Application of MCL in a dialog agent

Darsana P. Josyula[†,*], Scott Fults[*], Michael L. Anderson[◇,*], Shomir Wilson[*]
Don Perlis[*]

[†]Bowie State University
Bowie, MD, USA 20715
[*]University of Maryland
College Park, MD, USA 20742
[◇]Franklin & Marshall College
Lancaster, PA, USA 17604
{darsana,fults,anderson,shomir,perlis}@cs.umd.edu

Abstract

We report on a natural language agent, originally developed as a command driven interface, that was enhanced with time-dependence, contradiction tolerance, meta-linguistic abilities, and an overall meta-cognitive awareness. We show how these new capacities together can make an AI system's natural language processing more robust and human-like.

1. Introduction

In human-human dialog, if a listener does not hear the speaker clearly, the listener will typically notice the problem and take some action to address it. He might ask the speaker to repeat the statement, or possibly he will ignore the problem and move on in the hope that its meaning can be surmised from later context. Of course, potential problems are not limited to signal reception. Even when a listener hears the speaker, he may not understand what was meant, perhaps because an unfamiliar vocabulary word or an ambiguous phrase was used. Here, too, the listener can notice that there is an issue and consider possible actions that could remedy the situation. Again, one such possible action is to engage in meta-linguistic dialog, i.e., the listener might temporarily put aside the current conversation topic and start a new one *about the previous dialog* in order to ask for clarification.

When faced with such perturbations in dialog, humans very effectively *note* the anomaly, *assess* how to deal with it, and *guide* a response strategy into place. We call this the N-A-G cycle, and we have tried to model such a N-A-G cycle in our artificial dialog agent, Alfred, to help improve the notoriously poor perturbation tolerance of compter dialog systems. We believe a dialog agent that maintains and monitors expectations about the dialog, along with a set of responses that it can choose from if an expectation were to fail, will behave more like a human dialog partner, and will therefore be easier and less frustrating to use. And if success is to be measured by a comparison to human dialog partners, then the agent must also have the ability to converse about the conversation (and language in general) when the need arises.

Our approach involves an ample amount of meta-reasoning: the system must monitor, assess, and in some cases change ongoing reasoning processes involved in maintaining the dialog. We therefore call our implementation of the N-A-G cycle–*the Meta-Cognitive Loop*, or, *MCL*. In addition to MCL, Alfred employs a reasoner called Active Logic that is both time-sensitive and contradiction-tolerant. This reasoner is instrumental in getting MCL to work: time-sensitivity allows examination and manipulation of reasoning processes and a tolerance of contradictions provides a way to monitor expectations. In the following sections we outline exactly how Alfred engages in meta-reasoning and meta-dialog. We describe Alfred in the next section. Then we turn to a discussion of MCL followed by a brief introduction to Active Logic. In the last few sections, we discuss related work and future goals.

This research was supported in part by AFOSR and ONR.

2. Alfred

Alfred is a dialog agent designed to act as an interface between a human user and several different task-oriented domains. Alfred uses a machine-translation approach, transforming user instructions in English into valid, domain-specific commands. For each domain, Alfred has a dictionary that lists all of the domain's possible commands and objects, as well as specifying the command syntax required of that domain. Alfred's basic operation plan (Figure 1) runs as follows: obtain the user utterance; parse the utterance; determine the main verb in

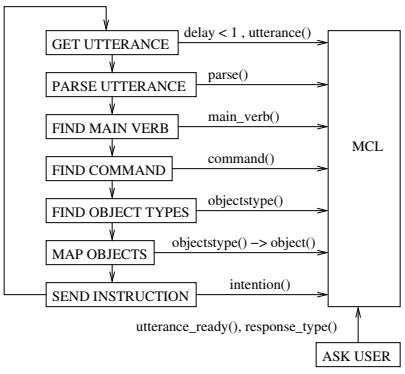

Figure 1: Alfred's plan and sample expectations

the utterance (thus finding the action the user wants performed); find the domain command that best captures the main verb; determine the syntax of the command, including the number and type of objects required for the command to be valid; map objects (nouns, names, etc.) from the utterance onto objects in the domain; construct and send a valid instruction to the domain.

An MCL component in Alfred implements the N-A-G cycle by monitoring a set of time-related, content-related and feedback-related expectations. Time-related expectations set time limits for various things, like completing an action, finding a solution, or continuing a conversation. So, for instance, Alfred has initial time-related expectations that each of the actions in its basic operation plan will succeed in "reasonable" amounts of time. It also monitors the dialog by expecting to receive user utterances within a specified amount of time.[1] Since Alfred interprets all expectation failures as indications of possible failures, if either of these expectations are violated, it will *note* the problem, *assess* the situation and *guide* a response strategy into place. In these cases, it may be that Alfred will say "Please tell me what to do" or ask "Are you there?"

Alfred also has content-related expectations about what the output of its actions will be. For instance, Alfred has the expectation that when it engages in trying to "determine the main verb" of the utterance, it will eventually find the verb and a $main\_verb$ predicate will be asserted. Similarly, for the action "determine the objects", Alfred has the expectation that an $objectstype$ predicate, which specifies the number and type of each object associated with the command, will be asserted. And, for the action "identify the domain objects", Alfred expects that the right number and types of $object$ predicates (as specified by the $objectstype$ predicate) will be asserted.

Once an expectation is met, MCL notes that fact and deletes it from the set of monitored expectations. But if there is an expectation violation, MCL adopts the response strategy associated with that particular violation. In the case where there are multiple strategies specified for a given violation, MCL attempts each one in succession until a resolution occurs. Alfred maintains expectations for any action it takes in response to perturbations, and thus it can monitor and repair its own repairs. This multi-layered expectation monitoring helps make Alfred robust in the face of various kinds of failures, as the following extended example will show. Suppose Alfred's knowledge of a TRAINS-like domain (Allen et al., 1995) is as shown in Figure 2,[2] and the user says "Send the Metro to Boston." However, as can be seen in Figure 2, Alfred does not know the train "Metro". But instead of ignoring the request, or saying "Sorry, I do not understand the word 'Metro'," Alfred engages in the N-A-G cycle in an effort to fix the problem. Since it was incapable of identifying the command's domain object (i.e., the train that goes with the "send" command), it will fail to fulfill the expectation that this action will result in an assertion of $object$ predicates. Alfred easily *notes* this violation, and then *assesses* the problem, determining that the problem lies in the fact that object $V1$ of type $train$, required by the "send" command, is missing. Alfred then *guides* a response strategy into place. First, it checks to see whether it possesses any other knowledge that will allow it to map the new word onto a domain object by examining the train instances in the concept space to see if they are linked to a word "Metro" via a chain of alt_name links. If that fails, (and, there is no such chain in this scenario) Alfred decides to ask the user for help. When communicating with the user, however, Alfred is able to use whatever knowledge it *does* have to determine that the word "Metro" must name a train, allowing Alfred to ask a very specific clarification question: "Which train is Metro?". Whenever Alfred asks a question, it maintains expectations about when the user will reply, and what information the user reply will contain. In this case, the expectations are that the user will provide a known train name within a reasonable amount of time.

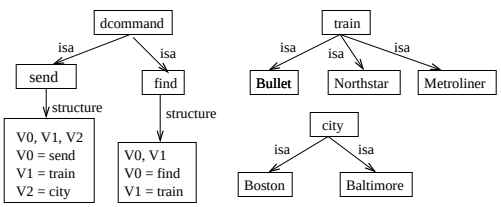

Figure 2: Initial Concept Space

At this point, if the user were to reply with the single word "Subway", Alfred's basic operation plan to deal with user utterances would fail, since there is no verb in this utterance. When the expectation that a $main\_verb$ predicate will be asserted by the "determine main verb" action is violated, Alfred *notes* the anomaly and *assesses* the situation. In this case, since Alfred is expecting a train name from the user, Alfred makes the assumption that the utterance is an elliptical answer to its question, and asserts that the $main\_verb$ of this utterance is "is" (as in "Metro is Subway"). Thus, the "find command" action selects the command $equil$ that adds new associations in Alfreds concept space, and the "identify objects" action chooses the objects to be included in the association as the words "Subway" and "Metro". The execution of the last action in Alfred's plan causes the new association to be stored in the concept space as in Figure 3.

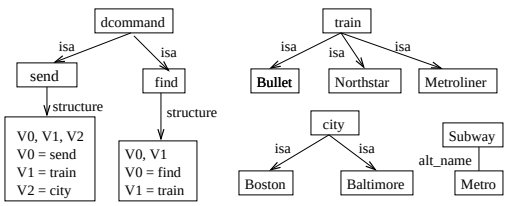

Figure 3: Concept Space after "Metro = Subway"

During all this activity, the *guide* phase of the original loop through the N-A-G cycle caused by the unknown

[1] The actual time duration specified in this expectation can be modified or learned over time.

[2] We call Alfred's knowledge base a "concept space".

word "Metro" is still active and monitoring the original response strategy that it adopted, *ask the user*. Since the strategy helped gather some information from the user, Alfred attempts to use this knowledge to repair the original command, by associating the word "Subway" with one of the known trains in the domain. But this also fails, since there is no train called "Subway" already in the concept space. Hence, the *guide* phase attempts a repair of the repair, and asks the user "Which train is Subway?". Suppose the user now says "Metroliner". Alfred will go through reasoning similar to that for the word "Subway", and store the association between the word "Subway" and "Metroliner" as shown in Figure 4. Since the word "Metro" is now linked to an actual train instance *Metroliner* in the domain via a chain of alt_name links, the "identify domain objects" action will finally succeed, and Alfred will be able to construct a valid domain instruction (send,metroliner,boston) and send it to the correct domain. We now turn to the two components which

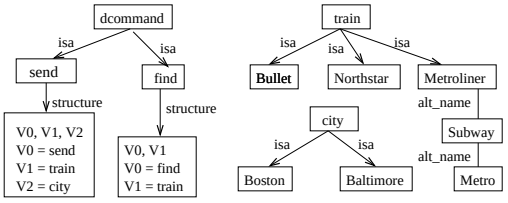

Figure 4: Concept Space after "Subway = Metroliner"

are primarily responsible for Alfred's ability to handle perturbations: MCL and Active Logic.

3. MCL

MCL has been applied to other applications like reinforcement learning (Anderson et al., 2006) and navigating in hostile/unknown environments (Anderson et al., 2007). The success of the MCL strategy on such diverse domains has led us to believe that MCL can be a general add–on component which can sit on top of any host AI system. From there it can monitor the host's expectations, note anomalous behavior, assess it, and guide the host to appropriate responses (and then loop around again if necessary). As a general system, however, it must be prepared for any expectation failure and any (possible) response. Thus, we propose that all errors and their responses, no matter what the system or environment, can be categorized into abstract, domain-independent ontologies.

We have developed a set of three ontologies that aim to do just that (Schmill et al., 2007). They roughly correspond to the three parts of the N-A-G strategy: an ontology of *indicators* for noticing when expectations fail, an ontology of *failures* for assessing the underlying causes, and an ontology of *responses* for choosing an appropriate action that the system can guide into place. Members of the ontologies are represented as nodes, and links between members express relationships between the concepts they represent. Nodes in each ontology are linked to associated nodes in the other ontologies via inter-ontological links.

System-specific knowledge is introduced through two sets of fringe nodes. The first serves as an input to MCL and is connected to the indications ontology. These are essentially a list of all expectations that must be monitored by MCL. The second set of fringe nodes serves as an output of MCL and is connnected to the responses ontology. These are a list of all possible actions the host can employ to address failures. The fringe nodes can be linked by hand into the main ontologies by the system engineer, or the MCL system can go through a training phase in which it learns to associate domain-specific expectations with domain-independent indications, and domain-independent responses to domain-specific actions.

4. Active logic

To implement both Alfred's MCL component, and its main dialog reasoner, we use active logic, a time sensitive, contradiction-tolerant logical formalism (Elgot-Drapkin and Perlis, 1990; Purang, 2001). To have better insight into the operation of Alfred and MCL, it is useful to have some sense of the special properties of this formalism. In active logic, reasoning progresses one step at a time; that is, given A, $A \rightarrow B$ and $B \rightarrow C$ at step 1, B is obtained at step 2 whereas C is obtained only at step 3. This involves step-wise control over inference, and allows the possibility of examining and manipulating the set of formulas at each step in order to modify the reasoning process itself. More generally, new formulas can be added and existing formulas can be deleted at any step to get a revised set of formulas in the next step. This feature can be used to model perception and forgetting, for example.

In addition, in active logic direct contradictions of the type P, $\neg P$ can be detected at each step. In the version of active logic that Alfred is based on, a direct contradiction causes a new formula $Contra(P, \neg P, t)$ to be asserted and causes the existing formulas P, $\neg P$ (as well as the descendants that were obtained in previous steps from P or $\neg P$) to be distrusted, preventing their use in further inference. This is a crucial feature for any logic-based agent intended for deployment in real-world situations. For, as the world changes, what is true at one time may not be true at another, causing inevitable contradictions between what is currently believed and what may be later observed. Since in a classical logical formalism, any well-formed formula follows from a contradiction, the presence of P and $\neg P$ in the knowledge base can quickly lead to swamping the knowledge base with vacuously warranted formulas. Active logic's contradiction-detection feature avoids this swamping problem, and helps maintain a more manageable and trustworthy knowledge base. This feature is especially relevant to our implementation of MCL since an expectation violation is very much like a contradiction—one expects E, but observes $\neg E$ instead. Thus, in active logic, a newly derived $Contra()$ predicate is an indication of an anomaly that needs to be dealt with. As currently implemented, Alfred uses the $Contra()$ predicate in just this way, triggering further reasoning about the causes of and possible remedies for the observed anomaly.

Finally, the step-wise nature of active logic inferences provides a natural way to monitor the passage of time, something that can be important in time-sensitive domains like natural-language dialog.

5. Related Work

The active logic approach to reasoning was motivated in part by the observation that all reasoning takes place in time. Other approaches such as (Bibel, 1998) and (Khalil, 2002) incorporate step-wise, time-dependent reasoning into their systems as well. Active logic, however, allows us to use contradictions as a way of monitoring expectations ($expect\ E$, but $observe\ \neg E$). Thus, active logic serves as the perfect reasoning system for MCL.

The problems that MCL tries to tackle are not new—Making systems that are flexible enough to handle unexpected situations has been a preoccupation of AI researchers for decades.[3] Systems as early as Shakey (Fikes and Nilsson, 1971), for instance, were designed to re-plan when encountered with failures or unmet preconditions. More recently, (Fox, 1995) and (Leake, 1996) have implemented case-based reasoning systems capable of learning about and modifying their own reasoning processes, and (Stroulia, 1994; Ulam et al., 2005) and others associated with Ashok Goel's lab have designed systems that engage in "reflective learning". These systems are quite similar in spirit to MCL; they can compare their own behavior to a self-model to monitor problems. Any differences trigger an analysis of the anomaly, which includes examining the history of the actions that lead to it, localization of the causes, and subsequent modification of the KB and/or reasoning rules. The work of Goel and his colleagues focuses on developing frameworks for representing reasoning that can be analyzed for causes and subsequently modified. Their work, however, is very domain specific. In this way, our work is complementary: our attempt to build a broad, general MCL implementation can build on the insights presented by the depth of their work.

Despite such work in other domains, we have found virtually no previous work using meta-language and meta-reasoning to accurately parse sentences that include meta-linguistic objects like spellings (*His name is spelled E L M O*), quotations (*'Elmo' has four letters*), reference to objects that are part of the discourse (*I didn't catch that last sentence...could you repeat it?*) or meanings (*'Metro' means the same thing as 'Metroliner'*), etc. The closest we have found is (J. G. Neal, 1987), in which the SNePS semantic network formalism (Shapiro, 2000) is used to represent certain knowledge about natural language in an attempt to endow a system with the ability to converse about that knowledge. The system allowed a human to use (a rather stilted form of) English to add syntactic categories and rewrite rules to its KB which could then be used by the system in subsequent natural language processing. But its overall meta-linguistic capabilities were rather limited in that it could only interpret meta-language about syntactic categories and rewrite rules. Furthermore, this system was confined to only talking about its language skills (which might trigger adjusting them), but it did not reason about its language in the way that Alfred does. For instance, Alfred can notice if a word is not in its dictionary, or if it cannot parse a sentence, or if a phrase is ambiguous as to its referent. These skills involve meta-reasoning about language, not just conversing with meta-language.

[3] See (Brachman, 2006) for an enlightening characterization of these problems.

6. Future Work

Our plans for Alfred include adding a new MCL component that is domain-independent (rather than the task-specific MCL that is currently integrated into Alfred's code.) This will make it easier for MCL to monitor all aspects of Alfred's behavior, as well making it easier to add MCL to other dissimilar systems. This domain-independent MCL has the potential to make Alfred a flexible, autonomous dialog agent – but only if we provide Alfred with the ability to "know" when its representations of concepts (specifically, its linguistic concepts) are not wholly appropriate for whatever task is at hand, and thus need modification. That is, an enhanced MCL/Alfred must have meta-linguistic knowledge and skills that permeate every level of language processing, from dialog management to orthographic identification.[4]

This knowledge is important. (Perlis et al., 1998) argues that meta-linguistic talk is perhaps one of the central methods by which human conversation partners manage conversations, and (Anderson and Lee, 2005) reports that more than 50% of dialog management is meta-linguistic. Alfred needs to have significant meta-linguistic reasoning and speaking abilities if it is to fully engage in and comprehend the conversations it has with humans. In order to achieve this level of meta-knowledge about language, Alfred's KB must contain all pertinent information about its language skills and how they have been employed in past conversations. In other words, the grammar of English must be represented *as a concept, in the concept space, using the same formalisms as other concepts*. In this way, Alfred can have concepts like *Metroliner* and *New York* and maintain relationships between those concepts and others. For instance, the former is a train, it is silver, it often participates in events of sending as the thing that is sent, etc. But Alfred must also maintain concepts of the *words* 'Metroliner' and 'New York' in its KB. These are related to the concepts *Metroliner* and *New York*, [5] and yet they are distinct: the word 'Metroliner' starts with the letter 'M', is a noun, and a token of it was used in a previous sentence, etc. But the concept *Metroliner* isn't spelled at all, it isn't a noun, and it wasn't used in a previous sentence (although a token of it would have been invoked when Alfred interpreted the previous sentence that contained a token of the word 'Metro'). This differs from Alfred's current incarnation in that right now Alfred's concept space does not separate the word 'Metroliner' and the idea that it represents, or the sentence "Send Metroliner to Boston" and the idea that it represents. Once this distinction is made, Alfred can truly refer to these items and process sentences like *'Metroliner' starts with the letter 'M'* and *I wasn't supposed to ask you to do that, so ignore the previous command.*

[4] Alfred currently only takes input from a keyboard, necessitating orthographic knowledge rather than phonetic knowledge, but we also have plans to expand Alfred's abilities to include analyzing speech.

[5] We say that the word 'Metroliner' has content *Metroliner*.

In summary, what we want for Alfred is the ability to meta-reason about every step of its dialog management, from interpreting user utterances, fitting them into larger conversational structure, learning new things about language, and forming the actual domain specific command. We believe the capacity to think and speak metacognitively will make Alfred much more human-like.

7. Conclusion

In this article, we have described our most recent efforts in applying MCL and the N-A-G cycle to the task of natural language processing. This project was spurred on by the observation that humans employ the N-A-G cycle, not only in many general circumstances, but also when they are managing conversations. For instance, we maintain a tacit expectation that all words used in a conversation will be in our mental lexicon. We also expect our questions to be followed by answers. When these expectations are not met, we are quick to notice, and assess the situation, and then respond in an appropriate way, often asking our conversation partners for help by engaging in meta-language (i.e., *I don't know that word, could you tell me what it means?*) That is, humans employ meta-linguistic reasoning to monitor and appropriately adjust their language use, and they engage in meta-language speech when it is necessary to maintain a conversation.

We believe that in order for computer dialog agents to be more human-like, they must also be able to monitor, adjust, reason about, and talk about their language skills. To this end, we have added an MCL component to Alfred, our natural language agent. Alfred knows about its language skills. It forms expectations when they are employed, and MCL monitors them. If any are not met, MCL notices, assesses the cause(s), and guides a response into place. That response might mean that Alfred must engage in meta-language with the user, which it is fully equipped to do.

These abilities certainly make Alfred more perturbation-tolerant than other dialog systems; as these abilities become more developed, we hope they will make Alfred more human-like, as well.

8. References

Allen, James [F.], James Hendler, and Austin Tate (eds.), 1990. *Readings In Planning*. Morgan Kaufmann.

Allen, James. F., L. K. Schubert, G. Ferguson, P. Heeman, C. H. Hwang, T. Kato, M. Light, N. Martin, B. Miller, M. Poesio, and D. R. Traum, 1995. The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:7–48.

Anderson, Michael L. and Bryant Lee, 2005. Metalanguage for dialog management. In *16th Annual Winter Conference on Discourse, Text and Cognition*.

Anderson, Michael L., Tim Oates, Waiyian Chong, and Don Perlis, 2006. The metacognitive loop i: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance. *Journal of Experimental and Theoretical Artificial Intelligence*, 18(3):387–411.

Anderson, Michael L., Matt Schmill, Tim Oates, Don Perlis, Darsana Josyula, Dean Wright, and Shomir Wilson, 2007. Toward domain-neutral human-level metacognition. In *Papers from the 2007 AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*.

Bibel, Wolfgang, 1998. Let's plan it deductively. *Artificial Intelligence*, 103(1-2):183–208.

Brachman, Ronald J., 2006. (AA)AI, More Than the Sum of its Parts. *AI Magazine*, 27(4):AAAI Presidential Address.

Elgot-Drapkin, J. and D. Perlis, 1990. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98.

Fikes, Richard E. and Nils J. Nilsson, 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208. Also appears in (Allen et al., 1990).

Fox, Susan, 1995. *Introspective Learning for Case-Based Reasoning*. Ph.D. thesis, Indiana University.

J. G. Neal, S. C. Shapiro, 1987. Knowledge representation for reasoning about language. In R. Jernigan J. C. Boudreaux, B. W. Hamill (ed.), *The Role of Language in Problem Solving 2*. Elsevier Science Publishers, pages 27–46.

Khalil, Hesham, 2002. *Logical Foundations of Default Reasoning*. Ph.D. thesis, University of Leipsig, Leipzig, Germany.

Leake, David B., 1996. Experience, introspection, and expertise: Learning to refine the case-based reasoning process. *Journal of Experimental and Theoretical Artificial Intelligence*, 8(3-4):319–339.

Perlis, D., K. Purang, and C. Andersen, 1998. Conversational adequacy: mistakes are the essence. *Int. J. Human-Computer Studies*, 48:553–575.

Purang, K., 2001. *Systems that detect and repair their own mistakes*. Ph.D. thesis, Department of Computer Science, University of Maryland, College Park, Maryland.

Schmill, Matt, Darsana Josyula, Michael L. Anderson, , Shomir Wilson, Tim Oates, Don Perlis, and Scott Fults, 2007. Ontologies for reasoning about failures in AI systems. In *Proceedings from the Workshop on Metareasoning in Agent Based Systems at the Sixth International Joint Conference on Autonomous Agents and Multiagent Sytems*.

Shapiro, S. C., 2000. Sneps: A logic for natural language understanding and commonsense reasoning. In L. M. Iwanska and S. C. Shapiro (eds.), *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. AAAI Press/MIT Press, pages 175–195.

Stroulia, E., 1994. *Failure-driven learning as model-based self redesign*. Ph.D. thesis, Georgia Institute of Technology.

Ulam, P., A. Goel, J. Jones, and W. Murdoch, 2005. Using model-based reflection to guide reinforcement learning. In *IJCAI Workshop on Reasoning, Representation and Learning in Computer Games*.