A Concurrent Atkinson-Shiffrin Memory Model

Elena Cokova*
Dept Computer Science,
Tufts University
elena.cokova@tufts.edu

Jen Hammelman*
Dept Computer Science,
Tufts University
jennifer.hammelman@tufts.edu

Naomi Zarrilli*
Dept Computer Science,
Tufts University
naomi.zarrilli@tufts.edu

1. INTRODUCTION

The goal of our project was to develop a concurrent model of the Atkinson-Shriffrin (multi-store) memory model [1]. With this model we hoped to demonstrate how computational implementation can act to realize the details of theoretical models (Fig 1). Though the model itself has since been rejected as our knowledge of the brain has increased, its' implementation still gives insight into the complexities of shared memory access in concurrent information-processing systems. The model can be accessed at https://github.com/ecokova/Memory-Model.

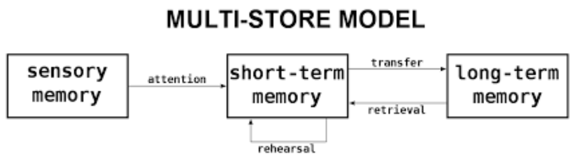

Figure 1: Atkinson-Shiffrin memory model diagram

2. ATKINSON-SHIFFRIN (MULTI-STORE) MODEL

In the Atkinson-Shiffrin Memory model (Fig 1), it is suggested that the brain is made up of three individual memory stores - sensory, short-term, and long-term memory. As a person interacts with the outside world, input is stored into sensory-memory, after which attention can bring some of these memories into short-term memory, followed by rehearsal to enter long-term memory. Sensory and short-term memory stores are 1) bounded in size and 2) subject to forgetting if an individual memory sits for too long within the store. Retrieval acts to take memories from long-term, triggered by similar memories in short-term, and moves them to short-term. This model appeared to be a natural framework from which to explore the access of shared memory stores in concurrent systems. To our knowledge, this is the first implementation of the Atkinson-Shiffrin Memory model.

*The authors contributed equally to this work and declare no conflicts of interest. The authors would like to thank Mike Shah and Mark Sheldon for their guidance in this work. This project was implemented as a course assignment for COMP50 Concurrent Programming Fall 2015. Copyright Naomi Zarrilli, Jen Hammelman, and Elena Cokova, 2015.

3. CLASSES

In our implementation, *memory*, *action*, and *retrieval* are three classes inheriting from the thread class in python (Fig 2). These three classes encompass the parts of Atkinson-Shiffrin Memory model. With the three memory stores as instances of the Memory class, rehearsal and attention as instances of the Action class and retrieval is an instance of the Retrieval class.

3.1 Memory Class

The memory class represents an individual memory bank. It has a queue which stores memories and can be accessed by other classes. The memory class inherits from the threading library and its run function simply removes the oldest memory in its queue every x seconds, which represents memory decay. The FlexQueue is used as the memory store for the implementation is a modified version of the Python thread-safe Queue class, so it is naturally protected against multiple thread accession problems (Fig 3).

3.2 Action Class

The action class represents a generic cognitive process that is responsible for moving memories from one store to the next. Memories each have an associated frequency to control when they may be moved to the next memory story. The action chooses a random memory to act on and checks its frequency. If the frequency is at the action's predetermined threshold, the memory is moved to the end of the next memory store. If not, the action increments the frequency.

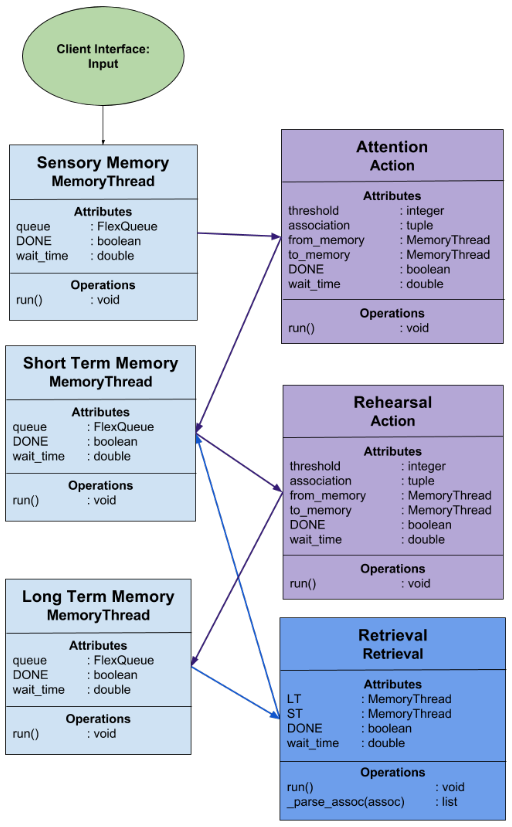

Figure 2: Class diagram for Memory-Model implementation in Python.

3.3 Retrieval Class

The retrieval class represents the process of "remembering" something from long-term that is similar to a current short-term memory. This involves first accessing short-term to select a random memory, finding a like-association in long-term, and then moving said association into long-term. To determine "likeness" we used the Wordnet package [2] for the command-line to look for similar words between short-term and long-term.

4. MULTI-STORE MEMORY FOR MBTI TEST

We decided to provide a use-case for our program using the MBTI questionaire as our system input and evaluation of personality on the back-end. The MBTI test is as described in Hammer et al., 1996 [3].

4.1 Information Overload

Our model demonstrates what happens when a brain is overloaded with information through its response to being given piped in input versus user input. When a user answers the MBTI questions, these answers go into sensory memory and as time passes and more questions are answered, some of these memories are paid attention to and rehearsed which allows memories to go to short term and long term memory. Also, there is time for memories to decay from these memory banks because the time it takes for a user to process and answer questions gives the model enough time to exercise retrieval and decay. When the model is given piped in input, there is very little time for these memories to decay or be paid attention to or rehearsed. The result will be that the sensory memory ends up storing most of the memories, and there are very few changes in the short-term and long-term memory banks. This simulates the process of information overload because when a human is overloaded with information, a lot of it ends up remaining in sensory memory and very little of it will actually be recalled later on because there was so little time to process the information given.

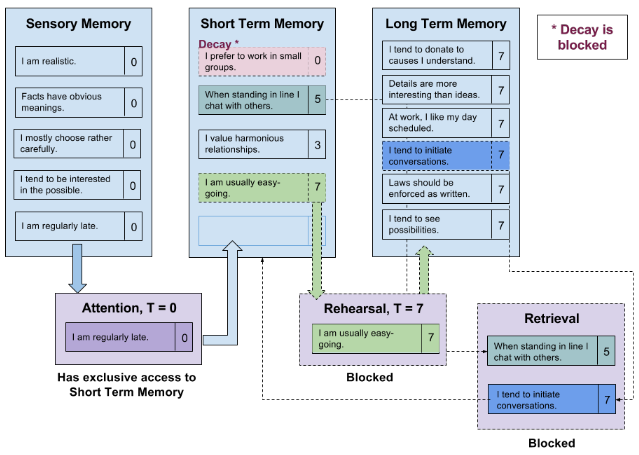
Sensory Memory
I am realistic. 0
Facts have obvious meanings. 0
I mostly choose rather carefully. 0
I tend to be interested in the possible. 0
I am regularly late. 0
Short Term Memory
Decay *
I prefer to work in small groups. 0
When standing in line I chat with others. 5
I value harmonious relationships. 3
I am usually easy-going. 7
Long Term Memory
I tend to donate to causes I understand. 7
Details are more interesting than ideas. 7
At work, I like my day scheduled. 7
I tend to initiate conversations. 7
Laws should be enforced as written. 7
I tend to see possibilities. 7
* Decay is blocked
Attention, T = 0
I am regularly late. 0
Has exclusive access to Short Term Memory
Rehearsal, T = 7
I am usually easy-going. 7
Blocked
Retrieval
When standing in line I chat with others. 5
I tend to initiate conversations. 7
Blocked

Figure 3: Example state of the program during runtime. Memory stores can be accessed by only one thread at a time.

4.2 Variability of Results

Although we hoped within the course of this project to analyze how the parameters of the different threads could cause different behavior, we found this behavior more difficult to characterize than was expected. In fact, several runs of the same system were rarely the same (Fig 4). This is likely due to the unpredictable behavior of the thread activity. As such, to full understand the dynamics of our multi-store memory model was outside the realm of possibility for this project.

5. DISCUSSION

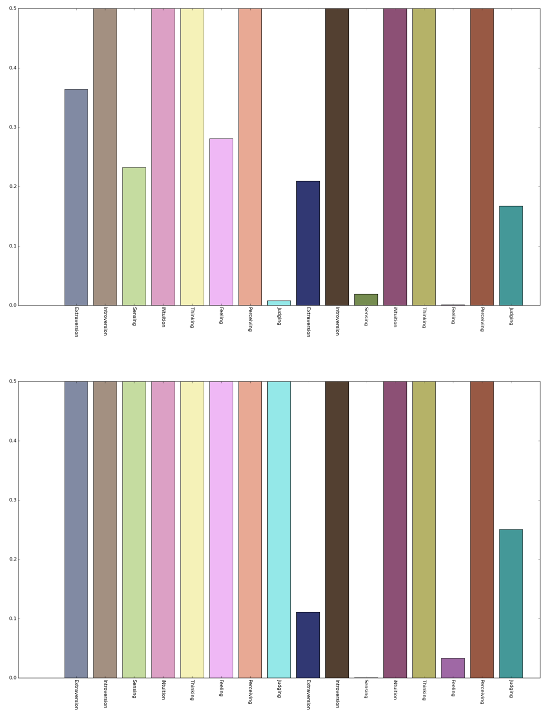

Figure 4: Results of two independent runs of the model with the same input have variable results.

5.1 Analysis of Outcome

We achieved a modified maximum deliverable. Our maximum deliverable was originally to model the Atkinson-Shiffrin Multi-Store model "along with the memory of the person, building a "personality" that generalizes their understanding (association) of the world" (from our Project Proposal document). We modified this goal by using the uClassify API to produce an MBTI report for the words that are stored in the memory banks. We had originally considered using Machine Learning to analyze memories and produce a report of the personality based on this analysis but using the API made our goal but easier to achieve.

5.2 Reflection on Design

Our best design decision was to modify the Python concurrent Queue with new functions as our "safe memory access" because it made the access to memory trivial and allowed us to modify these thread safe functions to test the impacts of bad concurrent practices (unsafe memory access) on the model. We could have used a different library for our visualization so the updating could be a live thread.

5.3 Reflection on Division of Labor

We divided the project so that each project member worked on one class. Elena worked on the Action class, Jennifer on Retrieval, and Naomi on Memory. We worked together to write the main file that made all of these classes work together. We spent a greater portion of time working together to integrate our pieces into a whole than we did on our individual class implementations. This was expected, seeing that most bugs and issues came up when these pieces were trying to work cohesively. Ultimately, working together on the main file allowed us to learn more about the pieces of code that we didn't write while contributing knowledge of our individual work, which made the division of labor subsequent integration effective.

5.4 Bug Report

Our most interesting bug by far was a problem in our code that was causing our action and rehearsal threads to run but not ever do their assigned functions. This came as a result of a try-except statement that was catching all errors, including an error that came from our failure to import a certain library for a function call. This took us maybe an hour and a half to discover, because everything in our code was logically correct. We found the source function of the problem and after some thought realized we had never added the library to the top of the file. We could have found it faster had we specified an error for the try except statement. As a bonus, this gave us a good familiarity with each other's code.

5.5 Incremental Improvements

We believe this model could be improved to include:

- A graph that changes as you input answers to the personality test, as opposed to rendering at the end of the test
- A quantitative characterization of the behavior of our code as a study of concurrent memory with and without safe memory accession (mutexes)

6. CONCLUSION

Taken together, the results of our project suggest that even a highly simplified model of the brain such as the Atkinson-Shiffrin model can be incredibly complex in terms of debugging errors and predicting results.

7. REFERENCES

[1] R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes. *The psychology of learning and motivation*, 2:89–195, 1968.

[2] C. Fellbaum. *WordNet*. Blackwell Publishing Ltd, 2012.

[3] A. L. Hammer and N. J. Barger. *MBTI applications: A decade of research on the Myers-Briggs Type Indicator*. Consulting Psychologists Press, 1996.