A Script for Learning Scripts: Automatic Acquisition of Procedural Knowledge by Ontologically Grounded Agents

Christian Arndt (arndtc@rpi.edu), Sergei Nirenburg (nirens@rpi.edu) &
Sanjay Oruganti (orugas2@rpi.edu)
Department of Cognitive Science, Rensselaer Polytechnic Institute

Abstract

Scripts describe events with component subevents. Ontologically grounded AI systems use scripts to represent the procedural knowledge necessary to execute complex tasks. In this paper, we present recent advances in dialogue-based script learning by agents configured in an ontologically grounded cognitive architecture. Specifically, we describe the procedure that details the sequence of events agents undertake when learning scripts in some subject domain or application. We demonstrate this script-learning script in a simulation system by tracing how an agent learns to replace a fuse through dialog with a human partner.

Keywords: Script learning; Knowledge representation; Interactive learning; Cognitive architectures; Symbolic AI

Introduction

The "knowledge bottleneck"—that is, the cost of acquiring high-quality, interpreted knowledge—has been a longstanding challenge for symbolic AI. A promising way of overcoming it is enabling agents to learn while they perform their tasks. We present recent improvements to the learning procedures used by ontologically grounded agents called OntoAgents, which are the subject of current research in the LEIA lab at RPI. By learning, we mean automatically acquiring lexicon entries and ontological concepts. This is distinct from the definition of learning used in data-driven AI research, where it refers to honing probability distributions based on large data sets or reward functions.

OntoAgents depend on a variety of types of knowledge, including ontological scripts, which are complex events consisting of one or more subevents along with their participants and props. The notion of a script was introduced by Schank and Abelson (1977) and further developed for ontologically grounded agents (Nirenburg & Raskin, 2004). Scripts allow agents to reason about and execute event sequences.

The OntoAgent ontology records physical, verbal, and mental events. The latter include the agent's knowledge about internal processing algorithms for things like perception interpretation, goal and plan selection, attention management, and—most important for this paper—learning. That is, agents know how to learn because they have a script that tells them what to do.

Scripts that operate over other scripts are called metascripts. This paper describes the OntoAgent metascript for learning domain scripts (such as how to build a chair or fix an engine) through dialog with an instructor. This work advances previous work on script learning by OntoAgents (McShane et al., 2024b).

Scripts contain not only a list of subevents but various kinds of information about how to execute them. These involve time management, checking and updating property values, instantiating effects of given events, and so on (McShane & Nirenburg, 2021b). Langley et al. (2020) refer to this kind of knowledge as a form of expertise fundamental to complex task execution.

Developing this kind of learning has been the subject of prior research. Kiddon et al. (2015) use machine learning techniques to extract scripts in the form of "action graphs" from a corpus of written recipes. Mohan and Laird (2014) describe a formalized way to represent tasks, alongside algorithms which can extract those representations from texts in limited domains. Kirk and Laird (2019) present a method for transferring knowledge about one task to another in the domain of games and puzzles involving physical objects. Scheutz et al. (2024) introduce TRACS, a cognitive architecture designed to support teaching and updating tasks with a limited lexicon for use in manufacturing. The use of LLMs to assist in task learning has also been explored (Kirk et al., 2024).

In this line of research, textual pattern matching or LLMs are used to process text. These approaches to learning cannot make use of ontologically grounded meaning representations and ontological knowledge about the context as the source of decision-making heuristics for, and expectations about, the material to be learned. Also, most of the above systems operate over very limited vocabularies and very small world models, which limits their applicability to small domains. Our work seeks to address this gap: the learning metascript described in this paper is designed for applications in any domain and is built alongside the OntoAgent semantic analyzer, a robust language understanding system operating over a lexicon of about 30,000 English word senses and an underlying ontology of about 9,000 concepts, with an average of sixteen properties defining each of them (McShane & Nirenburg, 2021c).

Another meaningful contribution of our work is the holistic conceptualization of learning procedures alongside processing services to support other cognitive functions such as language understanding and generation, agenda management, and memory. Our past research has shown that these cognitive functions, while often treated separately in research, are

In G. Boleda, I. Dautriche, M. Maldonado, C. Saldana and T. Torrent (Eds.), *Proceedings of the 48th Annual Conference of the Cognitive Science Society*

interdependent in complicated ways (McShane & Nirenburg, 2021b). The learning metascript presented here reflects this understanding: learning depends on many kinds of processing, most centrally involving language understanding, and it cannot be isolated as a single, separate task. Similarly, other cognitive functions like action selection and memory are intertwined with the methods OntoAgents use to learn and represent complex events.

One might ask how a metascript is different, formally and functionally, from a domain script. Formally, it is no different: it is a complex event recorded in the ontology using the same expressive means as any other script. Functionally, however, it ranks among a handful of cognitively-oriented scripts that function as daemons, meaning that they are always available on an agent's agenda. Among other things, this means that OntoAgents can learn alongside other activities, in addition to learning as a primary goal. The metascript for learning tells agents how to extract information from an input, when to begin and end learning, and what to do with newly learned knowledge.

Over the years, research into OntoAgents has experimented with a variety of kinds of and approaches to learning: opportunistic learning of lexical and ontological knowledge from inputs with unknown words (McShane & Nirenburg, 2021a) and (McShane & Nirenburg, 2021d); deliberate learning by scraping the web to expand lexicon coverage (English, 2010) and (Nirenburg et al., 2007); experiential learning, when an agent must remember something that has happened in its world (McShane et al., 2009); and learning scripts in physical and simulated environments (McShane et al., 2024b).

The work reported here is different in two ways. First, it adds new functionalities to the learning process and goes beyond focusing, as did our earlier work, on individual examples. Second, by introducing the concept of metascripts, it illustrates the general trend in OntoAgent research to move from procedural implementations to inspectable, declarative knowledge-oriented implementations of the agent's reasoning apparatus.

The Metascript in Action

The learning metascript has been implemented in a demonstration system available here (RPI LEIA Lab, 2026). In this demo, an instructor teaches an agent a script for replacing a fuse. The agent is then told to replace the blown fuse in an electrical panel inside a simulated ship engine room (Figure 1). The agent executes the relevant event sequence, which demonstrates that it can both learn and execute complex tasks.

Learning Scripts through Instruction

A high-level overview of the script-learning metascript, along with the related attention service subroutines, is displayed in Figure 3.

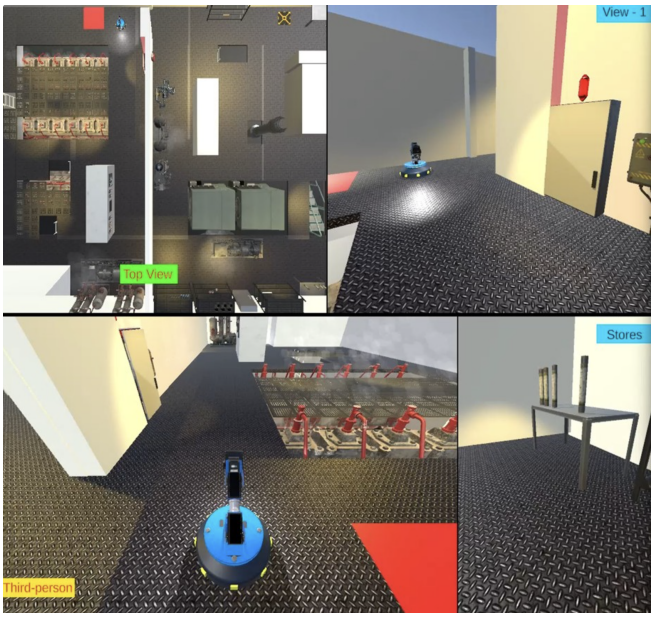

Figure 1: The simulated ship engine room used in the fuse-replacement demo.

Triggering Learning

OntoAgents use a semantic analyzer to translate natural language inputs into ontologically grounded meaning representations (MRs) (McShane & Nirenburg, 2021c). MRs are built from instances of ontological concepts and their properties. They serve as inputs to all internal decision-making and execution, including the execution of the learning metascript. A detailed description of MRs is outside the scope of this paper.

We will use a simplified version of the fuse-replacement script learned in the demo as a running example to demonstrate all the different steps of the metascript. Specifically, we will walk through the steps an agent would use to learn from the input text: "Let me teach you how to replace a fuse. First, make sure the power is off. Remove the old fuse. Insert the new fuse, and you're done." First, the agent will generate meaning representations for each of the sentences in the input. Then it will recognize that an instance of script learning must be carried out, and start an instance of the metascript. By the end of the metascript's operation, the agent will add a script named REPLACE-FUSE to its ontological hierarchy. This new script will be a child of REPLACE-PART and contain as subevents the events described in the input. Figure 2 illustrates the meaning representation of the first sentence in the input text.

The MR shown in Figure 2 includes instances of three ontological events: REQUEST-ACTION-1, SCRIPT-LEARNING-EVENT-1, and REPLACE-PART-1. It also contains instances of three ontological objects—an instance of FUSE, and the instances representing the participants in the learning session: the instructor (an instance of HUMAN) and the particular ONTOAGENT. FUSE-1 contains a link to the ontological concept it instantiates (through INSTANCE-OF) and the event in which

REQUEST-ACTION-1	
AGENT:	HUMAN-1
BENEFICIARY:	ONTOAGENT-1
THEME:	SCRIPT-LEARNING-EVENT-1
SCRIPT-LEARNING-EVENT-1	
THEME:	REPLACE-PART-1
REPLACE-PART-1	
THEME:	FUSE-1
FUSE-1	
INSTANCE-OF:	FUSE
THEME-OF:	REPLACE-PART-1
HUMAN-1	
ONTOAGENT-1	

Figure 2: The MR generated from the input text "Let me teach you how to replace a fuse" (simplified for readability).

it participates (through THEME-OF). REQUEST-ACTION-1 is an instance of the REQUEST-ACTION concept. It represents the meaning of the input sentence: HUMAN-1 (its AGENT) is asking ONTOAGENT-1 (its BENEFICIARY) to complete SCRIPT-LEARNING-EVENT-1 (its THEME). The MR for the entire text is a set of similar structures, with all mentions of particular actions and objects cross-referenced.

The metascript is triggered when the agent's attention module determines two things: the input MR contains a learning trigger, and the information being learned is a script (a complex event). These decisions are illustrated in cells 1 and 2 of Figure 3. Learning triggers are clues that an agent should learn from an input. The MR in Figure 2 contains this trigger—the agent is being told to learn a script—so the agent instantiates an instance of the script-learning metascript. The metascript has three subtasks: 1) choosing the learning target, 2) learning from the input, and 3) wrapping up the learning session. The subsections below describe each of these in turn.

Choosing a Learning Target

The agent must first decide whether it will a) start learning a new script, b) continue to learn a script it has already started learning, or c) modify an existing script stored in its ontology. In other words, the agent must choose where to store the information it is about to learn. This is called the **learning target**, and its choice is shown in cells 4-8 of Figure 3. The ability to identify learning targets and associate new inputs with existing learning targets is central to building agents that can manage complex learning sessions over multiple dialogue turns.

When the agent is deciding whether to associate a new input with an existing learning target it asks itself two things: 1) Is the speaker talking about the same thing as before? This is adjudged based on the cumulative distance between concepts in the new MR and concepts in the MRs from previous dialog turns. 2) Does the rhetorical structure of the new dialog turn suggest that it continues something already started? Phrases like "Next" and "The second thing you need to do" give rise to MR interpretations that serve as such clues.

Learning from the Input

Once the agent has established the learning target, it must select and sequence information from the MR that will be included in the script being learned. This process is called **event sequence normalization** (McShane et al., 2024b), and it is shown in cells 9-11 of Figure 3. The agent must then merge the newly normalized event sequence into the learning target, as shown in cell 12. These steps are described below.

Identify core semantic content in the script. The first step of this normalization is to determine what parts of the input meaning representation form the script's core semantic content, shown in cell 9 of Figure 3. This means the agent must filter out irrelevant information and identify all events that are important to a script's execution. So far we have addressed the following eventualities:

1. Information in the MR that is about the learning process itself is filtered out. In our example, the meaning of the phrase "Let me teach you..." is not a part of the script to replace a fuse.
2. The different ways that script events can be expressed in language are neutralized. For example, "Open the electrical panel", "You need to open the electrical panel", and "It is necessary for the electrical panel to be open" will result in slightly different MRs, but they all mean that OPEN (THEME: ELECTRIC-PANEL) is the subevent in question.
3. Important subevents that were not mentioned in the dialog are recovered via inferencing. Consider the text: "Remove the bolt with a wrench. Then take off the cover." This explanation is fine if intended for humans. But the agent must understand—and include in the script being learned—that before taking off the cover, it must let go of the wrench. This missing event can be identified using the following heuristic: if the MR for a subevent requires a particular filler for its INSTRUMENT case role, but the next subevent requires a different filler or no filler at all, then the metascript inserts an instance of the PUT-DOWN event between the two, with its THEME case role filled by previously used tool.
4. Tools that were not mentioned in the input are recovered using inferencing. Consider a text like: "Remove the bolt." Here, the agent must recognize that the removal of a bolt requires a wrench. The need for inferencing like this is not specific to learning, but it is required since the agent needs to be able to immediately carry out the processes it is learning. Prior work has addressed this capacity (McShane & Nirenburg, 2021d) and (McShane et al., 2024a).
5. MR segments that are not related to script execution are not learned as part of the script, though they will be recorded as general information in the ontology, if applicable. Consider a dialog turn such as: "Find a new fuse with the same amperage rating as the old one. If the amperage rating is too low, the fuse might blow." The second sentence contains useful information about why amperage rating matters, but none of the events it describes are executable script subevents.

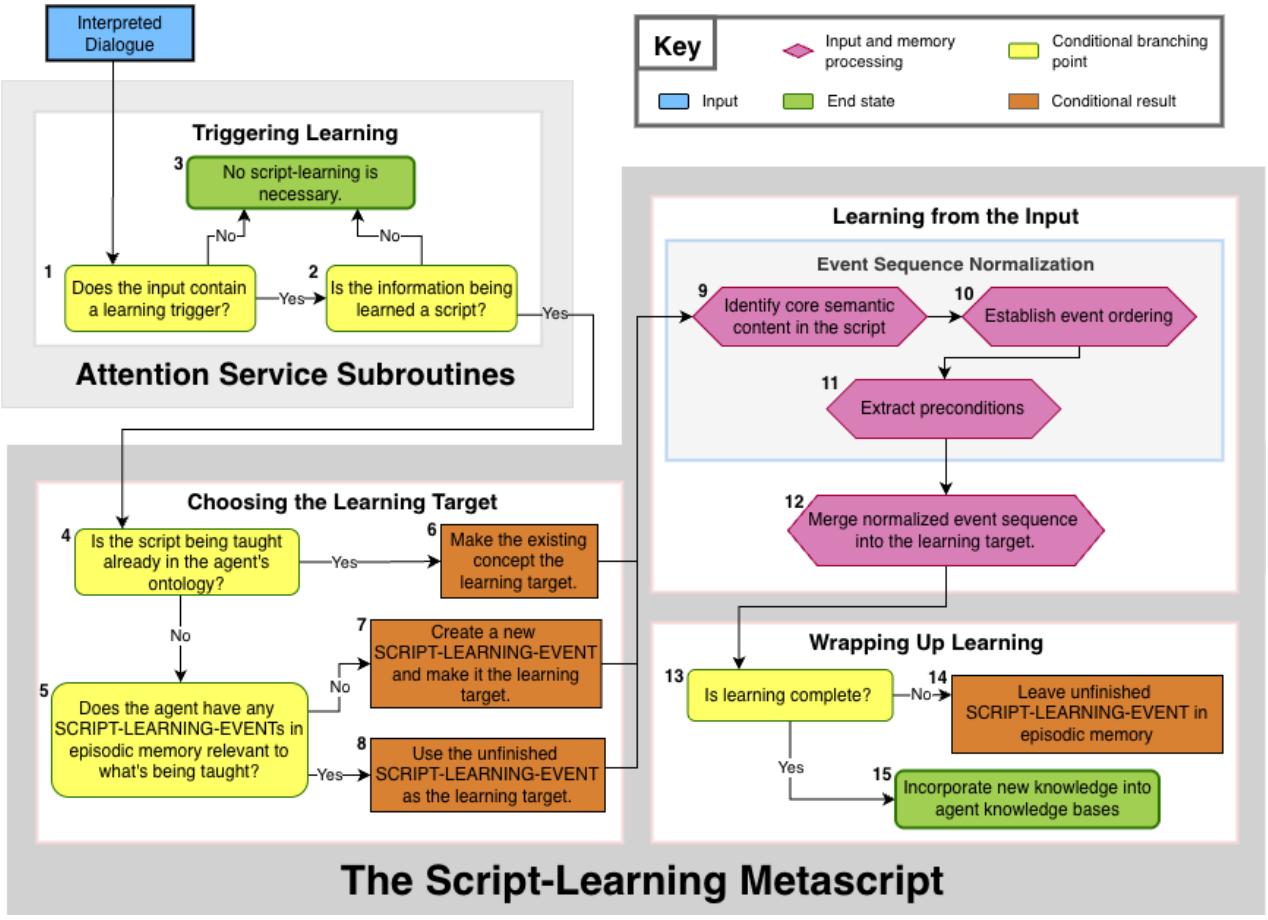

Figure 3: Overview of the script-learning metascript.

After the agent has identified core semantic information in the MR, the MR from our running example has been reduced to the set of frames displayed in Figure 4.

VERIFY-1	
AGENT:	ONTOAGENT-1
THEME:	ELECTRICAL-PANEL-1.POWER = OFF
PRECEDES:	REMOVE-1
REMOVE-1	
AGENT: ONTOAGENT-1	
THEME:	FUSE-1
INSERT-1	
AGENT:	ONTOAGENT-1
THEME:	FUSE-2

Figure 4: The events determined to be relevant to the script being learned.

Establish Event Ordering. Once the agent has a script's core subevents, it needs to establish their ordering (McShane et al., 2024b) (cell 10 of Figure 3). This is because events are not always presented in their correct temporal sequence: for example, "Remove the fuse. You need to open the door first, though."

Event ordering is established on the basis of Allen's set of temporal relations (Allen, 1983) along with non-temporal relations like conditionals. Consider two subevents in a script for replacing the fuse in an electric panel: 1) opening the panel's door, and 2) removing the blown fuse. The relevant relation is PRECEDES: the OPEN event PRECEDES the REMOVE event.

Agents assume that, unless explicitly stated otherwise, events in an input are listed in chronological order: "Open the door. Remove the old fuse." If they don't follow this default, people use sequencing cues to make the relations explicit: "Remove the fuse. You need to open the door **first**, though." Interpretation of these sequencing cues is carried out by the semantic analyzer and recorded in the MR. The metascript, therefore, only needs to ensure that the implicit sequential relations are made overt. In practice, the agent: 1) identifies all events that are not a part of temporal relations made explicit by the instructor, and 2) establishes PRECEDES relations between them in the order in which they occurred in the input.

Of course, there are more complicated sequencing relations as well, such as temporal overlaps and subsumptions, conditionals, and loops. Although the OntoAgent ontology can readily represent such relations, dynamically interpreting and

learning them is challenging. Given our focus on preparing agents to carry out useful learning in near-term cognitive robotic applications, such relations are not an immediate priority.

The MR from our running example is shown in Figure 5. REMOVE-1 now has a PRECEDES relation whose filler is INSERT-1.

VERIFY-1	
AGENT:	ONTOAGENT-1
THEME:	ELECTRICAL-PANEL-1.POWER = OFF
PRECEDES:	REMOVE-1
REMOVE-1	
AGENT:	ONTOAGENT-1
THEME:	FUSE-1
PRECEDES:	INSERT-1
INSERT-1	
AGENT:	ONTOAGENT-1
THEME:	FUSE-2

Figure 5: The ordered sequence of script subevents.

Precondition Extraction. Next, an agent must check to see if any of the newly ordered subevents should be represented as preconditions of other events. Formally, preconditions for an event are states of the world (i.e. property values of objects or events) that must be true before the event can be executed. For example, to clean an oil spill, one must first have a mop or rag: possessing that tool is a precondition to cleaning. Similarly, effects are states that represent the result of the script's completion: the effect of cleaning an oil spill is that the floor is clean. The script-learning metascript can choose to store events in the input MR as preconditions for the chronologically following events. This process is called **precondition extraction**, shown in cell 11 of Figure 3. Precondition extraction does not change the semantic content of a script—it organizes that semantic content based on a set of best practices. Currently, precondition extraction takes place under two conditions:

1. The effect of one subevent is a precondition for another subevent. Consider: "Pick up a wrench and remove the bolts." Possessing a wrench is a precondition for removing the bolts, so an agent learning these events as part of a script will demote it from the top-level sequence of events in that script.
2. One of the overtly stated subevents is the verification of a state. Consider a part of the text from our running example: "First, make sure the power is off. Remove the old fuse." In the MR, this as represented as a VERIFY event preceding a REMOVE event. According to our principles of ontological organization for scripts, a better interpretation is that the power being off is a precondition to the REMOVE event. The metascript updates the event sequence to reflect this interpretation.

Our example event sequence is shown in Figure 6 after precondition extraction.

REMOVE-1	
AGENT:	ONTOAGENT-1
THEME:	FUSE-1
PRECONDITION:	ELECTRICAL-PANEL-1.POWER = OFF
INSERT-1	
AGENT:	ONTOAGENT-1
THEME:	FUSE-2
PRECEDES:	REMOVE-1

Figure 6: The event sequence after precondition extraction is complete.

Merge Normalized Event Sequence into Learning Target. At this point, the agent's earlier assessment of the learning target comes back into play. If the learning target is a newly created script-learning event, the normalized event sequence represents all the agent knows about the script being learned. If the learning session enhances an existing script or continues an ongoing learning event, the agent must identify the relationship between the newly normalized event sequence and the event sequence already present in the learning target. In the current implementation, the metascript attaches newly learned event sequences to the end of existing event sequences.

Learning Completion

Once the agent has finished learning from a particular dialog turn, it must decide whether learning is complete (cell 14 of Figure 3). The agent interprets phrases like "...and you're done" as instances of the END-LEARNING-EVENT concept, as shown in Figure 7:

END-LEARNING-EVENT-1	
AGENT:	ONTOAGENT-1

Figure 7: The MR generated from the input "and you're done."

If an END-LEARNING-EVENT instance is present in the MR, the metascript considers the script-learning event complete. Otherwise, the potentially unfinished script-learning event remains in the agent's episodic memory (cell 15 of Figure 3).

When the agent determines that a script is complete, the metascript's remaining tasks are to record the new script into the agent's knowledge resources (cell 16 of Figure 3). This involves two subtasks: 1) anchoring the new script in the ontology, and 2) creating a new lexical sense to refer to the event captured by the script (McShane et al., 2024b).

The initial learning trigger provides one heuristic for deciding on the new script's placement in the ontology as well as its name. Here is how it works out in our running example.

The agent knows (see Figure 2) that the concept for the newly learned script is a kind of REPLACE-PART event whose theme is constrained to FUSE. Therefore, the new concept is added to the ontology as a child of REPLACE-PART. In cases where this method for finding the most appropriate parent for a concept fails, we use the an algorithm previously developed by our research group (Nirenburg et al., 2018) as a fallback.

The metascript also includes naming rules for new concepts.

In this case, the new concept has a more narrowly specified constraint on its THEME than its parent concept: REPLACE-PART expects its THEME to be a generic part, whereas the new concept expects it to be a fuse. The naming process is as follows: 1) Remove the parent's THEME from it's name if present (turning REPLACE-PART into REPLACE-), and 2) append the new concept's narrow THEME constraint to it (turning REPLACE- into REPLACE-FUSE). Similar rules exist for other eventualities. For example, if a new script differs from its parent by having a more specific constraint in its INSTRUMENT property, then the agent concatenates the concept names with WITH in-between: thus, a child of REMOVE that involves a WRENCH will be named REMOVE-WITH-WRENCH.

Next, the agent needs to build a lexicon entry to refer to the new concept. The agent can reverse engineer a lexicon entry using the input text and MR that initially triggered the learning event. In our example, the agent traces REPLACE-PART-1 (shown in Figure 2) back to the construction "replace a fuse" in the input text. The agent recognizes "replace" is a transitive verb whose direct object is "a fuse". This information forms the content of the syntactic zone of the lexicon entry. The meaning of this phrase is represented by the newly learned concept REPLACE-FUSE, which becomes the content of the semantic zone of the entry. This is added to the lexicon as another sense of the verb "replace".

Conclusions

The implemented script-learning metascript described above is sufficient to prepare cognitive robots to learn simple tasks right away, using a strategy that makes the learned knowledge fully inspectable and enhanceable over time. The script learning process described by the metascript is generic across domains and applications, meaning that OntoAgents can be ported to any physical robots or disembodied agents using the OntoAgent cognitive architecture (Oruganti et al., 2025). Future work involves not only script learning itself (i.e., enhancing the robustness of all of the algorithms presented here by covering additional eventualities in the learning process), but also improving the prerequisites for script learning (e.g., better semantic analysis and grounding) and, in case of applications featuring embodied agents, enabling the robots to physically carry out learned scripts.

As regards script learning per se, we have two foci of near-term enhancement. The first involves dialog management. In the system we demonstrated, the complete script is learned from a single dialogue turn. While the script-learning metascript already addresses the eventuality of multi-turn dialogs, heuristics need to be developed to support associated reasoning. For example, the agent needs more sophisticated reasoning about how to merge new information into an existing event sequence, rather than defaulting to the assumption that new events follow in sequence from events already present. The capability of deciding whether and when to initiate a clarification dialog with the human teammate was investigated in our team's work in the past (Nirenburg et al., 2018) and must be incorporated into the script learning presented in this paper.

The second near-term enhancement involves using LLMs and the web as teachers. Whereas the metascript presented here focuses on dialog-based learning with human teachers, LLMs and the web can also serve as sources of information about complex events. An LLM can play the role of an instructor teaching the OntoAgent operational procedures. The OntoAgent can ask the LLM for a description of a complex event, semantically analyze the LLM's output, and extract salient learning material to be learned, as described above. Finally, the agent can generate a natural-language summary of the script it has learned and ask the LLM to doublecheck its correctness before committing the new knowledge to memory (Oruganti et al., 2024). Even though the content generated by LLMs is not reliable in the general case, LLMs seem to be a reasonable stand-in for a human teacher in script learning. Indeed, our initial experimentation suggests that procedural knowledge encoded in scripts is not particularly subject to hallucinations. We are hopeful that knowledge acquisition of this type may allow OntoAgents to quickly expand their ontological library of scripts at low cost.

References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, *26*(11), 832–843. https://doi.org/10.1145/182.358434

English, J. (2010). *Learning by reading: Automatic knowledge extraction through semantic analysis* [Doctoral dissertation, University of Maryland, Baltimore County]. https://jesseenglish.com/downloads/dissertation.pdf

Kiddon, C., Ponnuraj, G. T., Zettlemoyer, L., & Choi, Y. (2015, September). Mise en place: Unsupervised interpretation of instructional recipes. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 982–992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1114

Kirk, J. R., & Laird, J. E. (2019). Learning hierarchical symbolic representations to support interactive task learning and knowledge transfer. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 6095–6102. https://doi.org/10.24963/ijcai.2019/844

Kirk, J. R., Wray, R. E., Lindes, P., & Laird, J. E. (2024). Improving knowledge extraction from llms for task learning through agent analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(16), 18390–18398. https://doi.org/10.1609/aaai.v38i16.29799

Langley, P., Shrobe, H., & Katz, B. (2020). A cognitive task analysis of rapid procedure acquisition from written instructions. *Proceedings of the Seventh Annual Conference on Advances in Cognitive Systems*. https://advancesincognitivesystems.github.io/acs/data/ACS2020_paper_42.pdf

McShane, M., & Nirenburg, S. (2021a). Basic semantic analysis. In *Linguistics for the age of ai* (pp. 141–200). The MIT Press. https://doi.org/10.7551/mitpress/13618.001.0001

McShane, M., & Nirenburg, S. (2021b). A brief overview of natural language understanding by leias. In *Linguistics for the age of ai* (pp. 59–115). The MIT Press. https://doi.org/10.7551/mitpress/13618.001.0001

McShane, M., & Nirenburg, S. (2021c). *Linguistics for the age of ai*. The MIT Press. https://doi.org/10.7551/mitpress/13618.001.0001

McShane, M., & Nirenburg, S. (2021d). Situational reasoning. In *Linguistics for the age of ai* (pp. 285–300). The MIT Press. https://doi.org/10.7551/mitpress/13618.001.0001

McShane, M., Nirenburg, S., & English, J. (2024a). Language understanding and generation. In *Agents in the long game of ai: Computational cognitive modeling for trustworthy, hybrid ai* (pp. 83–116). The MIT Press. https://doi.org/10.7551/mitpress/14940.001.0001

McShane, M., Nirenburg, S., & English, J. (2024b). Learning. In *Agents in the long game of ai: Computational cognitive modeling for trustworthy, hybrid ai* (pp. 173–226). The MIT Press. https://doi.org/10.7551/mitpress/14940.001.0001

McShane, M., Nirenburg, S., Jarrell, B., Beale, S., & Fantry, G. (2009). Maryland virtual patient: A knowledge-based, language-enabled simulation and training system. *Bio-Algorithms and Med Systems*, *5*(9), 57–63. https://doi.org/TODO

Mohan, S., & Laird, J. (2014). Learning goal-oriented hierarchical tasks from situated interactive instruction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *28*(1). https://ojs.aaai.org/index.php/AAAI/article/view/8756

Nirenburg, S., McShane, M., Beale, S., Wood, P., Scassellati, B., Magnin, O., & Roncone, A. (2018). Toward human-like robot learning. In *International conference on applications of natural language to information systems* (pp. 73–82). Springer International Publishing.

Nirenburg, S., Oates, T., & English, J. (2007). Learning by reading by learning to read. *International Conference on Semantic Computing (ICSC 2007)*, 694–701.

Nirenburg, S., & Raskin, V. (2004). The static knowledge resources: Ontology, fact database and lexicons. In *Ontological semantics*. The MIT Press.

Oruganti, S., Nirenburg, S., English, J., & McShane, M. (2024). Automating knowledge acquisition for content-centric cognitive agents using llms. *Proceedings of the 2023 Fall AAAI Symposia*, *2*(1). https://ojs.aaai.org/index.php/AAAI-SS/article/view/27703

Oruganti, S., Nirenburg, S., McShane, M., English, J., Roberts, M. K., Arndt, C., Gonzalez, C., Seo, M., & Sentis, L. (2025). Harmonic: A content-centric cognitive robotic architecture. https://arxiv.org/abs/2509.13279

RPI LEIA Lab. (2026). *Interactive demo displaying an x-agent being taught via instruction.* [Accessed: 2026-01-17]. https://rpi-leia.github.io/harmonic/demo%E2%80%91teaching.html

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum Associates.

Scheutz, M., Oosterveld, B., Peterson, J., & Wyss, E. (2024). Dialogue-based task instructions and modifications for industrial robots. *Proceedings of ARCI'24*. https://hrilab.tufts.edu/publications/scheutzetal24arci.pdf