On Short-Term and Long-Term Memory for Brahms Agents

Maarten Sierhuis [1]
Ron van Hoof [2]

RIACS-NASA/Ames Research Center [1]
QSS group, Inc [2].
Computational Science Division, MS 269-1
Moffett Field, California 943055
msierhuis@mail.arc.nasa.gov

Abstract

Brahms is a multi-agent modeling language for developing distributed multi-agent systems, as well as multi-agent simulations. One of the issues we face developing Brahms agents is that the belief-set of an agent increases over time, slowing down the agent's processing capability. Another issue is how an agent can "remember" what happened in the past. How can an agent's memory be modeled sufficiently so that it can remember what activities were performed and how to continue a task that was not finished the day before? How does the agent "remember" the context of activities from the past? A design and use of agent memory is described that includes both short-term and long-term memory based on context, as well as the use of such memory to handle the growing set of beliefs. The interaction between short- and long-term memories over time is also discussed.

Introduction

"Memory" of Brahms agents is currently limited to a sequential set of first-order beliefs (we call this the belief-set of an agent). For a long time we have known that a growing and changing belief-set becomes a problem when we want agents to "remember" the past to decide what to do in the future. This is most evident when we model multiple "days" for agents. How do agents remember what is done in the past, if the work of the agent the next day is similar to the day before? Currently, beliefs are overridden and agents "forget" what they did in the past. What is "remembering"? How can memory be modeled sufficiently, so that agents can remember what activities were performed the previous day? How can an agent continue to work on a task that was stopped, but not finished the day before? How does the agent "remember" the context of activities from the past? It is obvious that representing the past as beliefs is not sufficient. This paper discusses how we can extend the memory for Brahms agents by separating memory into long-term and short-term memory for agents. Two questions are answered; How is long-term memory represented, and how does the agent use this long-term memory in its process to decide what activity to perform at each moment in time? In other words, how does the agent remember the past and how can the remembered past influence current activities? How does the past become part of the agent's current context? We have drawn on two books in which the notion of human memory is described (Clancey 1997) (Clancey 1999).

Clancey describes in what way the most common view of memory in the AI and cognitive science communities is wrong. Clancey discusses how knowledge representation is not equal to knowledge, contrary to what the two communities argued for years. Human memory is not simply a form association, but rather it has to do with situated context. In particular, Clancey writes that the conclusion is that what is remembered depends on the context, or better, what is experienced depends on the context (Clancey 1997). To make sure we are not misunderstood, Brahms is an associative knowledge representation, and we would never claim a Brahms agent model to be equal to a model of human cognition. Instead, a Brahms agent model is a representation of human behavior at some abstraction interpreted by the modeler. In this paper we present a newly designed capability for Brahms, allowing an agent to store historical context as that what is "experienced". One design idea is that the agent could store every workframe instantiation[1] (WFI) and its bindings into some long-term memory for the agent. The agent will then be able to recall past WFIs from the long-term cache and associate it with the current WFI of the same workframe, thus relating previous activity executions and the context in which they took place with the current activity.

Based on our experiences using Brahms as an agent development environment implementing and testing an agent-based architecture for planetary extra-vehicular activity (EVA)—the Mobile Agents Architecture—we identified a second need for long-term memory. As the number of beliefs about similar types of concepts (i.e. groups or classes and their attributes) increases over

[1] A workframe is a situated action rule with preconditions and a body that can consists of any number of sequential conclude and action statements. A workframe instantiation is an object-instance of a workframe in which the preconditions have matched to the beliefs of the agent and a workframe context of bounded variables has been created.

time—this can grow very quickly as new objects are dynamically created in the system—the matching of beliefs in preconditions of rules (i.e. workframes and thoughtframes) takes exponentially longer. During a recent field test in the Utah dessert this became problematic so quickly that the agents took minutes to respond to a single request from the human astronaut (Clancey et al. 2004). To resolve this issue we implemented an effective, but limited solution. We implemented retraction of beliefs at the moment the modeler was confident that the agent would not need that belief in the future to perform its activities. In this way the belief-set of agents was kept to a minimum, and reasoning time stayed constant. We should note that although retraction of beliefs seems like a hack, retraction does have benefits in agent-based systems to remove those beliefs that are not relevant for an agent's belief set and are only used once.

In this paper, we describe a design of agent memory representation and use that includes both short-term and long-term memory based on context, as well as the use of long-term memory to handle the growing set of agents' beliefs. We also discuss the interaction between short- and long-term memories over time.

Related Work

Research on human memory is a "hot" topic in cognitive science and cognitive neuroscience. In neuroscience there has been an insurgence of interest in neuroimaging studies of working memory (Beardsley 1997). Studies have revealed that various brain areas in the prefrontal cortex work together to produce *working memory.* Almost 30 years ago, cognitive psychologist Alan Baddley introduced the concept of working memory as a framework for understanding what was then called short-term memory. He proposed that short-term memory is part of a "working memory" that briefly stores and processes information that is needed for reasoning (Wickelgren 1997). Brain imaging techniques are not advanced enough to uncover the most detailed interaction between short-term memory and the executive regions in the brain. Today, computational models of working memory are pushing the research on a theoretical level. There are a number of well-developed models of working memory that are quite diverse (Miyake and Shah 1999).

In this paper we borrow the concept of working memory for the design of a memory for independent parallel software processes called agents. We note that even though we borrow the memory framework from cognitive psychology, we do not claim that our agent memory design is an implementation of a theory of human memory (cf. (Lewis 1999)). We merely designed this framework to solve practical issues that we ran into when executing our agents over extended periods in real-world systems.

Brahms

Brahms[1] is a multi-agent modeling and simulation environment a) for developing simulations of organizations and systems (Sierhuis and Clancey 2002) (Sierhuis et al. 2003) and b) for designing and implementing multi-agent software systems (Clancey et al. 2003) (Clancey et al. 2004). The Brahms environment includes a multi-agent language, compiler and virtual machine (VM), as well as a development environment and a post-execution viewer of agent execution and interaction. Brahms agents are BDI-like agents in that they have a belief-set and a set of rules that are executed based on a forward-inferencing mechanism (see Figure 1). An agent's procedural memory contains two types of rules, workframes and thoughtframes. Workframes are situated action rules that execute activities. Activities can be primitive or composite (decomposed into more workframes and thoughtframes) and have duration. Activities can be interrupted by higher-priority activities and are resumed when no other higher-priority workframes fired (Clancey 2002). Thoughtframes are pure production rules used for reasoning (i.e. changing the belief-set of the agent). Brahms agents are reactive in the sense that they can detect changes in the world-state (facts) and turn this into beliefs, which in turn can fire new workframes and/or thoughtframes.

Architecture

In this section we describe a new architecture of memory for Brahms agents.

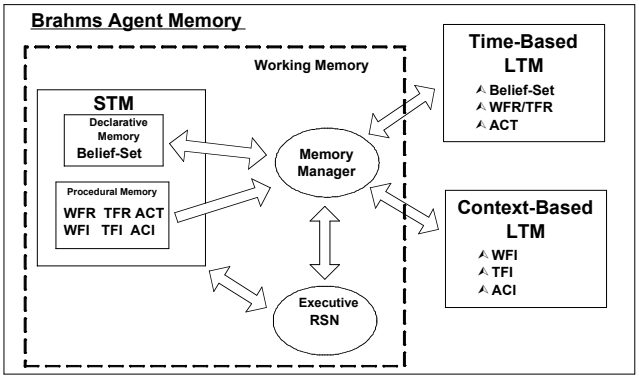

Figure 1. Agent memory architecture

In the current Brahms agent working memory architecture an agent has three types of memory, namely a chronologically ordered belief-set (declarative memory), a set of situated action rules, and a set of production rules (procedural memory). In terms of short-term memory

[1] For more information and for downloading Brahms, go to the Brahms web-site at http://www.agentisolutions.com

(STM) and long-term memory (LTM), we could say that currently agents only have a STM. The proposed new memory architecture of an agent will consist of a STM and two separate LTMs (see Figure 1). In the next three sections we will in turn describe these new STM and LTM.

Short-Term Memory

The STM of an agent consists of the belief-set of the agent plus all its workframes (WFR), thoughtframes (TFR), activities (ACT) and their instantiations (WFI/TFI/ACI). The Executive contains the compiled reasoning state network (RSN). Each agent actually has two RSNs; one for belief-based rules and one for fact-based rules. The Executive is used for an agent's activity activation and reasoning. Beliefs in the belief-set are used to match against preconditions of workframes and thoughtframes in the compiled RSNs. As soon as a belief enters the belief-set of the agent its reasoning process is started. The STM also contains all workframe instances (WFI), thoughtframe instances (TFI) and activity instances (ACI) that are created within the reasoning process. STM and LTM interact by a) moving beliefs from STM to a time-based LTM and b) allowing access of the context of historical frame and activity instances in a context-based LTM (see Figure 1). The WFI, TFI and ACI instances are created by the Executive through the triggering of workframes and thoughtframes, based on matching preconditions to beliefs in the belief-set. The context of a WFI/TFI/ACI in STM is stored in LTM after its completion (see section on context-based LTM).

Besides the addition of LTM, we also need to add a context stored with beliefs in STM. The current VM stores agents' beliefs as a data record of Object-Attribute-Value (OAV) triplets, including their creation time. The belief-set is ordered on belief creation-time, with the most recent OAV overriding the previous OAV for the same OA. However, currently agents have no access to the creation-time property stored in the belief-set.

Contrasting current Brahms agents' memory to human memory, it is natural for humans to know how they acquired a certain belief. People remember if a certain belief was acquired through reasoning, told by someone else or an observation in the world. Often, people remember when and where they acquired a belief, for example as in "Joe told me last night, when I was at his house having a drink, that he bought a new car." In the new memory architecture, Brahms agents will have a context stored with their beliefs in their STM belief-set. Using this context, the agent can access similar type of how, when, where belief information[1]. The context stored with beliefs includes the following:

Belief context

- Time at which it was created
- Agent that created the belief (this can be another agent)
- Activity instance in which it was created (this can be an activity of another agent)
- Workframe instance in which it was created
- Thoughtframe instance in which it was created
- Location of agent

Storing belief context is one thing, but being able to retrieve the context is another. We propose to add reflexive methods to a belief object that allow the agent to access the stored context of beliefs. Belief objects will be first-order type objects and the agent can retrieve belief objects from STM and then use the reflexive methods to access the context properties. We describe this further in the section "Retrieving elements from LTM."

Time-based Long-Term Memory

The time-based LTM (TLTM) is shown at the right top in Figure 1. It contains those beliefs that are removed from the STM belief-set and "deposited" into LTM. When a belief is deposited into TLTM it is no longer used in the reasoning process, and thus it will not match against preconditions. This mechanism allows for efficient reasoning over time. As the number of beliefs of an agent grows the reasoning stays efficient, because beliefs are moved back and forth between STM and TLTM. TLTM also contains those workframes, thoughtframes and activities that have been removed from STM. Workframes, thoughtframes and activities in TLTM are also not used in an agent's reasoning process. Using this mechanism we can model and agent's "forgetting" of how to perform certain activities.

Moving elements from STM to TLTM is different than removing them completely from memory, because removing them from memory means that an agent can never access them again. Even more, it means that an agent can never know that it once had these elements in STM. How elements in TLTM are moved from STM to TLTM is discussed next.

There are many ways in which we can implement beliefs being moved from STM to TLTM. Whatever the mechanism chosen, it needs to be based on some verifiable theory of (human) memory if we want to make the claim that this mechanism is an appropriate one. As is being done with cognitive modeling environments, such as Soar (Newell 1990) (Lewis 2001) and ACT-R (Anderson and Lebiere 1998), we would need to perform psychological laboratory experiments with (human) subjects and match their result to agent models of similar behavior. However, we remind the reader that our Brahms research is not about developing a theory of the mind and human memory.

[1] We again stress the fact that we do *not* claim that human memory "stores" belief-context information that is "retrieved" in the same or a similar manner. There is no known correlation between our suggested method of storing belief-context and human memory.

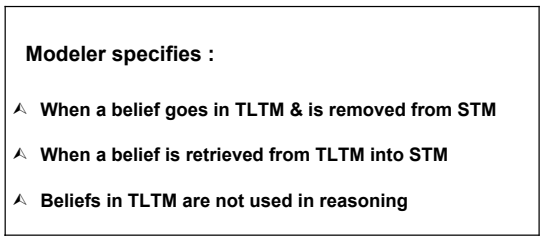

Figure 2. Moving STM to TLTM

Instead, we are interested in providing the Brahms modeler the capability of developing ad hoc models of agent memory. Our approach is to give the modeler all the needed mechanisms to implement a model of memory that either is or is not based on a verified theory of memory. The approach we take is to implement a mechanism for moving memory elements from STM to LTM, but leave the decision of implementing a theory of memory to the modeler (see Figure 2). We do this by providing the modeler with a new capability of assigning properties to STM elements that declare when these elements are to be moved to the TLTM. Each STM element has an access counter. The access counter is a time-constraint of how long elements can stay in STM if they have not been accessed in the reasoning process. In other words, if an element is not accessed in the reasoning process for the specified time (in clock-ticks) the element is moved to TLTM. If, however, the element is accessed within the specified time the access counter is reset. As shown in Figure 2, for beliefs this constraint can be provided in belief assert[1] statements.

There are other ways besides initial beliefs and asserts, agents can get new beliefs; a) through detection of facts and b) beliefs received through communication with other agents. For beliefs created by detectables we can allow the modeler to specify an access counter constraint in the detectable. For beliefs received through a communication activity there are two possibilities. First, if the agent receives the belief from a receive transfer-definition in one of its own communication activities the modeler can specify the access counter constraint in the transfer definition. Another possibility is that the agent receives the belief through a send transfer-definition in a communication activity from another agent. In this case, the modeler cannot define the access counter constraint a priori and there needs to be another way to declare the access counter. One possible solution is to provide a belief method to set a belief's access counter after the belief has been created. Using this method, the agent can change the belief access counter property at any time. The modeler can specify when this method is called, in the body of a workframe, thoughtframe or detectable. Another possibility is to provide a default access counter as a property on an attribute. The next section discusses the second type of LTM, called context-based long-term memory.

[1] In the current language this is the *conclude* statement.

Context-based Long-Term Memory

The context-based LTM (CLTM) is shown at the right bottom in Figure 1. It contains the complete context of every WFI, TFI and ACI that has been created and performed by the Executive. In the current Brahms virtual machine (VM), a WFI, TFI or ACI exists only during its execution. After the execution has completed it gets deleted from STM, and thus the agent has no means to access past workframe, thoughtframe or activity executions. With the CLTM, the instance is stored so that the agent can access this past instance as a historical context in which activities were performed. There are two issues that determine the effectiveness of the CLTM; first, what is saved as the context of a WFI, TFI, and ACI and stored in CLTM? Second, how the CLTM is accessed by the agent and made use of in its reasoning process. We first discuss how and what context is stored.

Context storing

Figure 3 shows what should be stored as part of a WFI, TFI, or ACI. It is obvious that each instance's context should include the variable bindings from its precondition matching. This is what in the current engine is called "the context of an instance".

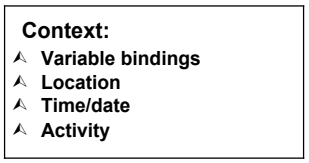

Figure 3. Context stored as part of a WFI, TFI or ACI

However, the newly designed context should include all activity parameters, and also all properties that are associated with the workframe and activity instances, such as priority, repeat, defined and actual duration (including start, interruptions, impasses and resumes). What should also be stored within the context is the location of the agent at the start, during and at the end of the activity. This might be available from bound variables within the instance (such as a location attribute value bound through a precondition), but this does not have to be the case. The agent can execute workframes, thoughtframes or activities, without specifying location as a precondition. Therefore, the location of the agent needs to be stored separately.

Bellow, we specify the contexts that need to be stored in CLTM for WFI, TFI and ACI's (these lists might not be complete):

WFI context:

Name

Type

Repeat

Priority
Time (start, end)
Agent locations (at start, during, at end)
Variable bindings
Precondition instances
Detectable instances
Activity Instances
Consequences (beliefs/facts created)

TFI context:
Name
Repeat
Priority
Time (start)
Agent location (at start)
Variable bindings
Precondition instances
Consequences (beliefs/facts created)

ACI context:
Name
Type
Parameter values
Duration (max and min duration, randomness)
Time (start, interruptions, impasses, continues, end)
Priority
Activity-type properties
Parent WFI
Detectable instances
Workframe instances
Thoughtframe instances
Agent Locations (at start and end)

In the next section, we describe how elements from both TLTM and CLTM are retrieved by the agent and stored back or used in STM.

Retrieving elements from LTM

When elements are stored back into STM, they behave as if they were newly created elements in STM. Retrieving elements from either TLTM or CLTM needs to be done differently. In the next two sub-sections we describe them separately.

Retrieving TLTM elements

Retrieval of elements from TLTM is not done according to a built-in mechanism. When beliefs, WFRs, TFRs or ACTs are moved from STM to TLTM they will stay in TLTM until the agent does a specific TLTM element retrieval method-call. Agents, beliefs, WFRs, TFRs, and ACTs have TLTM store and retrieval methods that the agent can execute.

Using the agent TLTM element retrieval methods, the agent can specify a "retrieval condition" for an element. The retrieval condition is used to match elements in TLTM. The method will return a list of matched element objects that the agent can now iterate over. For any specific belief, WFR, TFR, or ACT object, the agent can execute a retrieve method. The elements returned in the method's return list are not automatically stored in STM. Instead, the agent can decide which of the returned elements need to be stored into STM. This is done by simply executing an assert method. This way, the agent (and thus the modeler) has complete control over the retrieval of elements from TLTM.

Retrieving CLTM elements

Retrieval of elements from CLTM is done via a two-step method. The retrieval of elements from CLTM is always context specific. This means that it is specific to the current WFI, TFI and ACI. When an agent starts a new WFI, TFI or ACI, the engine can retrieve all elements from CLTM that match its name, and associate these elements with the current instance. The agent can now use specific CLTM retrieval methods to 1) return an element from the list of CLTM elements associated with the instance, and 2) access specific properties (for example, bound precondition variables or belief elements that matched the preconditions) from the CLTM element returned.

Figure 4 shows an example of CLTM retrieval for the workframe "Bussiness Trip @ Ames".

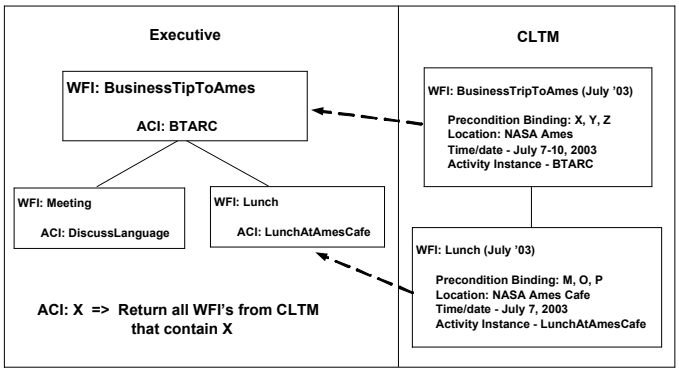

Figure 4. Retrieval of CLTM elements

In this example, we assume that the workframe is the current WFI in STM. The workframe contains a composite activity called BTARC. In BTARC there are two workframes called Meeting and Lunch. Workframe Meeting calls the DiscussLanguage activity, while workframe Lunch calls the LunchAtAmesCafe activity.

CLTM in Figure 4 shows two past WFI's that match both the current WFI and the sub-workframe in activity

BTARC, namely Lunch. These two CLTM elements represent WFI's that the agent performed at some time in the past. They represent previous business trips to Ames, and a lunch at the Ames Café during this previous business trip.

Besides, precondition matching, the engine retrieves these CLTM elements and associates these with the current WFI. When executing the current WFI, the agent can use the WFI's reflexive methods to retrieve these associated CLTM elements, and "memorize", i.e. assert the retrieved properties as beliefs in STM that are necessary.

Summary

In this paper, we discussed an architecture and associated methods for agent short-term and long-term memory. Over the past several years we have known that Brahms cannot deal appropriately with multiple-day simulations. Recently, in the Mobile Agents project, we also identified a significant limitation of an agent's memory on its reasoning efficiency. As the belief-set of an agent grows, the agent's precondition matching becomes exponentially slower, significantly impacting efficiency of the overall system. The new memory architecture discussed in this paper will resolve both limitations. In addition, the architecture will allow the Brahms modeler to design and implement complex memory retrieval methods, based on appropriate ad hoc models of memory. Instead of implementing a pre-specified long-term memory retrieval method, we have chosen for an unbiased and flexible retrieval capability, allowing the Brahms modeler to specify how agents deal with retrieval of "memorized" elements from long-term memory.

Acknowledgments

We like to thank our long-term collaborator and friend, Bill Clancey, for his comments. Without him Brahms would not exist. We also like to thank Jeff Bradshaw and the KAoS development group for their collaboration and support. Funding for this work is provided in part by the NASA/Ames Intelligent Systems Program, Human-Centered Computing area, managed by Mike Shafto.

References

Anderson, J. R., and Lebiere, C. (1998). *The atomic components of thought*, Lawrence Erlbaum Associates, Mahwah, NJ.

Beardsley, T. (1997). "The machinery of thought." Scientific American, Vol. 277, 78-83.

Clancey, W., J. (1997). *Situated Cognition: On Human Knowledge and Computer Representations*, Cambridge University Press.

Clancey, W. J. (1999). *Conceptual Coordination: How the mind Orders Experiences in Time*, Lawrence Erlbaum Associates, Mahwah, New Jersey.

Clancey, W. J. (2002). "Simulating Activities: Relating Motives, Deliberation, and Attentive Coordination." *Cognitive Systems Research*, 3(3), 471-499.

Clancey, W. J., Sierhuis, M., Alena, R., Crowford, S., Dowding, J., Graham, J., Kaskiris, C., Tyree, K. S., and Hoof, R. v. "The Mobile Agents Integrated Field Test: Mars Dessert Research Station 2003." *FLAIRS 2004*, Miami Beach, Florida.

Clancey, W. J., Sierhuis, M., Kaskiris, C., and Hoof, R. v. "Advantages of Brahms for Specifying and Implementing a Multiagent Human-Robotic Exploration System." *The 16th International FLAIRS Conference 2003*, St. Augustine, Fl.

Lewis, R. L. (1999). "Cognitive modeling, symbolic." The MIT Encycolpedia of the Cognitive Sciences, R. Wilson and F. Keil, eds., MIT Press, Cambridge, MA.

Lewis, R. L. (2001). "Cognitive theory, Soar." International Encylopedia of the Social and Behavioral Sciences, Pergamon (Elsevier Science), Amsterdam.

Miyake, A., and Shah, P. (1999). "Models of Working Memory: Mechanisms of Active Maintenance and Executive Control." Cambridge University Press, Cambridge, UK.

Newell, A. (1990). *Unified theories of cognition*, Harvard University Press, Cambridge, MA.

Sierhuis, M., and Clancey, W. J. (2002). "Modeling and Simulating Work Practice: A human-centered method for work systems design." *IEEE Intelligent Systems*, Volume 17(5)(Special Issue on Human-Centered Computing).

Sierhuis, M., Clancey, W. J., Seah, C., Trimble, J. P., and Sims, M. H. (2003). "Modeling and Simulation for Mission Operations Work System Design." *Journal of Management Information Systems*, Vol. 19(No. 4), 85-129.

Wickelgren, I. (1997). "Getting a grasp of working memory." Science, Vol. March (275), 1580-1882.