Large Declarative Memories in ACT-R

Scott Douglass (Scott.Douglass@mesa.afmc.af.mil)
Jerry Ball (Jerry.Ball@mesa.afmc.af.mil)
Human Effectiveness Directorate, 711th Human Performance Wing
Air Force Research Laboratory
6030 S. Kent Street, Mesa, AZ 85212

Stuart Rodgers (stu@agstechnet.com)
AGS TechNet
Dayton, OH

Abstract

The development of large-scale cognitive models introduces significant computational challenges. Large declarative memories are a case in point. It is not computationally feasible to load a large declarative memory into the process space available for execution of a cognitive model. Fortunately, computer science provides us with relational databases to support access to large external stores of information from within an executing process. This paper motivates and describes the interfacing of the ACT-R cognitive architecture with a relational database to support large declarative memories within ACT-R models.

Keywords: large-scale cognitive modeling; large declarative memory, ACT-R cognitive architecture, relational database.

The Need for Large Declarative Memories

The typical cognitive model models a specific laboratory task with modest declarative memory (DM) requirements. The DM of such models can be loaded into the process space of the model and executed efficiently. The ACT-R cognitive architecture (Anderson, 2007; Anderson et al., 2004) comes with efficient data storage and access mechanisms for managing modest size declarative memories within the process space of the model. However, the development of cognitive models of complex tasks requires more substantial DMs. At some point, the size of declarative memory becomes too large to be loaded into the executing process as a whole and external data storage and access mechanisms are needed.

Researchers in the Air Force Research Laboratory, Human Effectiveness Directorate, Cognitive Models and Agents Branch (AFRL/RHAC) in collaboration with the Cognitive Engineering Research Institute (CERI), AGS TechNet and L3 Communications are engaged in a project to develop a Synthetic Teammate (Ball et al., 2009) capable of functioning as the Air Vehicle Operator (AVO), or pilot, in a 3-person team task simulation of an Uninhabited Aerial Vehicle (UAV) performing reconnaissance missions (Cooke & Shope, 2005). All the major components of the Synthetic AVO are being developed within the ACT-R cognitive architecture, including language comprehension (Ball, Heiberg & Silber, 2007), language generation, dialog and situation models, and task behavior (Myers, to appear). The use of ACT-R reflects our commitment to develop a cognitively plausible, yet functional synthetic agent. We believe that adhering to well-established cognitive constraints may actually facilitate the development of functional agents by pushing development in cognitively plausible directions which are more likely to be successful in modeling complex human behavior than non-cognitively plausible alternatives.

The Synthetic AVO must communicate with the planning officer, who plans the route, and the payload operator, who takes pictures of targets, in order to accomplish a 40 minute reconnaissance mission involving more than 12 waypoints, many of which are targets. As a result, language comprehension is an important component of the larger Synthetic AVO model. Further, the range of vocabulary and grammatical constructions used by the teammates to communicate with the AVO is extensive and unpredictable. An analysis of spoken transcripts between all human teams participating in several earlier studies identified a total of 2500 unique words in 19K spoken utterances and an analysis of text chat transcripts in a recent study identified 1700 unique words in 5500 text chat messages. Overall, we expect the model to require a vocabulary of 10-15K words and multi-word units to be capable of adequately modeling human communicative behavior on this complex task. By itself, the language comprehension component is pushing the scale of DM beyond the capacity of the existing ACT-R data storage and access mechanisms.

To support the projected vocabulary, we are pursuing the integration of a large subset of the WordNet lexicon (Fellbaum, 1998; Miller, 1998) into the model.

Table 1: Summary of the SGP parameters introduced by the persistent-DM module.

Parameter Name	Description of Parameter Behavior
PDM-active	Enable/disable the use of persistent DM
PDM-add-DM-serializes	Determines if add-dm produces DB chunks or not
PDM-resets-clear-DB	Enable/disable the clearing of the persistent DM during model resets
PDM-DB-name	Name of PostgreSQL database containing the persistent DM of interest
PDM-user	Username required by the PostgreSQL DBMS for DB access
PDM-passwd	Password required by the PostgreSQL DBMS
PDM-DBMS-hostname	DBMS hostname provided as a machine name or IP address

To accomplish this, we are leveraging the WN-Lexical interface to ACT-R developed by Bruno Emond (2005). The WN-Lexical interface provides a capability to load all of WordNet into ACT-R's DM at once. However, on our hardware the model exhausts the memory capacity after just 30% of WordNet is loaded.

Although we do not envision using the entire WordNet lexicon in our language comprehension model, we do expect to use a large enough subset for it to be problematic for the existing ACT-R data storage and access capabilities. To support the integration of a large subset of the WordNet lexicon into the language comprehension model, an external data storage system is needed. WN-Lexical provides a capability to load individual words into DM as needed, retaining unused words in an external store. However, this capability is not tightly integrated with ACT-R's declarative memory module and cannot take advantage of ACT-R DM mechanisms like spreading activation. This is problematic since there is a high level of lexical ambiguity in the WordNet lexicon (e.g. the word "dog" has eight senses in WordNet) and spreading activation is a key mechanism for dealing with such ambiguity. Ideally, **the external data storage capability should be transparent from the perspective of ACT-R and DM**—i.e. whether the model is accessing a word from an internal or external data store should not affect the behavior of the model. The next section describes just such a capability.

Persistent DM for ACT-R

Current Declarative Module

The chunks constituting declarative memory in ACT-R 6 are stored internally in a single data structure. When a retrieval request is executed by the ACT-R declarative module, a process carried out by the module *essentially* uses constraints in the retrieval request and computed activations to identify which chunk matching the constraints (if any) should be accessed from the data structure and placed into the retrieval buffer. This process is simple and effective when the number of chunks in the data structure remains below a certain threshold. As the number of chunks in declarative memory increases, the process slows and eventually breaks down.

An ACT-R user wanting to model cognitive processes dependent on declarative memories larger than a critical threshold therefore requires new data storage and access mechanisms to support DM chunk storage and retrieval. Fortunately, the modular nature of ACT-R and the software design of ACT-R 6 greatly facilitate the development and deployment of alternative chunk storage and retrieval mechanisms.

New SQL Functionality

To meet large DM requirements, we've developed a chunk storage and retrieval capability in ACT-R 6 based on PostgreSQL, a powerful, open source object-relational database management system (DBMS). This "*persistent-DM*" module outsources chunk storage to an industrial-strength external DBMS while leaving ACT-R's retrieval calculus untouched. The persistent-DM module (defined in a single file) modifies the behavior of ACT-R's declarative module by: (1) introducing seven control parameters; (2) providing programmatic support for managing the interaction between ACT-R and the PostgreSQL DBMS; (3) extending the retrieval process; and (4) modifying the comparison of chunk slots. Table 1 describes the seven control parameters.

The persistent-DM module's parameters allow the ACT-R modeler to easily control the behavior of the module. For example, toggling the *PDM-active* parameter from T (on) to NIL (off) disables use of PostgreSQL and returns the chunk storage/retrieval behavior of ACT-R back to its default. Setting *PDM-add-DM-serializes* and *PDM-resets-clear-DB* to T (yes) when persistent DM is enabled allows a modeler to populate the persistent DM with chunks explicitly defined in a model. Lastly, setting *PDM-add-DM-serializes* and *PDM-resets-clear-DB* to NIL (no) when persistent DM is enabled allows a modeler to make a DM that persists across model runs available, without having to comment out parts of the model. The persistent-DM module provides the ACT-R modeler with programmatic support for the definition and management of external PostgreSQL databases. A modeler using the persistent-DM module can programmatically:

- Generate and use generic SQL queries to interact with persistent external knowledge bases.
- Serialize (write) and de-serialize (read) ACT-R chunks in massive knowledge bases.
- Use transactions, rollbacks and commits to protect, undo, and save changes to declarative memory.

Most importantly, a modeler using the persistent-DM module can transparently:

- Employ PostgreSQL DBMS-based alternatives to the default chunk addition, removal, and merging processes in ACT-R 6. These alternatives don't change the calculus underlying ACT-R's declarative module, they just change the way retrieval requests are used to determine the subset of chunks from declarative memory that will participate in the calculation of activation during the retrieval process.
- Use retrieval constraints based on regular expressions.

To use the persistent-DM module, an ACT-R modeler needs to: (1) install the PostgreSQL DBMS on a computer (the computer can be the modeler's workstation or a dedicated remote server); (2) install a common-lisp library supporting interaction with PostgreSQL; (3) drop the persistent-DM module definition file into the ACT-R 6 "modules" directory; (4) activate the module by adding something like the following to the SGP section of a model.

```
(sgp :pdm-db-name "model-v5-DM"
     :pdm-user "Scott"
     :pdm-passwd "Open_Seseme"
     :pdm-resets-clear-db T
     :pdm-add-dm-serializes T
     :pdm-active T
...
```

Figure 1: Activating and configuring the persistent-DM module in an ACT-R model.

The activation of the persistent-DM module has no impact on model behavior. However, wall clock performance of the ACT-R simulator is impacted. Chunk serialization and de-serialization processes depend on non-trivial information exchanged with the PostgreSQL DBMS and using persistent-DM when models have small declarative memories exacts a fixed and relatively high cost. If the cost of using persistent-DM remains essentially fixed, then persistent-DM will eventually outperform ACT-R's default declarative memory system when models have large enough declarative memories. To find out where this tipping point is, and to better understand when we should and shouldn't use the persistent-DM module, we conducted a comparative analysis of default and persistent-DM.

Computational Efficiency of Retrievals from Different Size Declarative Memories

When the number of chunks maintained by an ACT-R model remains low, keeping them on-hand in an internal data structure facilitates optimal simulator performance. Under these circumstances, the cost of forming an external query, dispatching the query to a DBMS, and interpreting the return from a DBMS exceeds the cost of comparing candidate chunks to retrieval constraints. When the number of chunks maintained by an ACT-R model exceeds a certain value, keeping them on-hand in an internal data structure exceeds lisp/machine memory limits and the framework crashes. Under these circumstances, modeling can only proceed if a DBMS is used. Between these two extremes is a decision space in which the benefits of using persistent-DM gradually exceed the costs. To start exploring the nature of this decision space, a controlled evaluation of the performance of the persistent-DM was conducted. During this evaluation, three factors were systematically varied:

- *Type of DM*: default or persistent
- *Size of DM*: ~1K, ~5K, ~10K, ~20K, ~80K or ~240K chunks
- *Retrieval Constraints*: 1, 2, 3 or 4 slot/value requirements

Each of the differently sized DMs was defined by a separate ASCII file containing a single call to ACT-R's "add-dm" command. Under conditions where the type of DM being evaluated was default, these files were loaded into ACT-R and "add-dm" added chunks to the internal chunk table. Under conditions where the type of DM being evaluated was persistent, PostgreSQL databases containing these same chunks were connected to by the persistent-DM module.

```
aardwolf-noun-pos
  ISA noun
   parent  "none"
   token  "type"
   type  noun
   super-type  noun
   subtype  noun
   form  nil
   word  aardwolf-word
   gram-form  common-sing
   animate  animate
```

Figure 2: ACT-R chunk specification of a noun describing an aardwolf part-of-speech.

Chunks used in the evaluation represented nouns. The ACT-R specification of the noun chunk type consisted of nine slots (see Figure 2).

Regardless which type of DM was being evaluated, retrieval requests intended to recover 10 randomly chosen chunks from each differently sized DM were executed to assess wall-clock retrieval times. Figure 3 shows example retrieval requests based on 1 and 4 retrieval constraints.

```
+retrieval>
  ISA noun
   parent  "none“
...
+retrieval>
  ISA noun
   parent  "none"
   super-type  noun
   word  aardwolf-word
   gram-form  common-sing
```

Figure 3: Example ACT-R retrieval requests based on 1 and 4 retrieval constraints.

Additional retrieval constraints lead to more specific retrievals but require additional slot/value comparisons. Evaluations of ACT-R's retrieval process lead us to believe that the use of additional slot/value constraints in more constrained retrieval requests would impose a time cost when ACT-R's default retrieval mechanisms are employed. We incorporated the retrieval constraints factor into the evaluation in order to systematically assess the actual costs (if any) of employing greater retrieval constraints. Due to database indexing and SQL query optimizations; additional constraints shouldn't impose similar time costs. The incorporation of the retrieval constraints factor into the evaluation allowed us to directly assess the efficiency (or lack of) of SQL queries based on composed constraints.

During the evaluation, 2 performance measures were recorded:

1. *Setup-time*: The amount of time it took to make chunks in the differently sized DMs available to ACT-R's retrieval process.
2. *Retrieval-time*: The amount of time it took to actually retrieve a chunk matching the retrieval constraints.

Table 2 lists the average setup times we measured in the evaluation of default and persistent DM. Times in the table clearly show that loading a declarative memory specification into ACT-R through default methods requires an increasing amounts of time when declarative memory size increases. The details of the relationship between the size of DM and set-up time, while interesting, do not contribute to the point that as declarative memory grows, a load-time problem appears. Consequently, they won't be discussed further. The failure of ACT-R to load 240,000 chunks into default declarative memory provides us with an initial estimate of the number of chunks—at least as complex as our noun chunk type—beyond which ACT-R becomes unstable on our hardware. Lastly, the cost of connecting to an external PostgreSQL DBMS was found to be relatively constant.

Table 2: Summary of setup times (in msec)

	~1K	**~5K**	**~10K**	**~20K**	**~80K**	**~240K**
default	98	375	981	2828	103395	NA
persistent	82	90	90	94	86	86

Ten retrieval times were recorded under all 6x4 combinations of DM size and retrieval constraints. These measures were analyzed using a repeated measures ANOVA. Since default DM was unable to accommodate 240,000 chunks, retrieval times are missing in Figure 4. The persistent DM retrieval times under these circumstances were not included in the repeated measures ANOVA. All main effects and interactions were found to be highly significant. The significant Size of DM X Retrieval Constraints X Type of DM interaction (F(12,108) = 3.682, $p<0.001$) is illustrated in Figure 4. The figure shows that while wall-clock retrieval times are uninfluenced by the size of declarative memory when persistent DM is used, they are significantly influenced by the size of declarative memory when default DM is used. When declarative memory contains more than approximately 80,000 chunks, the benefits of keeping chunks in an internal data structure are lost. 80,000 chunks seem to be the point at which things decidedly favor persistent-DM; at least given the complexity of our noun chunk type. Using additional retrieval constraints imposes no additional time costs on persistent DM. A clear, and eventually significant, relationship between constraints and retrieval time can be seen in default DM.

While this simple evaluation reveals some of the capabilities of persistent-DM, much work needs to be done. For example, activations were not computed in the comparative study. When sub-symbolic chunk properties are calculated and maintained in ACT-R's declarative module, symbolic properties of candidate chunks such as fan and type inheritance can easily lead to a dramatic need to obtain properties of non-candidate chunks. In order to begin to understand the costs and benefits of persistent-DM, the calculation of activation was inhibited. We are planning follow-up evaluations that will systematically control fan and chunk type inheritance in order to further explore the capabilities of the persistent-DM module.

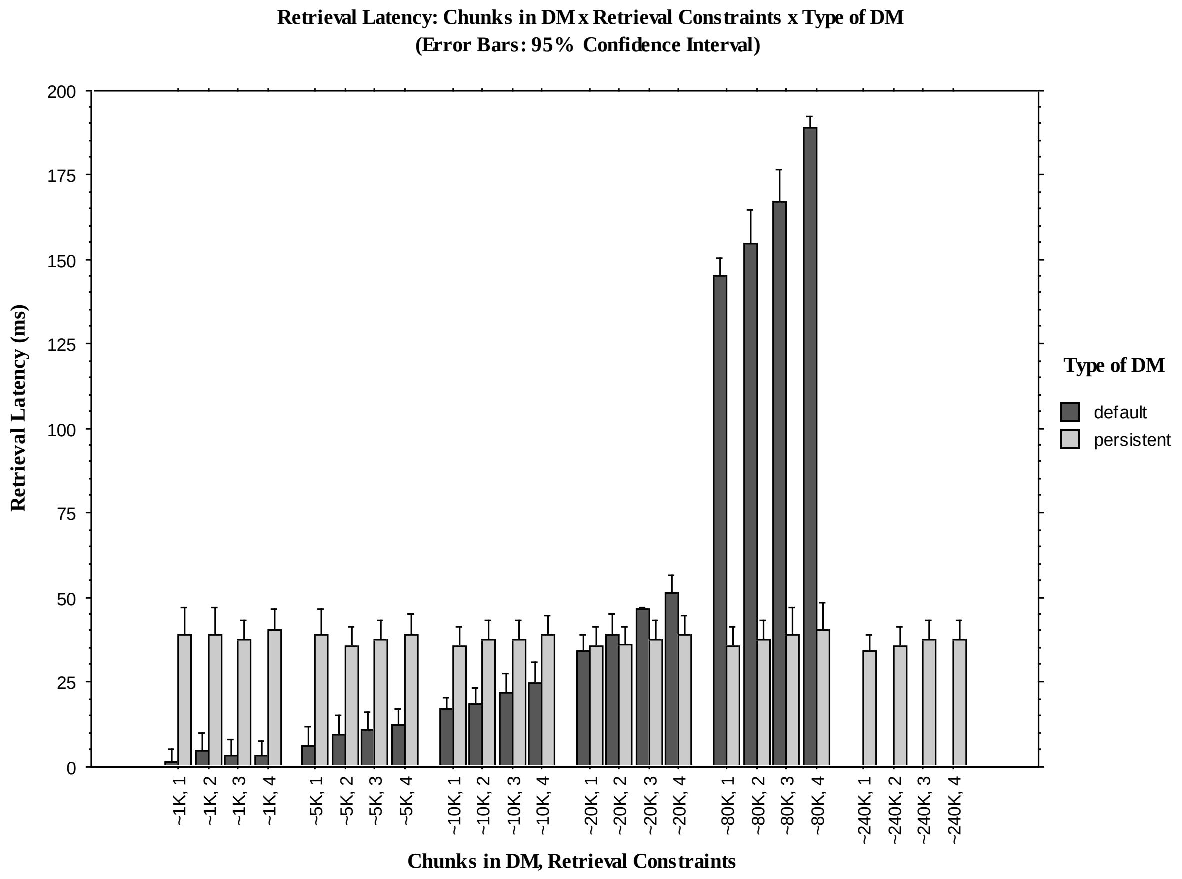

Figure 4: Summary of wall-clock retrieval times.

Conclusions

We have integrated a relational database with the ACT-R cognitive architecture to support the creation of large, externalized, persistent declarative memories whose behavior matches that of existing internal declarative memories. We are using this capability in the development of a complex model of a Synthetic AVO capable of communicating with human teammates in performance of a reconnaissance task. More generally, such a capability is needed in the development of complex cognitive models with significant declarative memory requirements and this capability aligns with our research focus on large-scale cognitive modeling (cf. Douglass & Luginbuhl, 2008).

Future Directions

The interface to the external database is currently functional and outperforms ACT-R's internal data storage and access mechanisms on large declarative memories under a range of conditions as demonstrated in the previous section. However, we believe there is room for significant improvement and optimization of the performance of the interface. In particular, the retrieval of a word from DM actually involves a chain of retrievals which includes all the elements in the fan list of the word. In the worst case, this chain could consume much of declarative memory, bringing the system to its knees. Besides needing to retrieve the fan list for a word in order to compute activations, if the retrieval template is highly unconstrained, many DM chunks will match and the computations may exceed process internal resource capacities. We bumped into this problem early on in a version of the model which only used spreading activation based soft constraints on word retrieval. If the retrieval template contains no hard constraints on the form of the word to be retrieved, relying exclusively on spreading activation to bias the retrieval, then ACT-R must compute the activation of every word in DM to determine which word to retrieve. This is computationally explosive and was unworkable with a mental lexicon of just 2500 words. We were forced to reinstate a whole word hard constraint on retrievals, with a fallback to a first letter hard constraint and spreading activation based soft-constraint on matching words if the whole word retrieval fails. Currently, these retrievals are executed by different productions. However, we are exploring the possibility of using the regular expression capability provided by the persistent DM module (a slot name preceded by "~"

invokes this regular expression matching capability) to conditionally retrieve the whole word first and if that fails then retrieve the word with highest activation matching the first letter, all within a single retrieval production. Given a conditional retrieval capability, a whole word match would terminate the retrieval, returning the matched word, and the less constrained and more computationally expensive first letter match with spreading activation over matching words would not occur.

We are also planning on using the regular expression matching capability to support retrieval of perceptual units at multiple levels of representation. The perceptual module of ACT-R currently divides the linguistic input into word units using a function called chop-string. This function relies on spaces and punctuation to delimit words. For example, the input "he went." would be divided into "he" "went" and ".". However, sometimes words contain punctuation and shouldn't be divided—for example "etc." and "didn't". And sometimes words can have a space as in "ad hoc" and "a priori". In the case of "didn't", the chop-string function returns "didn" "'" and "t" and it takes three attention fixations and several productions per "word" to process this input. Given the rapidity with which humans process language during reading—approximately 225 msec per space delimited word during silent reading (Rayner, 1998)—this treatment of "didn't" is unlikely to be cognitively plausible. To bring the language comprehension model into closer alignment with reading results, what is needed is a capability to recognize the largest unit in DM which matches the input, often matching multi-word units in a single attention fixation. To achieve this we are implementing a capability to do retrievals at multiple levels using a disjunction of perceptual units derived from the input. For example, "didn't" will lead to an attempt to retrieve either "didn't" or "didn" within a single retrieval specification, "John's" (as in "John's book") will lead to an attempt to retrieve either "John's" or "John", "etc." will lead to an attempt to retrieve either "etc." or "etc", "went." will lead to an attempt to retrieve either "went." or "went", "a priori" will lead to an attempt to retrieve either "a priori" or "a" and "because of" will lead to an attempt to retrieve either "because of" or "because".

Finally, longer term we are contemplating pushing ACT-R's activation computation into the database—transparently from the perspective of ACT-R and DM. This would avoid the need to retrieve large numbers of chunks from external DM in order to compute their activations within the ACT-R process.

References

Anderson, J. R. (2007). *How Can the Human Mind occur in the Physical Universe?* NY: Oxford.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S, Lebiere, C, and Qin, Y. (2004). An Integrated Theory of the Mind. *Psychological Review* 111, (4). 1036-1060

Ball, J., Heiberg, A. & Silber, R. (2007). Toward a Large-Scale Model of Language Comprehension in ACT-R 6. *Proceedings of the 8th International Conference on Cognitive Modeling*, pp. 163-168. Edited by R. Lewis, T. Polk & J. Laird.

Ball, J, Myers, C., Hieberg, A., Cooke, N., Matessa, M. & Freiman, M. (2009). The Synthetic Teammate Project. Paper presented at the *Behavior Representation in Modeling and Simulation Conference*.

Cooke, N. & Shope, S. (2005). Synthetic Task Environments for Teams: CERTT's UAV-STE. *Handbook on Human Factors and Ergonomics Methods*. 46-1-46-6. Boca Raton, FL: CLC Press, LLC

Douglass, S. & Luginbuhl, D. (co-chairs) (2008). Cognitive Modeling and Software Engineering Workshop.

Emond. B. (2006). WN-LEXICAL: An ACT-R module built from the WordNet lexical database. In *Proceedings of the Seventh International Conference on Cognitive Modeling* (pp. 359-360). Trieste, Italy.

Fellbaum, C. (ed) (1998). *WordNet: An electronic lexical database. Cambridge*. MA : MIT Press.

Miller, G.A. (1998). Foreword, In C. Fellbaum, (ed), WordNet: An electronic lexical database. Cambridge, MA: MIT Press

Myers, C. (to appear). An Account of Reuse and Integration. *Proceedings of the 9th International Conference on Cognitive Modeling*

Rayner, K. (1998). Eye Movements in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin*, 124(3), 372-422.