example, going back to our lug nuts, after each nut is unscrewed it is then put down. Formally, the input of the putdown action is constrained to be equal to the output of the unscrew action. Based on our recent learning-from-demonstration research, [40] we believe that once the inputs and outputs of tasks have been been correctly learned, the data flow follows quite easily. The main difficulty with learning such constraints is coincidences, i.e., when two objects are equal in a particular execution, but they don't need to be in general. One obvious pedagogical strategy to address this problem is to intentionally choose demonstration objects that avoid such coincidences.

This object-selection strategy does not seem to fit nicely into the meta-recipe formalism we introduced in our proof of concept system. In general, we expect to make extensions to our meta-level represention when we find that it does not have enough expressive power. These extensions will, however, maintain the consistency of representation and process between learning, doing and teaching.

### 4.3.2 Uncertainty in HTNs

A natural place to model uncertainty in HTNs is in the evaluation of task pre- and postconditions and recipe applicability conditions. Disco already implements a simple three-valued logic (true/false/unknown) for all HTN conditions, which avoids the explosion of data required for a full probabilistic representation. Disco also already has some simple strategies for executing HTNs with unknown conditions. We will explore strategies for teaching and learning HTNs with unknown conditions. Many of these strategies will involve new meta-utterances for telling and asking information.

For example, suppose a learner is unsure whether a task is executable in the current situation (unknown precondition). Two alternative strategies are to attempt the task anyways or to ask the teacher. Which strategy to try first depends on the cost of a failed attempt. Similarly, if a learner doesn't know whether a just-performed task has succeeded (unknown postcondition), the decision of whether to try executing the task again before asking the teacher depends on the cost (both in effort and effect) of duplicate executions.

From a teacher's point of view, suppose there is no good way to predict the best recipe for a particular non-primitive task (unknown applicability conditions). Instead of simply demonstrating a guessed recipe, a better strategy is to also tell the learner that there is no good rule for this decision.

We will also investigate approaches to adding more probabilistic information to HTNs without becoming impractical for human interaction.

### 4.3.3 Learner Initiative

Collaboration is a mixed-initiative interaction. For example, in Figure 5, the agent has taken the initiative (spoken or acted first) to achieve B in the first subsegment (lines 3–6), whereas the human has taken the initiative to achieve C in the second subsegment (lines 7–12). The discussion of pedagogical strategies above emphasizes the teacher's initiative. However, there are many reasons why a learner may want to temporarily take control of the interaction. For example, the learner may want to ask a specific clarifying question [9], report that she is confused on a particular point, etc. The teacher needs to have strategies (meta-recipes) for responding to such situations.

### 4.3.4 More General Pedagogical Strategies

The pedagogical strategies discussed thus far are closely tied to the structural features of HTNs. We also plan to explore the formalization of more general strategies (applied to HTNs), such as those used by human teachers (see Section 2.5). For example, with the addition of appropriate new pedagogical meta-tasks and meta-utterances, we should be able to express *scaffolding* (building up skills in a logically necessary order) and *fading* (incrementally withdrawing teacher assistance) as meta-recipes. The criteria for choosing a particular strategy in a particular situation are expressed in the applicability conditions of the corresponding meta-recipes. For example, the criteria for when to start fading (i.e., withdrawing teacher assistance) are expressed in the applicability condition of the fading recipe.

Other general pedagogical strategies center around learner errors and confusions. For example, a *validation* strategy would specify how much should be taught before the teacher asks the learner to perform the task in order to confirm correct learning (as Figure 7(a) and (b)). And if the validation fails, *correction* strategies would specify remedial teaching behaviors, with alternative recipes depending on the nature of the learner's error. For example, if the learner omitted a step in the validation, then that step needs to be retaught.

These more general pedagogical strategies will also require enriching the student model beyond what was used in the proof of concept. For example, the student model will need to keep track of the questions asked by the student, which tasks the student has tried on her own and whether she succeeded or failed at them. Formalizing these more general pedagogical strategies will also contribute to the broader impact of this research (see Section 6).

To support more general pedagogical strategies and other useful functions, such explanation and traceability, we also plan to add generic meta-knowledge to our theoretical framework. For example, it would be useful if an agent could answer questions regarding from whom it learned a particular piece of knowledge, to whom it taught it, which recipes and meta-recipes were used, etc. Practically speaking, this information is readily available in Disco.

### 4.3.5 Learning and Teaching Pedagogical Strategies

In our proof of concept system, all of the pedagogical strategies were manually coded as meta-recipes. In the final year of our project, we plan to explore the feasibility of applying reflection one more time and using our unified theoretical framework to learn and teach the meta-recipes—in other words, "teaching the teacher."

One way of approaching this problem would be to stay within the two-participant collaboration setting and suppose that a student agent learns the teacher's pedagogical meta-recipes at the same time as it learns the application-specific task model being taught. However, this seems too difficult and does not align with how human teachers are taught, i.e., teaching skills usually come *after* mastery of the knowledge being taught. Therefore, an alternative approach we will investigate is a three-party collaboration with a teacher, a student, and an observer, where the observer is trying to learn the pedagogical strategies demonstrated by the teacher. Although Disco has been used almost exclusively for two-party collaborations, we have built prototype three-party collaboration systems [26] in Disco.

## 4.4 The Learn-Do-Teach Challenge

Figure 1 is a pictorial representation of the Learn-Do-Teach Challenge. The basic idea is that an agent must (a) learn a new task from a human, (b) do the task, and then teach it to (c) another copy of itself and (d) another human. (These letters correspond with the examples in Figure 7.) Steps (e) and (f) are related to scoring the competition, as discussed below. This challenge will serve two purposes: It will provide a rigorous evaluation of our own agent and it is a key component of our broader impact.

We start by focusing on how to run the challenge as a competition for other researchers. Since the ultimate goal is to motivate other agent researchers, such as those mentioned in Section 2, to unify learning,

doing and teaching in their architectures, the competition must be fair, objective and hopefully engaging. We also need to worry about accidental or intentional cheating. For practical reasons, the competition will be limited to autonomous software agents and be automated as much as possible.

We will develop a set of target HTNs formalized in ANSI/CEA-2018 [44]. One likely domain is household tasks in a simulated world. Some of these HTNs will be provided to competition participants in advance for practice; others will be reserved for the competition rounds. For each agent in the competition, the goal of a round is to successfully learn, do and teach the given task model. As the competition rounds progress, either at one meeting or over successive meetings, the target task models will become increasingly complex.

We will provide a simple teaching language to be used by all agents on arrows (a), (c) and (d) in Figure 1. In our proof of concept, this language consisted of the utterances 'do' and 'done' and the primitive domain tasks. We expect to add additional utterance types based on our own research and in negotiation with potential competition participants. Our goal will be to make it easy for participants to translate their own teaching language into this shared language.

Each researcher (playing the role of the human teacher in Figure 1) is then free to compose a sequence of inputs (a) to teach the target task to their agent. After processing these inputs, the agent must then (b) perform the target task in a simulated world. We will provide an automatic program that scores the correctness of this and the other task performances on arrows (e) and (f). Next, the agent must generate a sequence of outputs (in the shared teaching language), which are given to both (c) another—fresh—copy of the same agent and (d) an unbiased human volunteer.

Finally, both the (e) agent copy and the (f) human volunteer will perform the target task in a simulated world with their performance scored by the same automatic program used in (b) above. We will combine the scores from (b), (e) and (f) into a final score to judge the winner of the competition. Of course, there will undoubtedly many other competition details to be debugged once we start working on it.

Notice that a competitor desiring to cheat could simply pass teaching inputs (a) without change to (c) and (d), which would likely result in a good score on (f) and avoid needing to implement teaching strategies at all. To prevent this cheating, we will require that output (c)-(d) is sufficiently different from input (a), as tested by an automatic comparison program. We will evaluate possible comparison algorithms with pilot experiments.

We plan to submit an application to the 2020 AAAI Spring Symposium (usually in Stanford, CA) for a workshop at which to host the first Learn-Do-Teach Challenge competition, accompanied by technical presentations and discussions by the participants. Such a workshop could also fit with the AAAI, IJCAI, Intelligent Virtual Agents (IVA), Human-Robot Interaction (HRI) or Autonomous Agents and Multi-Agent Systems (AAMAS) conferences.

To use the Learn-Do-Teach Challenge for evaluating our own agent, the process is the same as described above, except that instead of being a competition, we will run it as a controlled user study. Study participants will fill all the human roles in Figure 1 and we will run and score different versions of our algorithms to gain insights into the underlying theory.

# 5 Research Plan and Timeline

Figure 10 summarizes our proposed three-year timeline involving per year one and a third full-time graduate students (with undergraduate assistance), one faculty month of each PI, and three days of consultation by Dr. Person (see Section 2.5).

The focus of the first two years is to build upon our proof of concept system by expanding both the types of procedural knowledge and the pedagogical strategies supported. The focus of our third year will be on the development and hosting of the Learn-Do-Teach Challenge, as described in Section 4.4, which will serve both as an evaluation of our technical work and as a vehicle for broader impact on the scientific subfields we seek to influence. We will also explore learning and teaching pedagogical strategies, as discussed in Section 4.3.5.

To make our work more concrete and pave the path toward future practical applications, we have tentatively chosen three application areas in which we will pursue this research. The first two applications leverage prior domain modeling efforts: car maintenance (see Section 4.3.5) and the operational procedures for a gas turbine engine (see Section 2.2). The third application area, household tasks such as cooking and cleaning, will likely serve best for the Learn-Do-Teach Challenge, because of its familiarity. All of these tasks will be in simulation with a software agent.

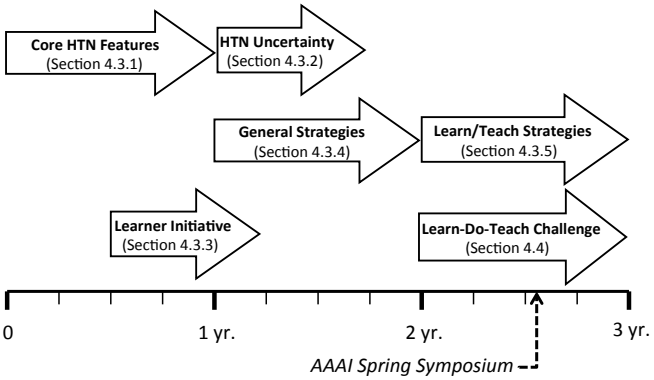

Figure 10: Research timeline.

# 6 Broader Impacts of the Proposed Work

Autonomous agents and robots that interact with humans in diverse environments are the next step in the evolution of automation technology. Although such agents and robots will have some basic capabilities builtin, their task repertoire will also need to be extended "in the field" via collaboration with humans or other agents. Basic research to make this possible will give the U.S. a competitive advantage in the new industries arising from this technology.

By developing computational models of human pedagogical theories, this project can also contribute to improving human pedagogy. For example, if you reverse the first human and agent in Figure 1 you obtain Figure 11, in which a human student learns from a computer agent, and then teaches what she has learned to another computer agent or another student. It is well known [6] that having students teach other students enhances learning. Our research will allow experiments to determine the potential benefit of having students teach computer agents. More generally, our software agent implementation will make it possible to explore the performance of different pedagogical strategies in different situations much more exhaustively than purely human studies, resulting in knowledge that can be applied by human teachers with human students.

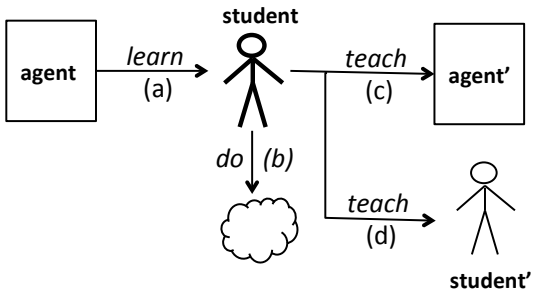

Figure 11: Human-centered learn-do-teach.

As a faculty member associated with the Learning Sciences and Technology Program at WPI, co-PI Rich will be able to transfer this knowledge into WPI's Assistments project,[6] which currently serves over 600 teachers and their students from 43 states, including many students in underrepresented minorities.

A substantial part of the research described here will directly contribute to the training of graduate and undergraduate engineering students, including women. Of the three graduate students most recently supervised by the investigators, two are women. We will continue to actively recruit women and underrepresented minorities for this project.

Finally, in addition to disseminating the results of this project through scientific publication and open-source software, we will raise the visibility and interest within the broader research community in a unified theory of hierarchical tasks via a novel and engaging research competition, called the Learn-Do-Teach Challenge.

[6] http://assistments.org

## References


[1] J. Allen, N. Chambers, G. Ferguson, L. Galescu, H. Jung, M. Swift, and W. Taysom. Plow: A collaborative task learning agent. In *Proceedings of the National Conference on Artificial Intelligence*, page 1514, 2007.

[2] E. Amir and A. Chang. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research*, pages 349–402, 2008.

[3] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111(4):1036, 2004.

[4] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 2009.

[5] D. Bohus and A. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech and Language*, 23(3):332–361, 2009.

[6] D. Boud, R. Cohen, and J. Sampson. *Peer learning in higher education: Learning from and with each other*. Routledge, 2014.

[7] W. Cade, J. Copeland, N. Person, and S. D'Mello. Dialogue modes in expert tutoring. In *Intelligent Tutoring Systems*, pages 470–479. Springer, 2008.

[8] M. Cakmak, C. Chao, and A. Thomaz. Designing interactions for robot active learners. *Autonomous Mental Development, IEEE Transactions on*, 2(2):108–118, 2010.

[9] M. Cakmak and A. Thomaz. Designing robot learners that ask good questions. In *ACM/IEEE Int. Conf. on Human-Robot Interaction*, pages 17–24, Boston, MA, March 2012.

[10] R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz. Tell me when and why to do it! Run-time planner model updates via natural language instruction. In *Proc. ACM/IEEE Conf. on Human-Robot Interaction*, pages 471–478, Boston, MA, 2012.

[11] B. Cheikes and A. Gertner. Teaching to plan and planning to teach in an embedded training system. In *Proc. 10th Int. Conf. on Artificial Intelligence in Education*, pages 398–409, San Antonio, TX, May 2001.

[12] M. Chi, S. Siler, H. Jeong, T. Yamauchi, and R. Hausmann. Learning from human tutoring. *Cognitive Science*, 25(4):471–533, 2001.

[13] H. H. Clark and S. E. Brennan. Grounding in communication. *Perspectives on socially shared cognition*, 13(1991):127–149, 1991.

[14] J. Davies, N. Lesh, C. Rich, C. Sidner, A. Gertner, and J. Rickel. Incorporating tutorial strategies into an intelligent assistant. In *Proc. ACM Int. Conf. on Intelligent User Interfaces*, pages 53–56, Santa Fe, NM, Jan. 2001.

[15] A. De Antonio, J. Ramírez, R. Imbert, and G. Méndez. Intelligent virtual environments for training: An agent-based approach. In *Multi-Agent Systems and Applications IV*, pages 82–91. Springer, 2005.

[16] E. DeKoven, D. Keyson, and A. Freudenthal. Designing collaboration in consumer products. In *Proc. ACM Conf. on Computer Human Interaction, Extended Abstracts*, pages 195–196, Seattle, WA, Mar. 2001.

[17] G. Ferguson and J. F. Allen. A cognitive model for collaborative agents. In *AAAI Fall Symposium: Advances in Cognitive Systems*, 2011.

[18] G. Ferguson, J. F. Allen, et al. Trips: An integrated intelligent problem-solving assistant. In *National Conference on Artificial Intelligence*, pages 567–572, 1998.

[19] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, 1999.

[20] A. Graesser, N. Person, and J. Magliano. Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology*, 9(6):495–522, 1995.

[21] J. Greer and G. McCalla. *Student modelling: The key to individualized knowledge-based instruction*. Springer Science & Business Media, 2013.

[22] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, Oct. 1996.

[23] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[24] B. J. Grosz and C. L. Sidner. Plans for discourse. In P. R. Cohen, J. L. Morgan, and M. E. Pollack, editors, *Intentions and Communication*, pages 417–444. MIT Press, Cambridge, MA, 1990.

[25] D. Gruen, C. Sidner, C. Boettner, and C. Rich. A collaborative assistant for email. In *Proc. ACM Conf. on Computer Human Interaction, Extended Abstracts*, pages 196–197, Pittsburgh, PA, May 1999.

[26] P. Hanson and C. Rich. A non-modal approach to integrating dialogue and action. In *Proc. 6th AAAI Artificial Intelligence and Interactive Digital Entertainment Conf.*, Palo Alto, CA, Oct. 2010.

[27] G. Hoffman and C. Breazeal. Collaboration in human-robot teams. In *Proc. of the AIAA 1st Intelligent Systems Technical Conference*, Chicago, IL, USA, 2004.

[28] S. B. Huffman and J. E. Laird. Flexibly instructable agents. *Journal of Artificial Intelligence Research*, 1995.

[29] J. Laird, A. Newell, and P. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[30] P. Langley and D. Choi. A unified cognitive architecture for physical agents. In *Proceedings of the National Conference on Artificial Intelligence*, page 1469, 2006.

[31] C. Lee and D. Berenson. Learning guiding constraints for narrow passages from one demonstration. In *Int. Symp. on Experimental Robotics*, Tokyo, Japan, October 2016.

[32] N. Lesh, C. Rich, and C. Sidner. Collaborating with focused and unfocused users under imperfect communication. In *Proc. 9th Int. Conf. on User Modelling*, pages 64–73, Sonthofen, Germany, July 2001. Outstanding Paper Award.

[33] J. C. Lester, L. S. Zettlemoyer, J. P. Grégoire, and W. H. Bares. Explanatory lifelike avatars: Performing user-centered tasks in 3D learning environments. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 24–31. ACM, 1999.

[34] H. Lieberman. *Your wish is my command: Programming by example*. Morgan Kaufmann, 2001.

[35] K. E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572, Dec. 1998.

[36] Ç. Meriçli, S. D. Klee, J. Paparian, and M. Veloso. An interactive approach for situated task specification through verbal instructions. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1069–1076, 2014.

[37] J.-J. Meyer. Dynamic logic for reasoning about actions and agents. In J. Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer, 2000.

[38] S. Mohan and J. Laird. Learning goal-oriented hierarchical tasks from situated interactive instruction. In *Proceedings of the Twenty Eighth AAAI Conference on Artificial Intelligence*, 2014.

[39] A. Mohseni-Kabir, C. Li, S. Chernova, D. Berenson, C. Rich, and C. Sidner. Simultaneous learning of hierarchy and primitives for complex robot tasks. 2017. In preparation.

[40] A. Mohseni-Kabir, C. Rich, and S. Chernova. Learning partial ordering constraints from a single demonstration. In *Proc. ACM/IEEE Conf. on Human-Robot Interaction*, Bielefeld, Germany, 2014.

[41] A. Mohseni-Kabir, C. Rich, S. Chernova, C. Sidner, and D. Miller. Interactive hierarchical task learning from a single demonstration. In *Proc. ACM/IEEE Int. Conf. on Human-Robot Interaction*, Portland, OR, 2015.

[42] M. N. Nicolescu and M. J. Matarić. Experience-based representation construction: Learning from human and robot teachers. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 740–745, 2001.

[43] K. O'Brien, J. Sutherland, C. Rich, and C. Sidner. Collaboration with an autonomous humanoid robot: A little gesture goes a long way. In *Proc. ACM/IEEE Conf. on Human-Robot Interaction*, Lausanne, Switzerland, 2011.

[44] C. Rich. Building task-based user interfaces with ANSI/CEA-2018. *IEEE Computer*, 42(8):20–27, Aug. 2009.

[45] C. Rich, N. Lesh, J. Rickel, and A. Garland. A plug-in architecture for generating collaborative agent responses. In *Proc. 1st Int. J. Conf. on Autonomous Agents and Multiagent Systems*, Bologna, Italy, July 2002.

[46] C. Rich and C. Sidner. Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3/4):315–350, 1998. Reprinted in S. Haller, S. McRoy and A. Kobsa, editors, *Computational Models of Mixed-Initiative Interaction*, Kluwer Academic, Norwell, MA, 1999, pp. 149–184.

[47] C. Rich, C. Sidner, and N. Lesh. Collagen: Applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4):15–25, 2001. Special Issue on Intelligent User Interfaces.

[48] C. Rich and C. L. Sidner. Collaborative discourse, engagement and always-on relational agents. In D. Bohus et al., editors, *Dialog with Robots, Papers from the 2010 Fall Symposium*. AAAI Press, Menlo Park, CA, Nov. 2010.

[49] C. Rich and C. L. Sidner. Using collaborative discourse theory to partially automate dialogue tree authoring. In *Proc. Int. Conf. on Intelligent Virtual Agents*, pages 327–340, Santa Cruz, CA, Sept. 2012.

[50] J. Rickel and W. L. Johnson. Task-oriented collaboration with embodied agents in virtual worlds. In J. Cassell, J. Sullivan, and S. Prevost, editors, *Embodied Conversational Agents*, pages 95–122. MIT Press, Cambridge, MA, 2000.

[51] J. Rickel, N. Lesh, C. Rich, C. Sidner, and A. Gertner. Building a bridge between intelligent tutoring and collaborative dialogue systems. In *Proc. 10th Int. Conf. on Artificial Intelligence in Education*, pages 592–594, San Antonio, TX, May 2001.

[52] J. Rickel, N. Lesh, C. Rich, C. Sidner, and A. Gertner. Using a model of collaborative dialogue to teach procedural tasks. In *Working Notes of AI-ED Workshop on Tutorial Dialogue Systems*, pages 1–12, San Antonio, TX, May 2001.

[53] J. Rickel, N. Lesh, C. Rich, C. Sidner, and A. Gertner. Collaborative discourse theory as a foundation for tutorial dialogue. In *6th Int. Conf. on Intelligent Tutoring Systems*, pages 542–551, Biarritz, France, June 2002.

[54] P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso. Interactive robot task training through dialog and demonstration. In *2nd ACM/IEEE International Conference on Human-Robot Interaction*, pages 49–56, 2007.

[55] M. D. Schmill, T. Oates, and P. R. Cohen. Learning planning operators in real-world, partially observable environments. In *Artificial Intelligence Planning Systems*, pages 246–253, 2000.

[56] M. Scorce, G. Pointeau, M. Petit, A.-L. Mealier, G. Gibert, and P. Dominey. Proof of concept for a user-centered system for sharing cooperative plan knowledge over extended periods and crew changes in space-flight operations. In *Proc. 24th Int. Symposium on Robot and Human Interactive Communication*, Kobe, Japan, 2015.

[57] C. L. Sidner and C. Forlines. Subset languages for conversing with collaborative interface agents. In *Int. Conf. on Spoken Language Processing*, Sept. 2002.

[58] C. L. Sidner, C. Lee, C. Kidd, N. Lesh, and C. Rich. Explorations in engagement for humans and robots. *Artificial Intelligence*, 166(1-2):104–164, 2005.

[59] L. Torrey and M. Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1053–1060, 2013.

[60] R. Wilensky. Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive science*, 5(3):197–233, 1981.

[61] Q. Yang, K. Wu, and Y. Jiang. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, 171(2):107–143, 2007.

[62] Y. Zhan, A. Fachantidis, I. Vlahavas, and M. E. Taylor. Agents teaching humans in reinforcement learning tasks. In *Proceedings of the Adaptive and Learning Agents Workshop (AAMAS)*, 2014.

[63] H. H. Zhuo, H. Munoz-Avila, and Q. Yang. Learning hierarchical task network domains from partially observed plan traces. *Artificial Intelligence*, 212:134–157, 2014.