In Proceedings of the AAAI 2007 Spring Symposium on Interaction Challenges for Intelligent Assistants, Stanford, CA March, 2007.

Explaining Task Processing in Cognitive Assistants That Learn

Deborah L. McGuinness[1], Alyssa Glass[1,2], Michael Wolverton[2], and Paulo Pinheiro da Silva[1,3]

Stanford University[1] Stanford, CA {dlm | glass}@ksl.stanford.edu
SRI International[2] Menlo Park, CA mjw@ai.sri.com
University of Texas, El Paso[3]. El Paso, TX paulo@utep.edu

Abstract

As personal assistant software matures and assumes more autonomous control of its users' activities, it becomes more critical that this software can explain its task processing. It must be able to tell the user why it is doing what it is doing, and instill trust in the user that its task knowledge reflects standard practice and is being appropriately applied. We will describe the ICEE (Integrated Cognitive Explanation Environment) explanation system and its approach to explaining task reasoning. Key features include (1) an architecture designed for re-use among many different task execution systems; (2) a set of *introspective predicates* and a software wrapper that extract explanation-relevant information from a task execution system; (3) a version of the Inference Web explainer for generating formal justifications of task processing and converting them to user-friendly explanations; and (4) a unified framework for explanation in which the task explanation system is integrated with previous work on explaining deductive reasoning. Our work is focused on explaining belief-desire-intention (BDI) agent execution frameworks with the ability to learn. We demonstrate ICEE's application within CALO, a state-of-the-art personal software assistant, to explain the task reasoning of one such execution system.

Introduction

Personalized software assistants have the potential to support humans in everyday tasks by providing assistance in cognitive processing. If these agents are expected to achieve their potential and perform activities in service of humans (and possibly other agents) then these agents need to be fully accountable. Before their users can be expected to rely on cognitive agents, the agents need to provide the users with justifications for their decisions, including that those decisions are based on appropriate processes that are correct and on information that is accurate and current. The agents need to be able to use these justifications to derive explanations describing how they arrived at a particular recommendation, including the ability to abstract away detail that may be irrelevant to the user's understanding and trust evaluation process. Further, if the agents are to be used to perform tasks, they need to explain how and under what conditions they will execute a task. Additionally, if their information and procedure base is being updated, potentially by learning algorithms, they may need to explain what has changed and why it has been changed.

One significant challenge to explaining cognitive assistants is that these assistants, by necessity, include task processing components that evaluate and execute tasks, as well as reasoning components that take input and determine conclusions. Thus, a cognitive assistant explainer will need to explain task processing responses as well as results of more traditional reasoning systems. An explanation solution thus must be able to encode, analyze, and summarize justifications of task execution along with other forms of reasoning, providing access to both inference and provenance information, which we refer to as knowledge provenance (Pinheiro da Silva, McGuinness, McCool, 2004).

Work has been done in the theorem proving community as well as in many specialized reasoning communities including description logics, expert systems, and FOL systems, to explain deductions. A limited amount of work has also been done in the task execution community on explaining task processing. What has not been done is work explaining task execution in a way that is also appropriate for explaining deductive reasoning and provenance. Our work provides a uniform approach to representing and explaining results (including provenance) from both communities, with the additional emphasis on providing explanations of learned information.

Our design and implementation work is in the setting of the DARPA Personalized Assistant that Learns (PAL) program, as part of the Cognitive Assistant that Learns and Organizes (CALO) project. In the CALO system, which includes work from 22 different organizations, the heart of the cognitive assistant is a belief-desire-intention (BDI) architecture (Rao & Georgeff, 1995). This presents a complex explanation challenge where CALO must explain conclusions from multiple knowledge sources, both hand built and automatically generated, with multiple reasoning techniques including task processing, deduction, and learning. In this paper, we will present our representation, infrastructure, and solution architecture for explaining BDI-based task processing; describe how it has been implemented in our new Integrated Cognitive Explanation Environment (ICEE); and show how it has been used to explain cognitive assistants. We also discuss preliminary results from a study of CALO users that show how explanation can be a key component in building user trust in cognitive assistants, particularly when those assistants are being updated by learning components.

Motivating Scenario

As a motivating scenario, we provide an example from the CALO office domain. In this example, the cognitive agent is working in the role of an office assistant, and has been tasked with purchasing a laptop for its user. In order to accomplish the high-level goal of buying a laptop, the cognitive agent uses a simple three step process with a laptop specification as input.

The first subtask, *GetQuotes*, requires the agent to obtain three quotes from three different sources. The second subtask, *GetApproval*, requires a particular form to be signed by an approval organization representative. The final subtask, *SendOrderToPurchasing*, requires a requisition form to be sent to purchasing. Note that in sequential tasks such as these, the termination condition of a previous subtask is typically a pre-condition to the next subtask.

The user may ask for an explanation of the agent's behavior at any time. ICEE provides a display of the current tasks/subtasks and a dialogue interface to explaining the agent's actions. The user can request explanations by starting a dialogue with any of several supported *explanation request* types. For example:

"Why are you doing <subtask>?"

This question is an example of an explanation request concerning the motivation for a task. Other task explanation request types include questions about status, execution history, forward-looking execution plans, task ordering, or explicit questions about time (for example, starting/completion times and execution durations). ICEE also handles extensive questions about task provenance, including explanations about the requestor of a task and the source of the represented procedure which is being executed by the system. These questions have been guided by an initial study focused on how to build user trust in cognitive agents.

ICEE contains one or more *explanation strategies* for each explanation request type. Based on context such as system status and past interactions, in addition to a model of the user, ICEE chooses one strategy. We have documented a set of question classes for cognitive assistants and designed explanation strategies for explanation request types (McGuinness *et al.* 2005). For example, in response to the above question, ICEE may choose the simple strategy of revealing the task hierarchy:

"I am trying to do <high-level-task>, and <subtask> is one subgoal in the process."

For each explanation request, ICEE can either reuse an existing task justification, which includes a task execution trace, or build and parse a new justification. Then ICEE presents as explanations the portions of the justification that are relevant to the query and explanation strategy. Additionally, ICEE suggests follow-up explanation requests for the user, enabling mixed initiative dialogue between the agent and the user. For example, follow-up questions in the above scenario might include:

"Why are you doing <high-level-task>?"
"Why haven't you completed <subtask> yet?"
"Why is <subtask> a subgoal of <high-level-task>?"
"When will you finish <subtask>?"
"What sources did you use to do <subtask>?"

This last example follow-up question in particular makes use of task provenance information, necessitating that the system keep track of sources used for both knowledge and task information.

The ICEE System

Architecture Overview

The architecture of ICEE (shown in Figure 1) is designed to be flexible and allow explanations to be derived from justifications gathered seamlessly from a variety of task processing and knowledge systems. An *explanation dispatcher* gathers structured explanation requests from the user through a collaboration agent or user interface. The assistant's user interface provides specialized mechanisms for users to request explanations.

Based on the type of the explanation request, the explanation dispatcher determines which explainer will handle the request, and forwards it to the proper explainer component, to identify relevant strategies and gather necessary information from the associated external component. For questions related to task processing, the *Task Manager (TM) explainer* handles the request. The TM explainer instructs the TM *wrapper* to gather task processing information about the requested tasks. The TM wrapper is closely coupled with a BDI execution system, or task manager. We have provided a TM wrapper for the task execution system used in CALO, which is based on the SRI Procedural Agent Realization Kit (SPARK); however, any similar task execution system could be similarly enhanced with explanation capabilities.

The TM wrapper stores the gathered task processing information in the *task state database*. This database is then used by the *justification generator* to create a justification for the tasks currently under execution, including any additional processing that is related to the current tasks. The generated justification can then be used by the TM strategies to create alternative explanations and

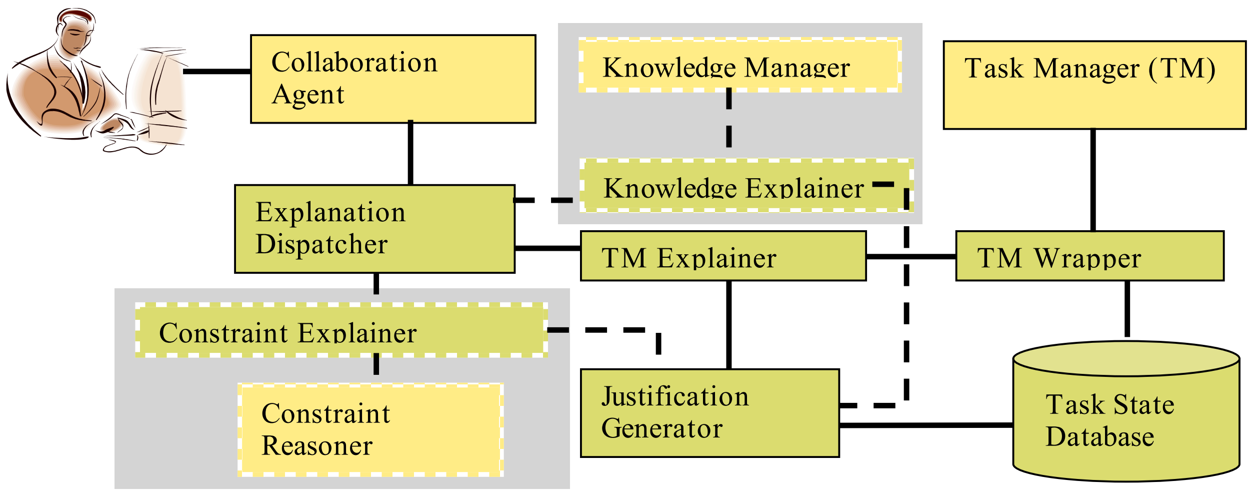


Figure 1: ICEE architecture. Shaded boxes show how additional explanation capabilities (beyond explaining task reasoning) are integrated into the overall framework; these additional components are not discussed in this paper.

select the one most salient to the user's questions. The explanation dispatcher returns the selected explanation to the collaboration agent for appropriate display to the user. Each of these architectural components is discussed below.

Task Oriented Processing

Any complex cognitive agent must have a mechanism for representing and executing tasks. A belief-desire-intention (BDI) model is a common framework for task reasoning components. BDI systems cover a range of execution capabilities, including hierarchical task encoding, control of procedural agent behavior, sequential and/or parallel execution of sub-procedures, conditional execution and branching, flexible procedure preconditions, and meta-level reasoning to determine which applicable procedure to pursue.

Task management in CALO is provided by SPARK (Morley & Myers 2004), one such BDI agent framework, which maintains sets of procedures that define agent actions. The CALO Task Manager's knowledge base of procedures includes human-authored procedures along with procedures that were partially or completely learned based on evolving information. ICEE gathers information on both static aspects of procedures within SPARK as well as dynamic information about its past and current execution state.

Introspective Predicates

ICEE is designed to provide the users of cognitive agents with the ability to ask detailed questions about task execution and to engage in a mixed initiative dialogue about its past, current, and future task execution. To provide detailed explanations of the behavior of the task processing system, justifications must be annotated with enough meta-data to support a wide range of behaviors. Particularly when answering complex questions, an explanation system must have access to information about many aspects of execution and planning.

To allow usable, detailed explanations to be generated, a task execution system must expose this meta-information. One of our contributions is a specification of a set of *introspective predicates* that were designed to provide access to meta-information required for explainable task processors.

These introspective predicates fall into three categories:

1. *Basic Procedure Information*: relatively stable, static information that is not dependant on when or how a task is executed. Provenance information about how task definitions have been created or learned is a key aspect of these introspective predicates.
2. *Execution Information*: dynamic information that is generated as a task begins being executed, and remains valid in some form throughout the execution of that task. This information also includes history related to completed tasks.
3. *Projection Information*: information about future execution, as well as alternatives for decision points that have already passed.

A task execution system, such as SPARK, that provides access to this set of introspective predicates can be linked to ICEE and allow it to fully explain all the question types and strategies described above. Details on the introspective predicates can be found in (Glass & McGuinness 2006).

Wrapper, Action Schema and Action Database

In order to collect explanation-relevant information from the task execution agent and store it in a format understandable by the explainer, we designed and built a *wrapper* for SPARK and an intermediate *action schema* in which to record task execution information. These elements were designed to achieve three criteria:

- *Salience*. The wrapper should obtain information about an agent's processing that is likely to address some possible user information needs.
- *Reusability*. The wrapper should obtain information that is also useful in other cognitive agent activities that require reasoning about action—for example, state estimation and procedure learning.
- *Generality*. The schema should represent action information in as general a way as possible, covering the action reasoning of blackboard systems, production systems, and other agent architectures.

The wrapper collects a snapshot of SPARK's current processing state as well as the previous decisions that led to that state. It uses SPARK's expanded introspective predicates to extract the portions of its underlying intention structure relevant to its current intentions, building this structure by recursively querying for the supporting elements of intentions and procedures. Example queries include: What are the agent's current intentions? What is the procedure instance that led to intention *X*? What are the preconditions that were met before procedure *P* could be executed?

After collecting the snapshot, the wrapper stores it in a SPARK-independent task execution action database. A portion of the schema of the action database is shown in Figure 2. This schema reflects that most task execution systems share the same common structure. While the current terminology in our schema is consistent with SPARK's, the concepts are general and consistent with other cognitive architectures. For example, "procedures" in our schema are equivalent to "knowledge sources" in BB* and other blackboard architectures, "procedure instances" are equivalent to "Knowledge Source Activation Records (KSARs)", etc. (Hayes-Roth 1985). The database records the relationships between key entities relevant to the agent's current state, for example, which intentions were established by which procedure instances, which procedure a given procedure instance instantiates, which variables were bound (and to what value) within a given procedure instance, and which of a procedure's termination conditions were satisfied and which were not.

Our approach of creating a system-specific wrapper and a generic action schema achieves multiple design goals. Because the action schema is generic across multiple task execution systems, only a new wrapper is needed in order to explain a cognitive agent using a task management system other than SPARK; no change to the justification representation discussed below is needed, and the same explanation strategies can be used. Additionally, the action schema can be reused by other components in the overall cognitive agent architecture for purposes beyond explanation, such as state capture, time-line archiving, or snapshotting.

Generating Formal Justifications

A cognitive agent's actions should be supported by *justifications* that are used to derive and present understandable explanations to end-users. These justifications need to reflect both how the actions support various user goals, and how the particular actions chosen by the agent were guided by the state of the world. More specifically, our approach to task justification breaks down the justification of a question about a particular task *T* into three complementary strategies, described here using terminology from SPARK:

- *Relevance*: Demonstrate that fulfilling *T* will further one of the agent's high-level goals, which the user already knows about and accepts
- *Applicability*: Demonstrate that the conditions

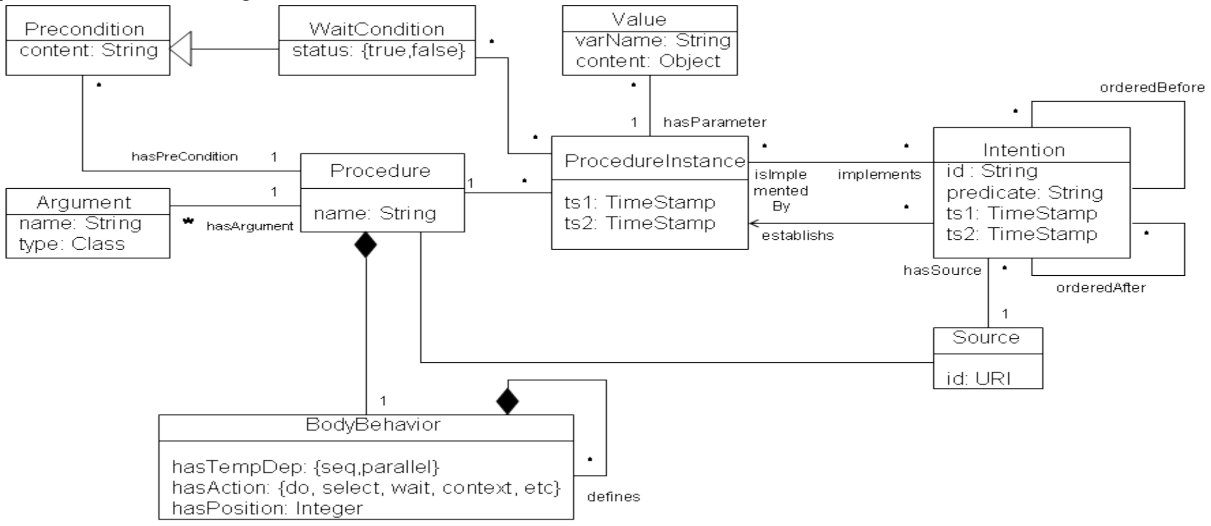


Figure 2: A portion of the action schema used to store task execution information.

necessary to start *T* were met at the time *T* started (possibly including the conditions that led *T* to be preferred over alternative tasks)

- *Termination*: Demonstrate whether one or more of the conditions necessary to terminate *T* has not been met.

This three-strategy approach contrasts with previous approaches to explanation, most of which dealt with explaining inference (Scott *et al*. 1984, Wick & Thompson 1992). Previous approaches generally have not dealt with termination issues, and they also generally have not distinguished between relevance and applicability conditions. These are critical aspects of task processing and thus are important new issues for explanation.

Justifications can be seen and represented as proofs of how information was manipulated to come to a particular conclusion. We have chosen to leverage the Inference Web infrastructure (McGuinness & Pinheiro da Silva, 2004) for providing explanations. Inference Web was designed to provide a set of components for representing, generating, manipulating, summarizing, searching, and presenting explanations for answers from question answering agents in distributed heterogeneous environments such as the Web. At Inference Web's core is an Interlingua for representing provenance, justification, and trust encodings related to answers called the Proof Markup Language (PML) (Pinheiro da Silva, McGuinness, & Fikes, 2006). Inference Web and PML provide our basic building blocks for representing and manipulating information concerning how recommendations are generated and what they depend on. Our work demanded a method for sharing justifications across the many components in CALO. PML provided an Interlingua and it was already being used to encode explanations for answers generated by an array of reasoning and text extraction systems. Prior to this effort, Inference Web and PML had not been used to explain learning results or task execution engines.

PML provides core representational constructs for provenance (source, author, etc.), information manipulation steps (antecedent, consequent, inference rule, etc.), and trust. Inference Web also provides tools for interactive web browsing of PML as well numerous UI tools for presenting summaries, dialogue interfaces, validators, search modules, etc. (McGuinness, et al, 2006). PML documents contain encodings of behavior justifications using PML *node sets*. An OWL (McGuinness & van Harmelen 2004) specification of all PML terms is available, which separates out provenance[1], justifications[2], and trust[3]. PML node sets are the main building blocks of OWL documents describing justifications for application answers published on the Web.

Each node set represents a step in a proof whose conclusion is justified by any of a set of inference steps associated with a node set. A task execution justification is always a justification of why an agent is executing a given task *T*. The final conclusion of the justification is a FOL sentence saying that *T* is currently being executed. There are three antecedents for this final conclusion, corresponding to the three strategies discussed above. Each antecedent is supported by a justification fragment based on additional introspective predicates.

It is important to note that all the task processing justifications share a common structure that is rich enough to encode provenance information needed to answer the explanation requests identified so far. By inspecting the execution state via introspective predicates, explanation components can gather enough provenance information to support a wide range of explanations.

Producing Explanations

Different users may need different types of explanations. In order to personalize explanations, ICEE uses *explanation strategies*. An explanation strategy provides a method for retrieving provenance and inference information from justifications, selecting the information relevant to the user's request, and presenting the information to the user. Given the wide range of questions that a user might want to ask a complex cognitive agent, we conducted a study which helped us to identify which questions users would find more immediately helpful in their daily interactions with a system like CALO. The feedback from these users motivates our choice of supported explanation strategies and follow-up questions, and is further discussed in the next section.

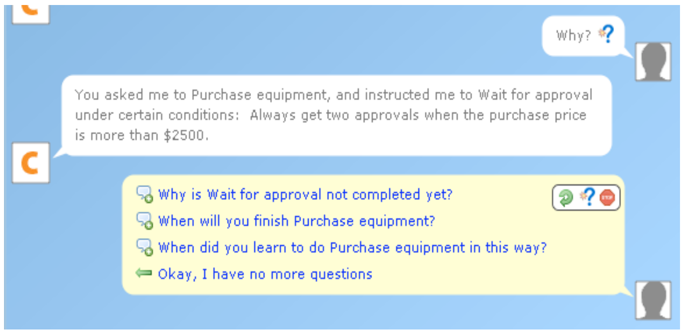
User modeling and strategy selection are handled by the explanation dispatcher. Currently, user modeling is restricted to user preferences. Additional approaches based on user interaction and machine learning techniques are under investigation.

The explanation strategies are closely tied to the explanation request types discussed above. In the example scenario presented earlier, the user asked a question about the motivation for a subtask, and the explanation used a strategy of revealing task hierarchy. Other strategies include providing a task abstraction, exposing preconditions or termination conditions, revealing meta-information about task dependencies, or explaining provenance information related to task preconditions or other task knowledge. See (McGuinness *et al.* 2005) for more details on ICEE's explanation strategies.

ICEE also provides context-dependent follow-up questions for the user. Follow-up questions might include requests

[1] http://iw.stanford.edu/2006/06/pml-provenance.owl
[2] http://iw.stanford.edu/2006/06/pml-justification.owl
[3] http://iw.stanford.edu/2006/06/pml-trust.owl

for additional detail, clarifying questions about the explanation that has been provided, or questions essentially requesting that an alternate strategy be used to answer the original question. Figure 3 shows an example user interface linked to ICEE, in which a list of currently executing tasks is provided to the user. The user has


Why?

You asked me to Purchase equipment, and instructed me to Wait for approval under certain conditions: Always get two approvals when the purchase price is more than $2500.

Why is Wait for approval not completed yet?
When will you finish Purchase equipment?
When did you learn to do Purchase equipment in this way?
Okay, I have no more questions

Figure 3: An example explanation dialogue, with suggested follow-up questions.

requested an explanation of the motivation for a subtask of the first task, and an explanation is provided along with three suggested follow-up questions.

Establishing Trust in Cognitive Agents

To evaluate the use and effect of explanation in cognitive agents, we conducted a preliminary trust study among a set of CALO users. Using a structured interview format, we studied 10 users who performed a variety of tasks with individual CALO agents over approximately two weeks. We had two primary aims for this study:

- To identify what factors users believe influence their trust in a complex cognitive agent.
- To identify which types of questions, if any, users would like to be able to ask a cognitive assistant to better understand the assistant's answers.

Preliminary results from this study show that understanding the results, including any unexpected results (e.g., inconsistencies), and thus establishing trust in a system as complex as CALO, requires a multi-pronged approach. While trust in any system generally requires a series of positive interactions over an extended period of time, the complexity and constant change of an adaptive agent presents additional issues. Many users identified two key factors that would aid in building trust.

The first factor is transparency. The users in our study had varying levels of familiarity with the components of CALO, ranging from an active developer familiar with the algorithms being used in the system, to a user with no prior knowledge of the project. Regardless of their familiarity with the system or their technical background, most users were leery of the opaqueness of the system. When actions were taken by the system for which the underlying computational reasoning was not apparent, the users mistrusted the results. Even when results appeared reasonable, they sought to verify the results rather than trusting them outright, fearful that a result may be anecdotal or based on inappropriate information. These users identified explanations of system behavior, providing transparency into its reasoning and execution, as a key way of understanding answers and thus establishing trust.

The second factor is provenance. Many users commented that, when presented with a result in CALO, they did not know whether they could trust the result because they did not know what source information was used by the system. These users reported that explanations of source provenance would enable them to trust results without the need for much, if any, further verification.

Previous work on building trust recommendations (Zaihrayeu *et al*., 2005; McGuinness *et al*. 2006) has shown the complexity of understanding the notion of trust, and how explanation of deductive reasoning can help users to establish and build trust. We believe that explanation of task execution in cognitive agents, particularly in the presence of learned behaviors and information, can similarly be key to helping users to build trust in these complex assistants. We are continuing to evaluate the results of our trust study, and plan to use the results to guide our future work.

Related Work and Discussion

There has been an abundance of work in explaining expert systems and, to a lesser extent, explaining automated reasoning systems. Most of these have focused on some notion of explaining the deductive trace of declarative rules. Previous work on Inference Web and related work explaining hybrid reasoners added to the field by focusing on settings that are web-based or distributed. Our current work further expands the coverage by using an approach that can support explanations of task executions in the presence of learned knowledge in combination with that of declarative rule processing, along with provenance.

The literature on automated explanation research that explicitly addresses explaining actions is sparse. (Johnson 1994) presents a module to explain a Soar agent's actions by reconstructing the context in which action decisions were made, and then tweaking that context (hypothetically) to discover the elements of the context that were critical to the decision. While Johnson's approach does have some similarities with the way we handle gating conditions, it does not deal with relevance and termination strategies that are important to our agent explanation module. Earlier, (Schulman and Hayes-Roth 1988) developed a BB1 module that explains actions using the architecture's control plan, but it does not address explaining when the control plan does not exist, as is the case in CALO and most other intelligent architectures. Work on plan description (Mellish & Evans 1989, Young 1999, Myers 2006) has focused on summarizing an agent's aggregate behavior (i.e., its entire plan), rather than justifying individual task choices. Our work thus fills an important gap in explaining agent actions, providing fine-grained explanations of a wide range of agent activities, taking into account more aspects of BDI agent architectures, including relevance and termination issues (while using an approach that is compatible with explaining hybrid reasoning components, such as standard FOL reasoners). One promising area of future work would be to allow the user to switch between coarse-grained and fine-grained explanations of agent activity, combining our work with the previous approach.

Our current work is driven by the needs of explaining cognitive assistants. This focus by necessity forces us to address the issue of explanation infrastructures and architectures that can work with task execution systems as well as with deductive reasoners and learning components. This paper has addressed our current progress on designing and implementing an extensible and flexible architecture that is capable of explaining the breadth required by cognitive assistants.

To start this work, we produced an enumeration and categorization of classes of cognitive assistant explanations (McGuinness, et al., 2005), which was further motivated and refined using the initial results of the CALO trust study previously described.

We also analyzed the existing execution environment to identify the actions taken that would need to be explained, leading to a characterization of the SPARK execution environment. We believe that this characterization is representative not just of SPARK, but also of other agent architectures, most notably BDI-based agents and blackboard-based systems. The above characterization is integrated with an explanation architecture capable of explaining standard reasoners and hybrid theorem proving environments. For example, we are currently designing an explanation architecture based on this approach that is to be used in a new system (GILA – the Generalized Integrated Learning Architecture – funded by the DARPA Integrated Learning program (IL 2006)). The program is still in an early stage, but initial indications support our hypothesis of leveragability across adaptive learning agent architectures for broad re-use.

Our architecture and implementation described here demonstrates that an explanation mechanism initially designed for explaining deductive reasoning can also be successfully applied to explaining task-oriented reasoning. Further, we focused efforts on explaining a complex task execution system at the heart of our (and potentially any) cognitive assistant. We developed a three strategy approach to explaining task execution including relevance, applicability, and termination.

Our contributions include key ideas for explaining task processing, such as the distinction between relevance and applicability and the new work on termination. Additionally, work devoted simply to explaining task execution has not traditionally focused on explaining a broader setting including deduction and provenance. We developed a framework for extracting and representing the state and history of a task system's reasoning, a framework that is already proving to be useful for building trust and explanation services in other agent activities.

Work on simply explaining task execution and deduction may not be adequate by itself to establish user trust. Knowledge provenance must also be considered. Our integrated framework includes a component that provides access to sources used and meta-information concerning the sources. The current ICEE system includes initial implementations of all of the components described above, and is seeded with a limited number of strategies, as prioritized by our trust study. Current and future work includes expanding these components to provide more advanced functionality; additional capabilities for handling the results of procedure learning (including learning from instruction and learning from demonstration) that extend and/or update the procedures that the task execution system uses; as well as a strong focus on explaining conflicts, explaining failures, and further explaining knowledge provenance.

Conclusions

If cognitive agents are to achieve their potential as trusted assistants, they must be able to explain their actions and recommendations. In this paper, we presented our explanation infrastructure and describe our implementation for the CALO cognitive assistant. While we focused in this paper on explaining the core task processing component, our solution provides a uniform way of encoding task execution as well as deductive reasoning and learning components as task processing justifications. Additionally, the approach is integrated with the Inference Web infrastructure for supporting knowledge provenance so that explanations derived from task processing justifications may be augmented with source information, improving end-user understanding of answers. Further evaluation has demonstrated that a better understanding of answers by users is a key factor for users to establish trust in cognitive assistants. The primary contributions of this paper are (1) the design and reference implementation of a general and reusable explanation infrastructure for cognitive assistants that integrate task processing, deductive reasoning, and learning; and (2) a supporting design describing the necessary information for explaining task processing in changing environments (including the specification of the relevance, applicability, termination strategy, and the categorization of explanation requests and follow-up strategies), with a focus on changing environments, such as those being updated by learners.

Acknowledgements

We gratefully acknowledge funding support for this effort from the Defense Advanced Research Agency (DARPA) through contract #55-300000680 to-2 R2. We thank Cynthia Chang, Vasco Furtado, Karen Myers, Jim Blythe, Ken Conley, and David Morley for helpful collaborations and feedback on this work.

References

CALO, 2006. http://www.ai.sri.com/project/CALO

Glass, A. and McGuinness, D.L. 2006. Introspective Predicates for Explaining Task Execution in CALO, Technical Report, KSL-06-04, Knowledge Systems, AI Lab., Stanford Univ.

Hayes-Roth, B. 1985. A Blackboard Architecture for Control. *Artificial Intelligence* 26(3):251-321.

Integrated Learning, 2006. www.darpa.mil/ipto/programs/il/

Johnson, W. 1994. Agents that Explain Their Own Actions. *4th Conference on Computer Generated Forces and Behavioral Representation*.

McGuinness, D. L., Pinheiro da Silva, P. 2004. Explaining Answers from the Semantic Web: The Inference Web Approach. *Journal of Web Semantics*. 1(4), 397-413. http://iw.stanford.edu

McGuinness D.L., van Harmelen F. 2004. OWL Web Ontology Language Overview, Technical Report, World Wide Web Consortium (W3C), February. Recommendation.

McGuinness, D.L., Ding, L., Glass, A., Chang, C., Zeng, H., and Furtado, V. Explanation Interfaces for the Semantic Web: Issues and Models. *Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI'06)*. Athens, Georgia, November, 2006.

McGuinness, D.L., Pinheiro da Silva, P., and Wolverton, M. 2005. Plan for Explaining Task Execution in CALO, Tech. Report, KSL-05-11, Knowledge Systems, AI Lab., Stanford Univ.

McGuinness, D.L., Zeng, H., Pinheiro da Silva, P., Ding, L., Narayanan, D., and Bhaowal, M. 2006. Investigations into Trust for Collaborative Information Repositories: A Wikipedia Case Study. WWW2006 Workshop on the Models of Trust for the Web (MTW'06), Edinburgh, Scotland.

Mellish, C. and Evans, R. 1989. Natural Language Generation from Plans. *Computational Linguistics*, 15(4).

Morley, D. and Myers, K. 2004. The SPARK Agent Framework. *3rd International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-04)*, New York, NY: 712-719. http://www.ai.sri.com/~spark/

Myers, K. 2006. Metatheoretic Plan Summarization and Comparison. *International Conference on Automated Planning and Scheduling (ICAPS-06)*.

PAL, 2006. http://www.darpa.mil/ipto/programs/pal/

Pinheiro da Silva P., McGuinness D. L., Fikes R.. A Proof Markup Language for Semantic Web Services, Information Systems, Volume 31, Issues 4-5, June-July 2006, pp 381-395. Also, Stanford Technical Report, KSL-04-01.

Pinheiro da Silva, P, McGuinness, D., McCool, R. 2003. Knowledge Provenance Infrastructure. IEEE Data Engineering Bulletin 26(4), pp. 26-32.

Rao, A.S. and Georgeff, M.P. 1995. BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multiagent Systems, San Francisco, CA.

Schulman, R. and Hayes-Roth, B. 1988. Plan-Based Construction of Strategic Explanations, Technical Report, KSL-88-23, Knowledge Systems Lab., Stanford Univ.

Scott, A., Clancey, W., Davis, R., and Shortliffe, E. 1984. Methods for Generating Explanations. In Buchanan, B. and Shortliffe, E. (eds.), *Rule-Based Expert Systems*, Addison-Wesley.

Wick, M. and Thompson, W. 1992. Reconstructive Expert System Explanation. *Artificial Intelligence* 54(1-2): 33-70.

Young, R. 1999. Using Grice's Maxim of Quantity to Select the Content of Plan Descriptions. *Artificial Intelligence* 115(2).

Zaihrayeu, I., Pinheiro da Silva, P., and McGuinness, D.L. 2005. IWTrust: Improving User Trust in Answers from the Web. Proceedings of the 3rd International Conference on Trust Management (iTrust2005), Springer, Rocquencourt, France, pp. 384-392.