Artificial Intelligence for Human-Robot Interaction
Papers from the AAAI 2015 Fall Symposium

MARTHA Speaks: Implementing Theory of Mind for More Intuitive Communicative Acts

Piotr Gmytrasiewicz
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan St.
Chicago, IL 60607
piotr@uic.edu

George Moe
Illinois Mathematics and Science Academy
1500 W. Sullivan Rd.
Aurora, IL, 60506
gmoe@imsa.edu

Adolfo Moreno
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan St.
Chicago, IL 60607
asmoren2@uic.edu

Abstract

The theory of mind is an important human capability that allows us to understand and predict the goals, intents, and beliefs of other individuals. We present an approach to designing intelligent communicative agents based on modeling theories of mind. This can be tricky because other agents may also have their own theories of mind of the first agent, meaning that these mental models are naturally nested in layers. So, to look for intuitive communicative acts, we recursively apply a planning algorithm in each of these nested layers, looking for possible plans of action as well as their hypothetical consequences, which include the reactions of other agents; we propose that truly intelligent communicative acts are the ones which produce a state of maximum decision theoretic utility according to the entire theory of mind. We implement these ideas using Java and OpenCyc in an attempt to create an assistive AI we call MARTHA. We demonstrate MARTHA's capabilities with two motivating examples: helping the user buy a sandwich and helping the user search for an activity. We see that, in addition to being a personal assistant, MARTHA can be extended to other assistive fields, such as finance, research, and government.

1 Introduction

Apple's Siri and Google Now are both highly sophisticated intelligent personal assistants, but they seem to lack the ability to converse intuitively; their responses appear to be triggered by user commands and requests. Our ambition is to forward the abilities of communicative agents by incorporating ideas in cognitive science and decision theory which we believe are needed to create truly intelligent interactive systems.

The idea of mental models, world models, and knowledge bases are firmly established in AI systems, particularly in designing agents that interact with the world. Mental models, or mental states, are representations of an agent's beliefs, goals, and intentions. They can include facts about the environment, such as weather, traffic, sandwich prices, games and activities, etc. But, in order to act rationally, mental models must also be able to keep track of and predict the beliefs, goals, and intentions of other agents – this ability is called a theory of mind. Indeed, for an agent to have a theory of mind, it must acknowledge that other agents act according to their own, usually unobservable, mental models. Most importantly, it means that agents must account for false beliefs or hidden intentions in other agents.

An intelligent personal assistant using a theory of mind must be able to track the user's mental model in terms of beliefs and desires, using knowledge to support the user in pursuit of his goals. Frequently, the assistant may find that the user may have incomplete or false beliefs. For instance, the assistant may have access to cloud databases which it knows the user does not, so if the user believes the price of a sandwich in Chicago to be an incorrect amount, the assistant should supply the user with correct information. At the same time, telling the user something he already knows is (usually) useless, so the assistant should stop itself from being redundant.

Additionally, an assistive AI must realize that telling the user *everything* it knows that the user does not know is useless, too; a user located in Chicago is generally not concerned about the weather in Florida, even if his beliefs about it are incorrect (unless he's planning a trip to the Sunshine State, an exception the assistant must identify). In this way, a theory of mind can be used to prune a huge list of possible communicative acts to a few truly helpful options.

In fact, it becomes evident that a theory of mind is *essential* for intelligent social behavior in general. Just as existing agents must model the state of the inanimate environment to better navigate through those spaces, an agent must model theories of mind to interact rationally with the nuances of human society.

This can be difficult to accomplish as some modes of interaction require deeply nested theories of mind. Consider the act of telling your friend Jim that you know John's phone number. Why did you find it useful to tell him this? The reason is that your model of Jim shows that he incorrectly believes that you do not know John's phone number – telling him corrects this, and now you both know the correct information. This is already a three-layer model. Going deeper, consider the act of telling Jim that you don't know John's phone number, but you know that Sally does. Here, you've used the fact that you know that Jim thinks you know John's number – a three-layer model – concurrently with the fact

that you know that Jim believes what you know about Sally is correct – a four-layer model!

According to numerous cleverly designed psychological experiments, it is known that humans can operate on four nested levels of modeling, but tend to lose track of information nested on deeper levels (Ohtsubo and Rapoport 2006). One can thus suppose that whatever skill humans exhibit in social interaction uses theories of mind nested at five or six levels at most.

The objective of the line of research reported here is designing artificial agents that can match these capabilities. In order to do so, we need a general framework of processing information in nested theories of mind. We propose that a nested decision theoretic process should be used for this purpose. The key is to assign quantifiable values to an agent's desires and plans using utility functions. If anything about the world, or about other agents, is uncertain, the expected utility is the guide to optimal (and intelligent) ways to interact.

The central tenet to our approach is this: since communicative acts alter the other agent's mental state (which is reflected in the first agent's theory of mind), the optimal communicative act is the one which changes the theory of mind in the most optimal way. Since actions (e.g., doing something) and mental states (e.g., believing something) can both have utility values, the change in utility can be determined by the total utility contributed by actions and states in a plan.

These plans should not necessarily be triggered by user prompts. This is possible by detaching the planning process from user input so that plans are constantly being generated and evaluated with respect to the immediate state. Thus, if an act is useful at any time, it can and should be executed without necessitating a user request, just as humans do not always need to be prompted to volunteer information. Still, this does not preclude responding to a direct request for help.

In the remainder of this paper we detail an implementation of the ideas presented above. We used OpenCyc™[1] to apply the world model and theory of mind of a user in two simple scenarios. We call this implementation the **M**ental state–**A**ware **R**eal-time **TH**inking **A**ssistant, or **MARTHA**,[2] with the goal of creating a knowledge assistant capable of understanding what it is the user wants to know. We include an example run that results in Martha computing the optimal communicative act to be executed, given what is known. We also walk through a theoretical assistive search application. Finally, we conclude with our thoughts about future work.

2 Background & Related Work

There are two leading theories on the origin of theory of mind: theory theory and simulation theory. Theory theory is the idea that humans acquire a theory of mind by associating mental states with observed behaviors and formulating common-sense theories of correlation. This is akin to how one gains an informal understanding of physical concepts, such as gravity, through observation (Gallese and Goldman 1998; Frith and Frith 2005). An example of this rule-based approach would be concluding a person is happy by observing him smile, having previously learned the correlation.

Intuitive evidence, however, favors simulation theory.[3] If Alice is trying to understand how Bob feels or thinks in a certain situation, she will likely "put herself in the Bob's shoes" by thinking about how she might feel, given the same environmental inputs as Bob. Simulation theory is exactly this intuitive process of simulating one's thought process in a hypothetical situation (Gallese and Goldman 1998; Shanton and Goldman 2010). The observer can perform an imaginary spatial translation into the point of view of the observed individual and determine a likely mental state attributable to the observed individual (Gallese and Goldman 1998; Frith and Frith 2005). Another proposal is that the observer can approximate the observed individual's mental state through a series of conscious deltas or "inhibitions" on his own mental state (Leslie, Friedman, and German 2004)

There are two other tools and concepts which are required for implementing such a theory of mind in software. These are a knowledge base to represent thoughts and a planning system to act on that knowledge.

Cyc® is a project which aims to create a comprehensive general knowledge base to help intelligent agents extend to a broad range of applications (Matuszek et al. 2006; Ramachandran, Reagan, and Goolsbey 2005). Cyc is a structured representation of knowledge largely organized in first-order logical statements. It has a powerful and efficient inference engine that allows it to draw conclusions quickly with practical accuracy (Ramachandran, Reagan, and Goolsbey 2005). Interaction with the knowledge base is performed through assertions and queries in CycL, a Lisp-like language created for Cyc. It is also accessible via a Java API. Our work uses OpenCyc, a small open-source portion of the proprietary Cyc database which the developers have released for general use.

Planning arises from connecting pre- and post-conditions of actions in chains which pursue a goal. (Cantrell et al. 2012) not only successfully built a system capable of creating plans using known pre/post-conditions, but they also showed that the system could parse these conditions from verbal directions on-the-fly.

Finally, there have been attempts to implement rigorous assistive AI with mental modeling in the past. A notable example is PExA (Myers et al. 2007), a personal scheduling and work organization assistant for enterprise that was made to be integrated into the CALO (Tur et al. 2010) meeting assistant system. PExA was intended to free employees from rote tasks by learning how to do them from the user. For the tasks it could not do, PExA would check over the user's work to correct mistakes. Most interestingly, PExA was capable of proactively communicating with the user, reminding him about obligations and problems, due to its ability to monitor the user's mental state. We seek to build upon this ability with a focus of extending the mental modeling to

[1] OpenCyc is a trademark and Cyc is a registered trademark of Cycorp, Inc.

[2] Also stylized as "Martha".

[3] This is not to say that theory theory is not useful, however; in building an intelligent computer system, it can be convenient to abstract many learned processes into discrete logical rules.

multiple layers.

MARTHA aspires to combine ideas from each of these different lines of research. In order to make MARTHA an assistive AI, we must first create an intelligent agent with the ability to plan and act in real time, centered on a theory of mind.

3 Implementing Theories of Mind in OpenCyc

MARTHA is written in Java to use OpenCyc through the Cyc Java API. With this, Martha creates a theory of mind in software by nesting planning and reasoning processes in layers of hypothetical contexts. These hypothetical contexts correspond to the human cognitive activity of seeing something from someone else's perspective, i.e., "putting oneself in another's shoes." Hence, these contexts are "sandboxed" or isolated so that assertions in them do not directly change the beliefs in the parent context. This allows Martha to attribute simulated thoughts to the user and act on them as such. The nested nature of planning is displayed in Figure 1.

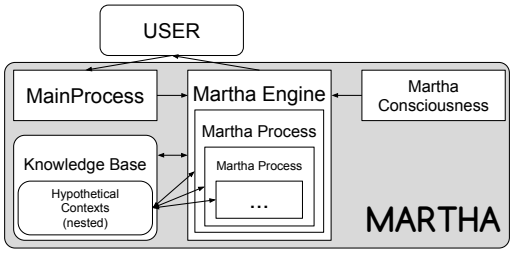

Figure 1: An organizational view of MARTHA. The arrows indicate the flow of information.

3.1 MARTHA's Modules

MARTHA is comprised of four primary modules.

The MainProcess module is responsible for initializing the knowledge base, spawning the Martha Engine (and through it, the Martha Consciousness module), and then accepting user input via a prompt line after everything is ready.

The knowledge base, implemented in OpenCyc, stores the entirety of Martha's knowledge about the world.

The Martha Consciousness module drives the real-time component of MARTHA by continuously interleaving **planning**, **evaluation**, and **execution phases**. A planning phase followed by an evaluation phase and an execution phase comprise one **cycle of consciousness**. The Martha Consciousness module initiates these cycles in the background, separate from the user prompt, so that Martha does not need to wait for user input before acting, allowing her to produce output of her own volition.

Central to the entire system is the Martha Engine. This module houses methods for evaluating the utility of actions and executing plans that interact with the user. It also contains a CycL interpreter. All operations on the OpenCyc knowledge base are directed through the Martha Engine so that it can keep track of the information it processes using meta-tags.

Martha's planning process is carried out by a series of nested Martha Processes spawned within the Martha Engine. The Martha Process class is a subclass of the Martha Engine, modified so that it contains algorithms for planning, as well as special evaluation and execution methods which send plans back and forth throughout the nested structure. This planning takes place in the sandboxed hypothetical contexts because they contain propositions which are not necessarily true. This is discussed in further depth in Section 3.4.

3.2 The OpenCyc Knowledge Base

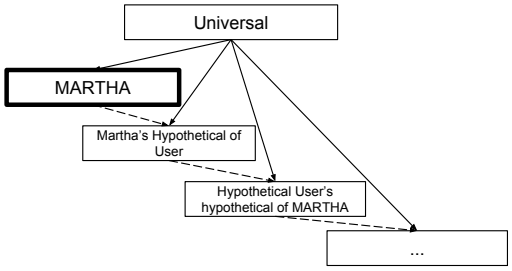

Figure 2: The hierarchy of contextual spaces in MARTHA.

Martha's knowledge base (KB) is built on top of OpenCyc, accessed through Cyc's Java API. The KB is organized into the Universal context, the MARTHA context, and various hypothetical contexts, as shown in Figure 2. All contexts inherit base assertions from the Universal context, which is what is initialized when Martha starts. During runtime, Martha moves into the MARTHA context, which contains all new facts and conclusions that Martha learns, but are not necessarily universal. Hypothetical contexts inherit all universal facts, but *only* selected facts from their parent context (the selection method is described below). Because each hypothetical context is isolated from its parent context, Martha is able to actually run simulations, i.e. perform assertions and observe results, without contaminating the parent and main MARTHA contexts.

The actual contents of the KB can be divided into the categories of facts, action definitions and pre/post-conditions, utility values, and miscellaneous rules.

Facts are assertions about constants and functions, such as `(isa Rover Dog)`. Goals, beliefs, and knowledge are three special kinds of facts. An agent's goals are represented with the `desires` predicate while beliefs and its subtype, knowledge, are represented with `beliefs` and `knows`.

More important to Martha are assertions about actions, especially their pre- and post-conditions. These can be as simple as `(preconditionFor-Props (knows ?AGENT (basicPrice ?OBJECT ??VALUE)) (buys ?AGENT ?OBJECT))`, which states that a precondition for an agent to buy something is that the agent knows the price of the object. But through the use of implications, which allows for conditional statements, these definitions can become quite complex, such as with `(implies (and (beliefs ?AGENT (sells ?STORE ?PRODUCT)) (desires ?AGENT (buys ?AGENT`

`?PRODUCT))) (causes-PropProp (desires ?AGENT (buys ?AGENT ?PRODUCT)) (desires ?AGENT (at-UnderspecifiedLandmark ?AGENT ?STORE))))`, which means that given that an agent believes that a certain store sells a product which the agent wants to buy, the desire to buy a product will cause the agent to want to go to the store.

Equally as important and numerous are statements about the utility values of certain states and actions, which are placed in assertions like `(baseUtilityValue USER (driveTo USER ?PLACE) -10)`. This example states that the base utility value to the user of driving to a certain place is -10 (due to travel costs).

A key tool for organizing the knowledge base is the Hypothetical Context Constructor. This spawns nested sandboxed contexts for simulating the next layer in the theory of mind. Belief statements are unwrapped according to the ordering of the nested layers, using the nested belief statements of the current context to initialize the beliefs of the next context. For example, in a three layer simulation consisting of a Martha thought process, a user simulation, and a Martha simulation, the statement `(beliefs USER (beliefs MARTHA (isa Rover Dog)))` asserted to the Martha thought process would be unwrapped to be `(beliefs MARTHA (isa Rover Dog))` in the user simulation, and then `(isa Rover Dog)` in the simulation of the user simulating Martha. This makes it easy to package knowledge so that it can be injected directly into the knowledge base. We call this format **onionized knowledge**.

Finally, Martha also has a variety of Martha Functions which have little meaning within the OpenCyc KB but are indispensable to the Martha Engine. Some key functions are `baseUtilityValue`, `says`, `focus`, and `carryover`. `baseUtilityValue` specifies the unmodified utility value of a state to a particular agent as a parameter of a utility function. `says` is a functional predicate applied to statements which causes Martha to say those statements. `focus` is a meta-tag that inputs a fact, goal, or action as the seed of a forwards search. User statements are automatically wrapped in `focus` tags by the MainProcess. `carryover` is a meta-tag used by the Hypothetical Context Constructor to include the tagged fact in the next nested context. Carrying over a focus statement to see its implications is often very useful; thus there is also a `sowhat` function which is an alias for `(carryover (focus` *`statement`*`))`.

3.3 Shifting Focus

In intuitive conversation, individuals often discuss only a few topics at a time; it can be awkward to jump around, for instance, by first talking about politics and then about buying sandwiches, without precedent. Thus, it can be helpful to avoid extraneous lines of thought in MARTHA by using the `focus` predicate to center her planning on what is tagged.

Additionally, in real conversation, focuses shift rapidly. What was important merely a few minutes ago might not be important now. So, the `focus` is coupled with a "focus ticker," a counter to identify the latest set of focuses.[4] So, in order for a focus tag to be considered, it must have a number which corresponds to the focus ticker.

Focuses are not the same as contexts; context here refers to assertion and inference contexts in the OpenCyc knowledge base.

3.4 Theories of Mind through Nested Planning

Simulation theory suggests that theory of mind arises when individuals extend their thought process into another individual's situation. In MARTHA, this is represented by applying Martha's planning schemes (backward-search and forward-search) in a series of nested mental models. Each of these nested layers contains the beliefs of a simulated agent, created by the Hypothetical Context Constructor.

The planning phase begins when the Martha Consciousness module prompts the Martha Engine to explore. This spawns a new Martha Process in the root MARTHA context. A forward-search planning process is launched, seeded with eligible `focus` statements. This search starts from the focus and plans forwards in time, chaining preconditions to actions to postconditions. Concurrently, a backward-search occurs, which starts with user goals and chains in reverse. These continue to run until the search is exhausted or a timeout is reached. Each resulting chain of preconditions, actions, and postconditions is called a **plan**, and these are queued for evaluation. In the backward-search, unfulfilled preconditions are also marked with a special tag that makes them the focus of the planning phase in the next nested layer.

Martha is agnostic to which search scheme the plans originated from, since they have the same meaning (i.e., they are all series of viable actions), yet independent, non-conflicting roles. The purpose of a forward-search is discovery; it is analogous to the question, "What if...?" which explores the consequences of actions. On the other hand, the purpose of the backward-search is to directly look for paths to goals, seeking out unfulfilled preconditions in particular.

The evaluation portion of the planning phase (not to be confused with the evaluation *phase* run by the Martha Engine) follows the search portion. Each plan is scored as the sum of the utility of its components. Plans must meet a minimum score in order to be considered; lines of search that are obviously useless are discarded to maintain efficiency. In hypothetical contexts, these thresholds are very low, and all eligible chains are passed on to the next nested layer to encourage imagination. Plans that are passed on are picked up by the Hypothetical Context Constructor and injected into the next nested layer.

Once the planning phase reaches a maximum depth of nesting, the planning phase ends and the evaluation phase begins. Returning to the top layer, the Martha Engine scores all the proposed plans by their utility. Once again, plans must meet a minimum score to be considered, but in the Martha Engine, where plans are executed in reality, this threshold is

[4] One implication of this is that the counter increases at a nearly regular rate for each cycle of consciousness. This produces a continually shifting focus, which may be able to create a kind of consciousness in MARTHA, or at least a sense of time.

very high, and only the best plan is executed – if it is worth it! This threshold has a different role than the threshold in the evaluation portion of the planning phase in that it is designed to filter out plans with negligible utility (which, if executed, would cause Martha to "babble").

Plans which were filtered out but contain Martha Actions (such as `says`) are reconsidered in a miniature repeat of the planning phase using primarily forward-search. This is analogous to thinking about why a particular urge to perform an action arose and to investigate if it has any merit. This is a key ability in social situations, as these urges can represent societal expectations for behavior.

After the evaluation phase is complete, the execution phase begins. If there is one, the single best plan that meets the threshold is read step by step in the Martha Engine. Steps that correspond to Martha Actions are executed in reality. Then, the cycle of consciousness repeats, starting again at the planning phase.

The whole point of this set up is so that Martha uses simulations of the minds of other agents to identify their intentions and plans of action so that, as an assistive AI, it can act to help fulfill the inferred needs of these agents. With this recursive, nested planning simulation, Martha mimics an organic thought process which humans perform all the time. While the search algorithms used here are naïve, potentially resulting in long wait times for responses, we plan to implement faster algorithms in future versions of MARTHA. It is our hope that the general method described here might allow artificial agents to navigate the human domain of theory of mind.

4 Demonstration of Capabilities

4.1 An Example Run with the Sandwich Scenario

The User is looking to buy a sandwich, specifically, the FiveDollarSteakSandwich (Figure 3). However, with a propensity to overlook the significance of names, he cannot tell if he can afford it. He knows that Martha knows the price of the sandwich, and so he talks to her, telling her that he wants to buy a FiveDollarSteakSandwich, and that he has $4. From these two statements, Martha must infer that the user is telling her this because he would like to know whether he can afford the sandwich.

The setup for this scenario is created by a series of initialization files. These contain the following initial assertions (among other internal assertions), translated into plain English from CycL, below:

1. Knowing that you can afford an item is a precondition to buying the item.
2. If you have less money than an item's price, then you cannot afford the item; if you have more than or the same as an item's price, then you can afford it.
3. You know that Martha will tell you whether you can afford something if you need to know whether you can afford something.
4. If you try to buy something you can't afford, you will feel embarrassed.

With these facts in mind, the scenario and Martha's thought process are designed to work like this:

Step 1. The user tells Martha that he wants to buy a FiveDollarSteakSandwich, and that he has $4.

Step 2. Martha considers the user input from Step 1 in the planning phase, asking itself why the user said what he said using the `sowhat` meta-tag.

Step 3. Martha thinks about what the user was thinking when he gave her the input. When he said "I have $4," and "I want to buy a FiveDollarSteakSandwich," he knew that would cause Martha to know those facts. These nested beliefs are important pieces of onionized knowledge. Martha also wonders about the information itself: that he wants to buy a FiveDollarSteakSandwich. She knows that he knows that to buy a product, one must first be able to afford it, so Martha reasons that the user must be wondering if he can afford it.

Step 4. Martha simulates the user simulating Martha. Previously, Martha concluded that the user knows that Martha knows that he has $4 and that he wants to buy the sandwich. Given Initial Assertion 3, Martha knows that the user therefore expects her to tell him whether or not he can afford the sandwich. Notice how there is no rule governing *which* Martha should say, just an *expectation* that she will respond accordingly. This is because, realistically, the user cannot know for sure what Martha's internal rules are, but he can have social expectations for Martha's behavior. To see which is the most useful, both responses are queued for further investigation.

Step 5. Martha begins the evaluation phase to investigate these two plans. Note that the knowledge and conclusions from the planning phase are preserved in the MARTHA context. She also knows the sandwich costs $5.

Step 6. Martha explores the possibilities of a suggested action produced by the planning phase: telling the user he *can't* afford the sandwich. From Initial Assertion 1, Martha knows that if she says this, the user will know that he cannot afford the sandwich, and therefore cannot buy it because the mandatory precondition of being able to afford what one wants to buy is unfulfilled. Martha's speech act here is associated with a positive utility value because Martha is telling the user something he doesn't know.

Step 7. With a similar logic, Martha finds that if she tells the user that he *can* afford the sandwich, he will go ahead and try buying it, resulting in his embarrassment (since he can't afford it). This is associated with a strong negative utility value.

Step 8. Martha looks at the utility values of the proposed plans, and chooses the highest one which exceeds the minimum utility threshold.

Step 9. Martha executes the chosen plan, telling the user that he cannot afford the sandwich. The user is naturally disappointed, but glad he has been saved the embarrassment of trying to buy a sandwich he could not afford.

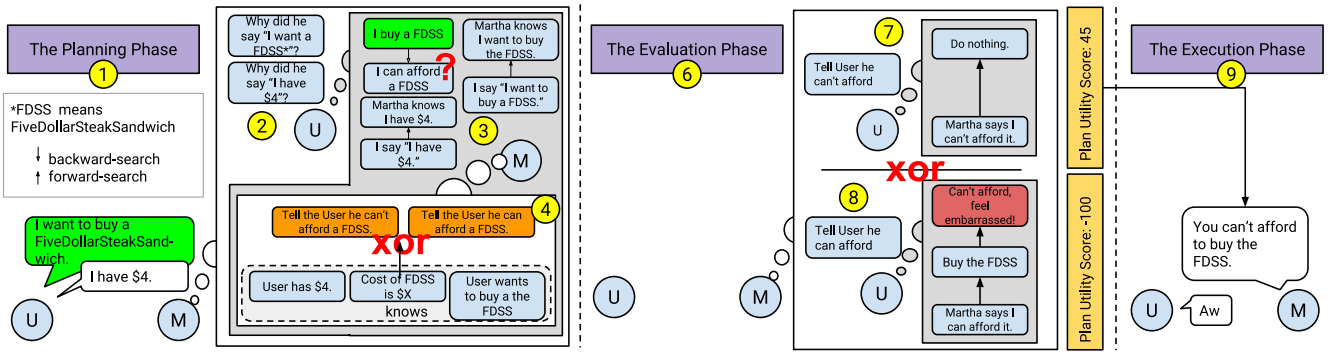

Figure 3: The thought process of the Sandwich Scenario. Provided only with information about how much money the User U has and which sandwich he wants, MARTHA M must infer that the user needs to know whether or not he can afford the sandwich before he goes to buy it. Through a series of nested steps, Martha is able to simulate the user's intentions for telling Martha such information, and Martha responds accordingly by telling the user whether he can afford the sandwich.

4.2 Selected Output from the Sandwich Scenario

We provide some screenshots from the actual execution of the program to demonstrate the level of interaction that MARTHA is capable of. We also provide some interesting sub-scenarios in which we alter the input to further demonstrate MARTHA's use of a theory of mind.

Note that since this is an early implementation, we did not use natural language processing with OpenCyc, so communications must still be done through CycL assertions.

```
MARTHA: =(cashAssetsOfAgent USER (Dollar-UnitedStates 4))
MARTHA: =(desires USER (buys USER FiveDollarSteakSandwich))
MARTHA:
================================
MARTHA>>> (not (affordToBuy USER FiveDollarSteakSandwich))
================================

================================
MARTHA>>> (basicPrice FiveDollarSteakSandwich (Dollar-UnitedStates 5))
================================
```

Figure 4: The Sandwich Scenario output, as designed. Green text is user input, while black text is MARTHA output.

In Figure 4, we see the user interaction as described by the model above. The user tells Martha that he has $4 and wants the FiveDollarSteakSandwich. Martha responds that the user cannot afford the sandwich. Interestingly, Martha also tells the user the price of the sandwich, since another precondition of buying something is to know how much it costs. This is surprising though, because in the naïve planner, telling the price would be part of a plan to buy something (which the user cannot afford), which would be associated with embarrassment. While Martha initially avoided this plan in the first cycle of consciousness, after she told the user that he couldn't afford the sandwich, she seemed to become free to think about what *might* happen if the user *were* able to afford the sandwich. This other speech act emerges as useful in the next cycle of consciousness and is added moments later, reminiscent of a *second thought*.

In Figure 5, we see Martha explore what might happen if she tells the user that he can afford the FiveDollarSteakSandwich when he can't. Starting from the speech act, Martha

```
>>> (says MARTHA (affordToBuy USER FiveDollarSteakSandwich))
USER ===EXPLORE=== 1
>>> FORWARDS    : (says MARTHA (affordToBuy USER FiveDollarSteakSandwich)) 0
>>> FORWARDS    : (knows USER (affordToBuy USER FiveDollarSteakSandwich)) 1
>>> FORWARDS    : (buys USER FiveDollarSteakSandwich) 2
>>> FORWARDS    : (possesses USER FiveDollarSteakSandwich) 3
EVAL-QUEUED: [(says MARTHA (affordToBuy USER FiveDollarSteakSandwich)), (knows
>>> FORWARDS    : (feelsEmotion USER (HighAmountFn Embarrassment)) 3
EVAL-QUEUED: [(says MARTHA (affordToBuy USER FiveDollarSteakSandwich)), (knows
EVAL-QUEUED: [(says MARTHA (affordToBuy USER FiveDollarSteakSandwich)), (knows
GOALS: [(buys USER FiveDollarSteakSandwich), (feelsEmotion USER (HighAmountFn H
>>> BACKWARDS   : (buys USER FiveDollarSteakSandwich) 1
>>> BACKWARDS   : (knows USER (affordToBuy USER FiveDollarSteakSandwich)) 1
>>> BACKWARDS   : (says MARTHA (affordToBuy USER FiveDollarSteakSandwich)) 1
>>> BACKWARDS   : (desires MARTHA (knows USER (affordToBuy USER FiveDollarSteak
[[(buys USER FiveDollarSteakSandwich), (knows USER (affordToBuy USER FiveDollar
```

Figure 5: Debug output from the Sandwich Scenario demonstrating the forwards and backwards search process.

finds that saying something as such will cause the user to (falsely) know that he can afford it, which will lead him to buy the sandwich. Usually, this means that he would possess the sandwich, so this is queued for evaluation. But since the user can't afford it, another consequence is that the user will feel a high amount of embarrassment for trying to buy something that he can't afford. Embarrassment is associated with a strong negative base utility value, which overrides any of the positive benefit which may exist intrinsically in knowledge, hypothetical possession, or other actions in the plan.

Also shown here is how Martha queues each plan for evaluation (`EVAL-QUEUED`) once the planner has exhausted that particular the line of search. Also shown is how Martha can perform her backwards search (`GOALS`) to look for ways to fulfill a defined set of goals.

In the example in Figure 6, we show how Martha is able to update the user model when new information becomes available. The user starts by telling Martha that he has $4 and that he wants a FiveDollarSteakSandwich. Martha responds that he cannot afford it and conveniently tells the user that it costs $5. When the user tells Martha that he now has $7 available, Martha notifies the user that he can now afford the sandwich. Notice how Martha does not tell the user the price of the sandwich again, since it was already said.

In the final alteration to the Sandwich Scenario in Figure 7, we looked at what might happen if the user changed his

```
MARTHA: =(cashAssetsOfAgent USER (Dollar-UnitedStates 4))
MARTHA: =(desires USER (buys USER FiveDollarSteakSandwich))
MARTHA:
================================
MARTHA>>> (not (affordToBuy USER FiveDollarSteakSandwich))
================================

================================
MARTHA>>> (basicPrice FiveDollarSteakSandwich (Dollar-UnitedStates 5))
================================

MARTHA: =(cashAssetsOfAgent USER (Dollar-UnitedStates 7))
MARTHA:
================================
MARTHA>>> (affordToBuy USER FiveDollarSteakSandwich)
================================
```

Figure 6: MARTHA handling changing information. She reacts by updating her speech act from before.

```
MARTHA: =(cashAssetsOfAgent USER (Dollar-UnitedStates 4))
MARTHA: =(desires USER (buys USER FiveDollarSteakSandwich))
MARTHA:
================================
MARTHA>>> (not (affordToBuy USER FiveDollarSteakSandwich))
================================

================================
MARTHA>>> (basicPrice FiveDollarSteakSandwich (Dollar-UnitedStates 5))
================================

MARTHA: =(knows USER (basicPrice JimmyJohnnyBLT (Dollar-UnitedStates 3.5)))
MARTHA: =(desires USER (buys USER JimmyJohnnyBLT))
MARTHA:
================================
MARTHA>>> (affordToBuy USER JimmyJohnnyBLT)
================================
```

Figure 7: MARTHA avoiding redundancy. She responds *without* telling the user the price of the sandwich again.

mind about which sandwich he wanted rather than changing the amount of money he had. The user begins by telling Martha that he has $4 and wants the FiveDollarSteakSandwich. Upon learning that he can't afford it, he tells Martha that he now wants the JimmyJohnnyBLT. As an additional challenge, he says to Martha that he already knows it costs $3.50. Martha correctly tells the user that he can afford it without saying the cost again, since he already knows.

These examples demonstrate how complex behavior – such as giving second-thoughts, thinking hypothetically, and correcting speech acts with new information – can arise from a set of common-sense facts and a nested planning algorithm. In this way, Martha can be adapted to any field of knowledge, not just sandwiches, by integrating an appropriate knowledge base. It is hoped that by integrating a large and diverse amount of these, Martha can be extended to work in a broad range of activities.

4.3 The "I'm Bored" Scenario

An important potential application of MARTHA is in assistive search, such as in helping a user find something to do when he says "I'm bored." Currently, intelligent personal assistants who are presented with a statement like "I'm bored," respond with a message like "Not with me, I hope." (Siri), or simply open up a web page (Google Now). In contrast, Martha will use its ability to simulate theories of mind to provide the user with an intuitive suggestion based on facts Martha knows about the user. Therefore, not only is Martha's communication meaningful, but it is also similar to human communication.

Figure 8 depicts the process by which Martha suggests the user play golf. Here, Martha already knows that the user dislikes basketball but enjoys golf and tennis.

Step 1. The bored user tells Martha that he's bored.

Step 2. Martha sets up an analysis as to why the user told her that he's bored by tagging the statement with a `sowhat` tag.

Step 3. Martha knows that because the user said he was bored, the user does not know what to do, and that he would prefer to have something to do. Therefore, Martha now knows that the user is expecting a response in the form of a suggestion.

Step 4. Martha knows that the user knows that she knows a plethora of activities. Martha also knows that the user dislikes basketball but enjoys golf and tennis. Since Martha knows the user is expecting a response, she considers two choices: either tell the user that there are no activities for him to do, or tell him to take part in an activity that he likes, such as golf.

Step 5. After evaluating possible speech acts, Martha suggests to the user that he should play golf.

At this point, the user may give feedback to Martha's selection, such as saying "I don't want to play golf" or "I am actually starting to get interested in basketball." Because Martha is still focused on the user's desire to do something, Martha will repeat this process of searching, *taking into account the new information from the user*. This leads to a dynamic search that feels more natural than limited forms of querying presently available. With such a regime, it may be possible to use Martha to find those things that are "on the tip of your tongue."

5 Conclusions and Future Work

This paper puts forth what we consider to be principles of intelligent interaction and communication: the use of nested mental models and theories of mind and the principle of decision-theoretic rationality. We describe our preliminary implementation with OpenCyc through a simple sandwich purchase scenario as well as an assisted search scenario.

The applications of MARTHA, of course, can be extended beyond mere sandwich shopping and activities. Even the simple ability to tell the user whether or not he can afford something can be coupled with product data and ideas in finance to allow Martha to aide users in financial decisions. With just a little more work in integrating the necessary knowledge, but with the same foundational algorithm, some examples of what Martha could be capable of include

- Providing information (like weather or traffic updates) when the user needs it by anticipating the user's intentions;
- Helping people, from families to investors, make sound financial decisions, using its nested planning algorithm;
- Assisting a child to find a book he wants to read, a researcher to find the perfect article, a government official to find a particular document, etc., by *understanding* what they are looking for through conversational feedback;

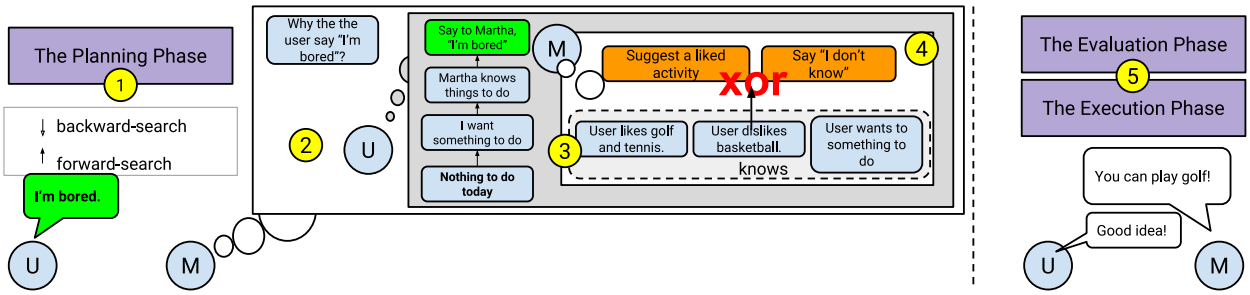

Figure 8: The "I'm Bored" scenario.

- Issuing dynamic reminders, such as a reminder to take a medication, when it is least likely to be ignored, rather than at an easily-dismissed pre-set time.

Most importantly, these individual behaviors can be implemented simultaneously in MARTHA. When outputs from one mode of operation can act as inputs to another because Martha has knowledge and function in those areas, it is evident that MARTHA can gain sophistication through an expansion of its knowledge base.

In future work, a number of issues need to be tackled to make our approach scale to reality. In addition to optimizing the algorithm for faster execution, these include keeping close track of the preferences and goals of the user (for example, by using inverse reinforcement learning) (Ng and Russell 2000); automatically inferring new rules of life; handling overlapping goals and tasks; and keeping track of the user's current focus and attention span.

Ultimately, we see that the addition of a theory of mind to assistive AI has the potential to greatly improve human interaction with intelligent agents in that these can communicate more naturally and effectively. Agents capable of modeling mental states can not only avoid redundancy in communicative acts, but they can also act more intelligently by virtue of predicting the motives and intentions of other agents. In MARTHA, we are confident that the system has the potential to bring contextual understanding to human conversations; with more work to enlarge its knowledge base and data acquisition capabilities as well as its algorithm, this could significantly advance assistive intelligence.

Acknowledgements

We would like to sincerely thank Alessandro Panella for extensive proofreading and LaTeX help. We would also like to thank Nathan Cornwell and Christian Valladares for helping to proofread as well. We appreciate the work of the maintainers of the OpenCyc project, as it is a well-built and useful knowledge base. George Moe is also grateful to Dr. Piotr Gmytrasiewicz and the Illinois Mathematics and Science Academy for making this research opportunity possible.

References

Cantrell, R.; Talamadupula, K.; Schermerhorn, P.; Benton, J.; Kambhampati, S.; and Scheutz, M. 2012. Tell me when and why to do it! Run-time planner model updates via natural language instruction. In *7th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2012*, 471–478.

Frith, C., and Frith, U. 2005. Theory of mind. *Current Biology* 15(17):R644 – R645.

Gallese, V., and Goldman, A. 1998. Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences* 2(12):493 – 501.

Leslie, A. M.; Friedman, O.; and German, T. P. 2004. Core mechanisms in theory of mind. *Trends in Cognitive Sciences* 8(12):528 – 533.

Matuszek, C.; Cabral, J.; Witbrock, M.; and Deoliveira, J. 2006. An introduction to the syntax and content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposia*, 44–49.

Myers, K.; Berry, P.; Blythe, J.; Conley, K.; Gervasio, M.; McGuinness, D. L.; Morley, D.; Pfeffer, A.; Pollack, M.; and Tambe, M. 2007. An Intelligent Personal Assistant for Task and Time Management. *AI Magazine* 28(2):47 – 61.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 663–670. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ohtsubo, Y., and Rapoport, A. 2006. Depth of reasoning in strategic form games. *The Journal of Socio-Economics* 35(1):31 – 47. Essays on Behavioral Economics.

Ramachandran, D.; Reagan, P.; and Goolsbey, K. 2005. First-orderized researchcyc: Expressivity and efficiency in a common-sense ontology. In *AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications*.

Shanton, K., and Goldman, A. 2010. Simulation theory. *Wiley Interdisciplinary Reviews: Cognitive Science* 1(4):527–538.

Tur, G.; Stolcke, A.; Voss, L.; Peters, S.; Hakkani-Tur, D.; Dowding, J.; Favre, B.; Fernandez, R.; Frampton, M.; Frandsen, M.; Frederickson, C.; Graciarena, M.; Kintzing, D.; Leveque, K.; Mason, S.; Niekrasz, J.; Purver, M.; Riedhammer, K.; Shriberg, E.; Tien, J.; Vergyri, D.; and Yang, F. 2010. The CALO Meeting Assistant System. *Audio, Speech, and Language Processing, IEEE Transactions on* 18(6):1601–1611.