COMPUTATIONAL MODELING OF UPDATING OPERATIONS IN THE PERFORMANCE OF BASIC VERBAL WORKING-MEMORY TASKS

by

Adam Krawitz

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Psychology)
in the University of Michigan
2007

ACKNOWLEDGMENTS

I have so many people to thank! I start with my parents who have provided unwavering support throughout my life. Next there are my longtime friends Brian and David with whom I discovered that life can be an intellectual adventure.

Teachers at all points in my life have mentored and inspired me: Tom Sisson at West Junior High, Hank Landry at Hickman High, and Tom Doeppner at Brown. I would never have followed the path from computer science to psychology if not for the writings of Douglas Hofstadter and Daniel Dennett.

Here in Ann Arbor my life has been greatly enriched by those in the Ultimate community with whom I have played, trained and celebrated over the past six years on Fliegenplatte, Viscous Coupling, Nemo, Party Fowl, Pursuit, and Truck.

In the Brain, Cognition, and Action Lab, I have been trained in experimental psychology and in life. I thank Shane who first pointed me towards working-memory updating. My data collection was helped by the hard work of research assistants: Sveta Yegorova, Stephanie Miller, Rachel Asquith, and Michaela Ferguson. Without Steve as my officemate I would never have survived those long hours with R.

I greatly appreciate my committee, Dave Kieras, Rick Lewis, and Thad Polk, for their guidance, support, and advice. To my chair, Dave Meyer, I am humbled and inspired by your dedication, generosity, and insight.

And finally, I thank Janine, without whom I might never have gotten done. We are just beginning a long journey together through life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

AD	Articulatory duration
ANOVA	Analysis of variance
AT	Articulation time
CBR	Cumulative backward rehearsal
CFR	Cumulative forward rehearsal
EPIC	Executive-process interactive-control
ISR	Immediate serial recall
nRMSE	Normalized root mean square error
PGF	Position gradient function
RBR	Rolling backward rehearsal
RFR	Rolling forward rehearsal
RMSE	Root mean square error
RT	Reaction time
RTF	Response-type function
SPF	Serial position function
VWM	Verbal working memory
WM	Working memory
WSEF	Word set error function

ABSTRACT

Verbal working memory is a mental system for the temporary maintenance and manipulation of information supporting ongoing performance. The ability to update knowledge representations in working memory is critical to this system's support for flexible and goal-driven behavior. While maintenance has been extensively studied, theorized and modeled, the understanding of working-memory updating has not progressed apace. At an abstract level, updating has been conceived as a control process handled by a central executive acting on information maintained in an articulatory loop (Baddeley, 1986). I revise and elaborate this theory by developing a set of explicit computational models that account for performance in four verbal working-memory updating tasks. Performing these tasks requires the operations of adding and deleting words from the beginning and end of serially ordered verbal sequences. Across four experiments, I compare updating operations and develop a series of computational models that account for important aspects of performance. This work reveals that updating is not implemented by unique processes of executive control, but rather by the same perceptual, motoric, and control processes that mediate maintenance and rehearsal. I also found that deletion of words is performed by passively omitting them from rehearsal, not by an active process of inhibition; and that when multiple updating operations are required they are performed each in turn through cycles of articulation. Through careful experimentation and detailed computational modeling, I have decomposed the abstract executive process of updating into an explicit theory of performance.

CHAPTER 1

Introduction

In this dissertation I investigate how humans update ordered sequences of words held in verbal working memory. A person can maintain, manipulate, and act based on a sequence of words that they have never before encountered. This can be done in the first seconds after presentation of a sequence without significant intervening time for practice. The words themselves are typically familiar, but their particular order may be unique. This dynamic use of memory is fundamental for our ability to act flexibly and spontaneously based not only on the immediate environment but also on our evolving goals.

For example, consider an Ultimate player during a game. The captain calls out a sequence of players who will take turns making moves to get the disc and progress down the field to score a point. Depending on the teammates who are on the field and how the game is going, this particular sequence may be unique. As play is starting, the captain sees how the defense is lining up and adds a player to the start of the sequence. As play progresses, our player must act based on the called sequence while monitoring for exceptional circumstances that might require on-the-fly revision. The team scores, the point ends, and immediately a new sequence is called out for the next point, with many of the same players involved but now in a new order. Success in this environment depends crucially on the maintenance and manipulation of verbal working memory. Similar scenarios play out in any team sport involving variations on set routines of action.

As another everyday example, consider the scenario of being lost in an unfamiliar town. You roll down the window and ask a stranger how to get to your destination. The local reels off a sequence of street names and turns: "Right on Main, and then left on Broadway…" At some point he makes a mistake: "…and then left on Fourth – no that's right on Fourth, and then left on Division." At the end of the exchange you need to have the correct sequence in mind, ready to guide your travel as you start back on your way. Again, the ability to store and update a unique verbal sequence is necessary to guide behavior.

As these brief examples illustrate, dynamic behavior often depends on acquiring and modifying sequences of words and then behaving based on them. The Ultimate player who gets the sequence wrong will not make the play and his team will fail to score. The driver who turns left instead of right will end up more lost than before. While we can practice the underlying skills of sports, driving, and other performance oriented activities, it is often our ability to dynamically compose actions based on new and changing information that leads to the sophisticated behaviors that are uniquely human.

1.1 Goals of Research

In this dissertation, I take an approach to studying verbal working-memory updating that has both empirical and theoretical components. Empirically, my goal is to establish a basic set of facts about memory updating performance. What is the time course of updating? What is the effect of memory load on updating? What sorts of updates are harder or easier? What sorts of errors do subjects make? The answers to these and other questions are pursued through the use of new experimental tasks performed by normal college-aged participants.

Theoretically, the goal is to develop an explicit characterization of verbal working-memory updating through a series of computational models. Given the basic facts about performance, how can I account for it? What strategies do people use? How do they recover from partial loss of information? To what extent do they use the same or different methods to perform different types of updates? I develop initial models based on basic updating operations, and then test, refine, and generalize them through further experimentation and modeling. Much of the past work on control processes for verbal working memory has been vague and purely conceptual. I demonstrate that, through careful experimentation and detailed computational modeling, an abstract executive process can be decomposed into an explicit theory of performance.

1.2 Overview of Dissertation

This dissertation is structured to achieve the goals just outlined by iterative cycles of experimental data collection and computational modeling. Subsequent chapters are briefly outlined here to provide a roadmap. Please refer to the table of contents for a complete listing of chapters and sections with page numbers.

Chapter 2. Theoretical Background and Framework

Here I lay out a theoretical context for this work and explain why it fills a critical gap in the literature. I also introduce the modified Executive-Process Interactive-Control architecture used for developing models.

Chapter 3. Methodological Review and New Rehearsal Tasks

In this chapter, existing verbal working-memory tasks are evaluated for their adequacy for studying updating. This is followed by a discussion of the new rehearsal tasks and how they address important methodological issues with previous tasks.

Chapter 4. Experiment 1: Appending Words to Sequences

I start the investigation of working-memory updating with the operation of appending a word to a sequence. I use immediate serial recall (ISR), and a new cumulative forward rehearsal (CFR) task. I compare ISR and CFR performance to establish that the new task is tapping into the same memory system as the canonical verbal working-memory task. A series of models is developed to provide an initial perceptual-motor account of basic updating performance, with predictions for other updating operations.

Chapter 5. Experiment 2: Removing Words from the Beginning of Sequences

This chapter expands the analysis to situations where words are being dropped from the beginning of a sequence as well as being appended to its end. The findings with CFR are replicated and predictions for a rolling forward rehearsal task (RFR) are tested. New insights are gained into (a) the non-selective processes involved when two updating operations are required simultaneously and (b) the passive strategy taken to implement deletion.

Chapter 6. Experiments 3 and 4: Prepending Words to Sequences and Removing Words from the End of Sequences

Two more experiments are reported that replicate the initial findings and that explore two additional updating operations: prepending words to sequences and dropping words from the end of sequences. The first experiment compares all four updating tasks, while the second is a further investigation of cumulative backward rehearsal (CBR) and rolling backward rehearsal (RBR). Modeling of these additional new tasks supports and expands upon the lessons learned so far, and suggests a new heuristic for generalizing to more difficult updating operations.

Chapter 7. General Discussion

In this final chapter I appraise the empirical and theoretical findings that I have reached. Strengths and weaknesses of the dissertation are addressed and further work is suggested.

CHAPTER 2

Theoretical Background and Framework

In this chapter I will frame the present work by presenting the theoretical context to which it relates. General theories have been developed to explain working memory. However, these broad approaches have left much unspecified. To address this, a wide range of models have been developed previously, but they have focused on a narrow range of tasks and limited types of performance – primarily focusing on maintenance and not updating. An alternative, bottom-up, neurally inspired set of models has addressed updating more directly, but cannot handle higher level issues including, crucially, that of strategy. In between these two types of model lies a significant gap that this dissertation works to fill. I go on to discuss some of the important theoretical issues that I will address. And finally I describe the architecture used for subsequent modeling.

2.1 Background

2.1.1 Working Memory

Alan Baddeley's (1986) tripartite theory of working memory has been influential for the past twenty years. According to it, storage is handled by two modal components: a phonological loop for verbal material, and a visuo-spatial sketchpad for visual material. The phonological loop consists of a passive phonological storage buffer and an active articulatory rehearsal process to refresh the buffer's contents. The modal nature of these stores is significant because it supports a view of cognition as embodied (M. Wilson, 2001). In other words, perception and motor control play a large and central role in cognition, as opposed to being purely inputs and outputs to a fully abstracted and insulated cognitive system. This theme will figure prominently in the approach to working memory taken in the current modeling efforts.

The tripartite working-memory theory was built on older theories of short-term memory including the approach of Waugh and Norman (1965). While this earlier work already emphasized the roles of storage and rehearsal, a key to Baddeley's theory is the inclusion of a central executive as a principle component of the model (Baddeley, 1986). It emphasizes that the utility and importance of working memory is not just for maintenance of information, but for the rapid change in that information as dictated by both the external environment and the evolving goals and broader mental state of the individual.

In the last few years, another competing theory of working memory, the embedded-processes model of Nelson Cowan (1999), has also begun to gain currency. This theory differs in that the storage aspect of working memory consists of activated items in long-term memory instead of independent short-term buffers. It also has a second level of privilege for several (approximately four) items that are said to be in "the focus of attention" (Cowan, 2001).

However, the difference between these theories is not entirely clear for the storage of sequences of words as studied here. When a small set of words is used repeatedly, the emphasis is on the particular relationship between the words, and the concept of this information as activated long-term memory loses some of its force. It would seem unavoidable that new memories must be created for each episode to represent the unique relationship between the words. The distinction between new long-term memories in Cowan's theory and new short-term memories in Baddeley's theory will depend on the details of implementation. Since both theories are vague and qualitative, those details are left for others to flesh out. The theories also share a central executive to which broad powers of control are attributed; however, this too is left largely unspecified in the theories.

2.1.2 Central Executive

Baddeley himself acknowledged that the central executive has been neglected and needs further study (Baddeley, 1996). He and others (e.g., Hazy, Frank, & O'Reilly, 2006) refer to the need to "banish the homunculus" or "fractionate the central executive" by explicitly describing executive functions and how they work instead of placing them in a black box. A primary approach to satisfy this need has been to break down the monolithic concept of the executive into a collection of more specific components (Baddeley, 1996). Interestingly, in the process of doing this, the executive, which was originally conceived as a system for managing modal memory stores, has become a vaguer concept undifferentiated from executive control more generally. In other words, the central executive has not been studied as a system to manage working memory per se, but as a system to manage cognition (e.g., Miyake, Friedman, Emerson, Witzki, & Howerter, 2000). While this may be a reasonable approach, it has meant that research on the central executive has focused surprisingly little on working memory. By studying the updating of verbal working memory, I have made an explicit effort to focus on executive control *as it relates to working memory*.

It is useful to look at how executive control has been fractionated. In doing so, an important distinction must be made between the performance demands of tasks and the mental processes used in that performance. It is clear that there are experiments in which participants have to switch back and forth between two tasks, but it is an empirical question whether there is in fact a distinct “task-switching” mental process. This distinction is easily lost. I focus on *how* people update sequences of words held in memory, but I do not make an a priori commitment about what processes, executive or otherwise, are necessary to accomplish this performance.

Research *has* been done that pays attention to these distinctions. For example, Miyake et al. (2000) used individual differences and latent variable analysis to measure the relative contributions of different postulated executive processes to the performance of different tasks. They found that tasks with a large updating demand in the performance sense indeed showed a relatively large correlation with working-memory updating in the process sense. However, they made no attempt to understand how this updating process was implemented.

Miyake et al. (2000) considered three executive processes: shifting of mental set/task-switching, updating and monitoring of information, and the inhibition of prepotent responses. Yet the set of executive processes is far from settled. Shimamura (2000) lays out four: selecting/selective attention, maintaining, updating, and rerouting/task-switching. Smith and Jonides (1999) lay out five: attention and inhibition, task management, planning, monitoring and updating, and coding. All of these taxonomies include updating – emphasizing its importance and centrality to executive control. Still, the *performance* of updating may or may not involve more *processes* than just updating. For example, arguments have been made both for (e.g., Hartman, Dumas, & Nielsen, 2001) and against (e.g., Monsell, 1984) the role of inhibition in the performance of working-memory tasks that require deleting words.

2.1.3 Need for Models

The tripartite working-memory theory and the enumeration of executive processes provide a less than fully satisfying level of explanation. It is informal and underspecified. It is as if I asked you how to make those delicious cookies and you told me that you used flour and eggs and a lot of love. What I really want to know is the exact recipe. I want to know how much flour and eggs, and what order to mix them, and what temperature to turn the oven on at, and how long to cook them. This is not the most detailed level of description – I do not want to know the molecular composition of each ingredient, or the chemical changes that occur during baking – but it is a useful intermediate level.

The equivalent level here is the symbolic computational model. Intuitive claims by a theory get tested explicitly in such a model (Kieras & Meyer, 1997). Key details that can go unmentioned in an abstract theory must be dealt with here. For example, the tripartite working-memory theory provides no explanation about how serial order is maintained during the performance of immediate serial recall – the very task used to provide much of the initial support for the theory (Baddeley, 2002). Clearly, serial order is fundamental to the performance of this task! Since I am developing computational models, I will not be able to make such omissions.

There is a crucial shift here between theory and model. A theory attempts to generally explain the functions of a mental system that may be employed across a wide variety of tasks, while a model specifically attempts to account for performance in a single task. The applicability of a model is greatly enhanced if it is developed in the context of a computational cognitive architecture. Then the lessons learned from it can potentially generalize across tasks through shared architecture, control and strategy (Newell, 1973a, 1973b). Indeed, strategy is a key when studying executive control (Gilmartin, Newell, &

Simon, 1976; Meyer & Kieras, 1997). Essentially, situations that employ executive control are those where the environment alone does not dictate response, and thus there is an explosion of possible courses of action. A further motivation for developing computational models within architectures is that they can provide a path for new scientific knowledge about cognition to be used constructively in real-world applications, for example, human-computer interaction and interface design (Hornof, 2004; Kieras & Meyer, 1997). I develop computational models within a modified Executive-Process Interactive-Control architecture (Meyer & Kieras, 1997). In doing this, I pay explicit attention to the reuse of both architecture and strategy in generalizing from one working-memory updating operation to another.

2.1.4 Performance Models of Immediate Serial Recall

I now consider existing modeling work and its relevance to verbal working-memory updating. Computational modeling of short-term memory for words began at least 30 years ago (e.g., Gilmartin et al., 1976; Newell, 1973a) and has been going strong ever since. Many of these earliest models focused on immediate serial recall, and to a large extent, the focus has remained the same since then. In fact, while there are notable exceptions (e.g., Anderson, Bothell, Lebiere, & Matessa, 1998; Brown, Preece, & Hulme, 2000; Lewandowsky & Murdock, 1989), the vast majority of work on modeling verbal working memory has focused on immediate serial recall. The most recent models account for a wide range of effects and elaborate details of the data (e.g., Burgess & Hitch, 1999; Farrell & Lewandowsky, 2002; Henson, 1998; Page & Norris, 1998). But this leads to models that are very narrow in their scope and not easily or obviously generalized to situations where memory must be used more flexibly and dynamically. In terms of

Baddeley's theory, it could be said that the modeling of working memory for ordered sequences of words has focused on the phonological loop, while largely ignoring the central executive.

Take as an example the Burgess and Hitch (1999) model of the phonological loop, which is a standard bearer in the field and can account for a wide array of phenomena found in immediate serial recall. One might think that this would make it an ideal candidate for generalization to other verbal working-memory tasks. As such, it is surprising just how much the model lacks in terms of machinery to perform updating.

One problem is that the mechanisms of control for the Burgess and Hitch model lie outside of the model proper. That is, the model consists of a set of nodes connected by links that correspond to input, output and internal representations, but this set of nodes is driven by an external homunculus that handles all of the details of task conformity. To get the model to perform other tasks would require a new control process to be devised, but the principles for doing this lie outside the scope of the model.

A second problem is that serial order is represented with a context-timing signal that evolves gradually over time as a moving window of activation. Recall involves replaying this signal to access items by association. As a result, it is not clear how an item can be accessed out of order, which seems necessary for updating operations where reordering is required. Even prepending a word to a sequence requires taking a word presented after a sequence and moving it to the beginning. It appears that entirely new mechanisms of control and access would be required to get the Burgess and Hitch model to perform any verbal working-memory updating tasks.

As a second very different example, consider the integrated theory of list memory in ACT-R (Anderson et al., 1998). It has been used to model performance in a variety of verbal working-memory tasks. It is implemented in a symbolic production-system architecture with sub-symbolic activation of memory objects (chunks). The fact that it makes use of an architecture, and has already been used for multiple tasks is promising. Serial order is represented by positional coding of items within groups and groups within a list. Rehearsal during immediate serial recall acts by retrieving the chunks for groups and items. The act of retrieval increases the activation for the chunks.

Performing a verbal working-memory updating task with this theory would first involve writing a set of production rules. This could clearly be done. Less clear is how memory representations would be used. In serial recall, under ACT-R, the chunks with their positional codes are created during perception of the stimuli and then reactivated repeatedly by rehearsal. But in an updating task, the positions of words change and so the old chunks will no longer properly represent the sequence that should be recalled. A new process would seem to be necessary to handle this. Perhaps new chunks could be constructed for each updated representation. Since this would be a substantial departure from the current models of immediate serial recall, it is unclear what the performance implications of this would be. ACT-R could certainly be used to model updating tasks, but it would require a significant departure from the way memory representations are used in the current model of immediate serial recall.

Note that none of the models discussed so far are of performance in verbal working-memory updating tasks – I am not aware that any such models exist. My additional point is that generalizing from the extant models to updating is neither trivial nor already specified. There are other models that have focused on elements of executive control such as attention and inhibition (e.g., Roelofs, 2003), but they do not deal with verbal working memory and are even further from being directly relevant.

2.1.5 Biological Models of Dynamic Updating

The modeling discussed so far comes from a tradition of taking a “top-down” theoretical approach. The primary currency of these models is behavioral data, and the mechanisms of implementation are justified by wont of their effectiveness in accounting for these data. Since this is also the approach taken in my dissertation, these models are the most directly relevant. However, there is another line of work that has taken a “bottom-up” approach and has tackled issues of updating and maintenance in working memory. This work is motivated by a desire to explain mental processes in terms of biology.

The most relevant effort along these lines is a model of the phonological loop (O'Reilly & Soto, 2002) based on the prefrontal cortex, basal ganglia working-memory (PBWM) model (Frank, Loughry, & O'Reilly, 2001; Hazy et al., 2006; O'Reilly & Frank, 2006). While this work is state of the art in terms of explicitly relating specific neuroanatomy to cognitive function, it is extremely preliminary in terms of realistically accounting for human performance.

A key claim of the PBWM model is that working memory and executive control are two aspects, state and process, of the same system, not two separable components. In this model, active memories are stored in prefrontal cortex. Memories are gated into this store by signals from the basal ganglia as part of a dopamine-based actor-critic reinforcement learning system. Individual pieces of information can be updated independently based on parallel circuits between the basal ganglia and prefrontal cortex. This approach draws a strong connection between working-memory updating and motor control, arguing that both involve actions learned by the same dopamine system.

The PBWM model learns when to gate items into memory slots through reinforcement learning. However, it takes many hundreds of trials to learn simple tasks, and the learning has been described as similar to the way a monkey learns (O'Reilly & Frank, 2006). This stands in contrast to humans who are able to perform updating tasks accurately from the first trial.

O’Reilly and Soto (2002) suggest that a timing signal that cues storage to different slots may come from the cerebellum and that the hippocampus plays a role in binding. However, none of this has yet been implemented. While the ability of the model to perform tasks is reported, no further connection to human performance is provided. This model is far from containing the components necessary to account for performance on tasks that require updating sequences of words. A primary point of contact between this research and the approach I take is in the general concept of a close relationship between working-memory updating and motor control.

2.1.6 Gap in the Literature

On the one hand, the storage of words in verbal working memory has been studied extensively for immediate serial recall, but that work does not handle situations that require updating. On the other hand, recent work has begun to address the biological substrate of working-memory updating in the context of basal ganglia/pre-frontal cortex interactions, but that work cannot yet scale up. In between lies a substantial gap. In fact, my search of the literature has failed to turn up any computational models of the running memory span or n-back tasks, in which ordered sequences of words must be updated.

This is true despite the fact that the running memory task was first used to study memory updating in 1959, with an emphasis already placed on the role of ”behavioral strategy” (Pollack, Johnson, & Knaff, 1959, p. 144).

While the extant work is important, it leads to myopic theories of working memory because of the narrow range of manipulations required of the participant. A recent review of research on order information in working memory declared that “there is a great need for computational modeling that goes beyond tasks of serial recall” (Marshuetz, 2005, p. 335). The work presented here takes immediate serial recall as a starting point, not the sum total of working memory. Only through the modeling of performance will we get beyond categorical descriptions of *which* executive processes are used in particular tasks (e.g., Miyake et al., 2000) to start explaining *how* executive control is implemented.

2.2 Issues to be Addressed

By constructing models I will address a number of theoretical issues that have been raised in the empirical literature on verbal working-memory updating. These include: the extent to which updating is a process separable from the phonological loop; the representation of serial order; the selectivity of updating operations; and the necessity of inhibition in the process of deletion.

2.2.1 Updating as a Separable Executive Process

In the preceding discussion, a distinction was drawn between the performance demands of tasks and the mental processes used in that performance. For updating, this translates into a debate about whether updating is a separate and independent executive process. In an influential paper, Morris and Jones (1990) investigated this question using the running memory task. The participant was presented a sequence of words of unknown length at a steady pace, followed by a recall cue. The participant then attempted to recall a pre-specified number of the last words presented. The dependent measure was the number of words recalled in the correct serial position. Morris and Jones varied the total number of words in the sequence, and had three conditions: quiet, articulatory suppression, and irrelevant speech. They found main effects of list length and condition, but no interaction. They argued that these results indicated that articulatory suppression and irrelevant speech interfere with maintenance in the articulatory loop, while updating takes place in the central executive, and thus these results "show that memory updating is not performed by the articulatory loop" (p.118).

This interpretation of independent processes of maintenance by the articulatory loop and updating by the central executive has been taken quite literally. For example, Van der Linden et al. (1999) performed a PET study in which they subtracted immediate serial recall performance from running memory performance to yield a map of "updating" activity. However, there are reasons to question Morris and Jones' interpretation. As Ruiz, Elosúa, and Lechuga (2005) have argued, Morris and Jones conflate list length with number of updates. However, their procedure does not guarantee that such conflation happened, since there is no measure of what intermediate states the participant is actually in, and thus their strategy is not known.

Postle et al. (2001) used running memory in an fMRI study. They included trials where no updating was necessary. They did not find activations associated with this manipulation and concluded that their imaging evidence

> "calls into question the assumption that the discarding and repositioning operations assumed to be required by the updating task are fundamentally

different from the encoding- and maintenance-related processes that are engaged by all, even the simplest, working memory tasks." (p. 19)

However, the lack of a reliable fMRI result is far from convincing evidence on its own. Whether updating is a separate process from that of maintenance remains an open question.

2.2.2 Representations of Serial Order

The problem of how serial order in word sequences is represented is by no means limited to working-memory updating. Indeed, it is a central concern of many of the immediate serial-recall models discussed before (e.g., Henson, 1998). However, since memory representations are manipulated during updating, this may provide unique insight into the structure of those representations.

There are at least two relevant dimensions along which representations of serial order may vary. One dimension is the manner in which order is encoded. Options here include links between words (link models), ordinal tags marking words as first, second, etc. (tag models), or an array where words are implicitly ordered by wont of the slots that hold them (slot models). A second dimension is whether representations are directly modified and refreshed when necessary (malleable), or whether the representations are static and must be replaced when necessary (fixed).

Different representations of serial order will differ in the time taken to update them in particular ways. For example, consider appending and prepending words to a sequence. For a malleable-link model, prepending may not be much different from appending. In either case, a single link needs to be modified to integrate the word into the list. On the other hand, for a malleable-slot model, appending is easy – the word is placed in the next slot, but prepending would require shifting all the words over a slot to open up the first slot for the new word.

2.2.3 Selectivity of Updating

Kessler and Meiran (2006) investigated working-memory updating where the value of numbers in memory needed to be modified. They used a version of the Garavan (1998) task. Participants needed to maintain two numerical counts – one associated with rectangles and one with triangles. At the beginning of a sequence, a starting count for each type of shape was presented. Then a series of trials occurred. On each trial, either a triangle or rectangle was presented with a positive or negative value to be added to the corresponding count. On some trials, the value to be added was zero, thus requiring no update. The participants indicated with a key press when they were ready to move to the next trial. At the end of the sequence, they were asked to provide the correct count for each type of shape. A measure of the time to update was calculated from the difference in reaction time between trials that required an update and those that did not. In a comparison task, only a single count and a single type of shape were used.

Kessler and Meiran found that participants were slower to update on blocks where they were updating two counts rather than one, but an additional memory load that was not being updated had no effect. From this they concluded that all of the potentially updateable items in memory had to be updated when any one item needed to be updated. In other words, they claim that updates are not selective when more than one updateable item is in memory. It remains to be seen whether the claims made by Kessler and Meiran are applicable for the updating operations of adding and deleting words from sequences.

2.2.4 Necessity of Inhibition/Suppression for Removing Words

Another open issue is whether deletion operations involve active inhibition and suppression or just passive neglect. Palladino et al. (2001) found that participants who performed better at the running memory task also performed better on a later surprise test of memory for the words they needed to discard during the task. This suggests that performing better at updating did not depend on doing a better job of actively deleting the words to be discarded. However, the long length of time between the initial task and the surprise test, and other details of this procedure make it unclear clear whether this is relevant to immediate updating operations.

De Beni and Palladino (2004) investigated working-memory updating using a semantic updating task. In this task, subjects heard a sequence of concrete and abstract nouns, and the participant was instructed to recall the concrete nouns that named a specified number of the smallest objects. Based on a comparison of intrusion errors by old and young participants, they claimed that suppression of no longer relevant information was crucial to working-memory updating. However, the semantic updating task has features that may encourage the use of memory systems other than verbal working memory. In particular, the use of concrete nouns and the requirement to judge size may have encouraged the use of visual encoding strategies. Furthermore, the use of new items on each trial may have emphasized the role of long-term memory activation.

Hartman, Dumas, and Nielsen (2001) used a modified delayed matching-to-sample (DMTS) task to investigate working-memory updating. Participants were presented with three stimuli. After a filled delay, three test items were presented for a forced-choice response about which of the three was in the original set. This task was argued to involve updating due to the need to activate and maintain new information on each trial, as well as the need to ignore information from the previous trial. Increasing the inter-trial interval improved performance, leading the authors to claim that suppression plays an important role in memory updating. However, this improvement could be attributed just as well to decay as to "more opportunity for inhibiting irrelevant information" (p. 23).

Ruiz, et al. (2005) analyzed item errors in the running memory task, and found that recently discarded items are quite likely to be incorrectly included in recall. They argued that these items were not actively inhibited or suppressed, but rather were treated similarly to the items in the current list. However, this may be a special case, since participants in the experiment were performing in a very passive manner and were not doing active maintenance. In summary, the data presented here are far from conclusive about the role of inhibition in deletion.

All of the issues just discussed will be addressed through the experimentation and modeling reported in this dissertation.

2.3 Executive-Process Interactive-Control Architecture

The modeling in my dissertation takes the Executive Process/Interactive Control (EPIC) architecture as its starting point (Kieras & Meyer, 1997; Kieras & Meyer, 1998; Meyer & Kieras, 1997). Modifications have been made to the original architecture, and so this is correctly described as an updated version of EPIC. EPIC has a symbolic production system. It is an end-to-end architecture that models task performance from the initial perception of the stimulus to the execution of an action. Hand in hand with this, EPIC is an embodied architecture that models the peripheral perceptual and motor apparatus as well as the central cognitive system. These are important features for the modeling of human behavioral data, because a significant portion of the response time in many task situations is due to peripheral processes. This attention to the entire perceptual-cognitive-motor arc of processing allows EPIC to model typical behavioral measures including the timing of responses.

Conceptually, there are three main components to an EPIC model: the task environment, the cognitive architecture, and the production rules. First, there is the task environment, which simulates everything in the external world that the cognitive agent interacts with. It presents stimuli to the agent and can detect and record the results of overt actions performed by the agent. When modeling an experiment, the task environment typically mimics the computer program and interface that the human subjects interacted with. Second, there is the cognitive architecture, which represents the information-processing structure of the mind/brain. In general, this remains constant across different models, but it has not yet been fully determined and is still under development. The architecture also contains a number of parameters that may vary from individual to individual or from context to context. The architecture and its parameters are discussed further in the following sections. Third, there are the production rules that represent learned procedural knowledge about how to perform tasks. Some production rules may be general and employed across a range of tasks, while others will be task specific. The production rules are discussed later in the chapters reporting on specific models.

2.3.1 Architecture

A simplified schematic of the EPIC architecture is shown in Figure 2.1. The EPIC architecture consists of a number of processing modules that execute in parallel. On the perceptual side, these modules include an auditory perceptual processor receiving simulated sound input from the ears and a visual perceptual processor receiving simulated visual input from the eyes. On the motor side, these modules include a vocal motor processor controlling a simulated mouth and vocal apparatus and a manual motor processor controlling simulated hands and fingers. Centrally, there is a cognitive processor, which includes a production-rule interpreter. In addition to these processors, there is a production-rule memory that stores condition-action rules, and there is a collection of working memory buffers that store the dynamic state available to the cognitive processor for production-rule matching.

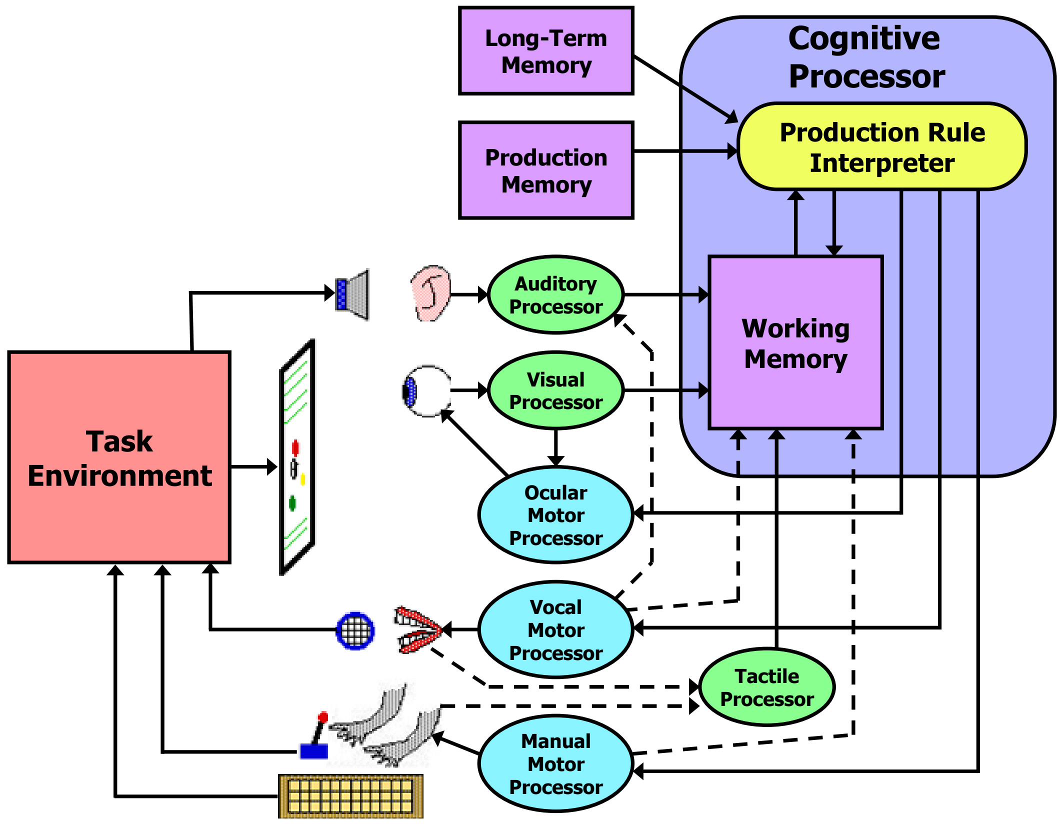


Figure 2.1 Schematic of the EPIC architecture (fromMeyer & Kieras, 1997). Perceptual processors are shown in green ovals, motor processors in blue ovals, and memory stores in purple boxes.

The cognitive processor operates cyclically. On each cycle, all of the production rules are matched against the current state of working memory. Every rule instantiation that has its conditions satisfied on a given cycle will have its actions executed on that cycle. These actions can modify the contents of working memory or send commands to the motor processors to prepare or execute motor programs.

2.3.1.1 EPIC's Working Memory

It is important to note that EPIC's working memory does not map directly onto the theoretical construct of working memory (e.g., Baddeley, 1986) discussed previously. EPIC's working memory is not a unitary structure, but rather a collection of different buffers. A distinction can be made between "control state" and "perceptual-motor state". Control state is created, managed and used by production rules in the cognitive processor. It is meaningful only in the sense that it is used in a consistent manner across rules. It follows a "destructive-state-modification" paradigm, which is to say items can be created and destroyed by production rules (R. L. Lewis, personal communication, September 26, 2005). There are a number of control-state working-memory buffers, including ones for goals representing what the system is trying to accomplish, steps representing where the system is in an extended procedure, and notes and tags maintaining other information about the state of the system that need to persist across control cycles.

Perceptual-motor state can be accessed by the cognitive processor, but it is created and modified only by the perceptual and motor processors. This includes status information about the current operation of the processors, for example, indicating if the vocal motor processor is currently busy generating an utterance. It also includes the percepts input by the perceptual processors. For example, if a red square is presented on the computer screen in the task environment, and it is within the fovea of the eye, then after an appropriate time for detection and identification, "red square" would be entered into visual working memory and become available for matching by production rules. Each perceptual and motor processor has its own working memory buffer for which it controls the contents.

While much more can be said about the general architecture and operation of EPIC, the reader is referred to various extant papers that provide this detail (e.g., Kieras & Meyer, 1997; Kieras & Meyer, 1998; Meyer & Kieras, 1997). Instead, the focus will be turned specifically to the portions of the architecture most relevant to the modeling of verbal working-memory tasks.

2.3.2 Verbal Working Memory in EPIC

The implementation of verbal working memory within the EPIC architecture is central to the modeling in this dissertation. This construal of verbal working memory is heavily dependent upon the embodied nature of EPIC and its pervasive parallelism, and is an instantiation of Baddeley's phonological loop (Baddeley, 1986). This interpretation of verbal working memory makes use of EPIC's working memory buffers, but it is not equivalent to them.

The implementation of verbal working memory in EPIC was initially done in the context of the "Boulder" model of immediate serial recall (ISR) (Kieras, Meyer, Mueller, & Seymour, 1998; Kieras, Meyer, Mueller, & Seymour, 1999). Further work was done with a modified version of the EPIC architecture to study the role of strategy in ISR performance, particularly for serial position curves (Mueller, 2002). The modeling presented here uses some of the features of this modified architecture.

The three processors most critical to this dissertation are the cognitive processor, the auditory perceptual processor, and the vocal motor processor. The auditory perceptual processor receives speech input and transforms it into perceptual representations that are placed in auditory working memory. Each word is represented in a separate speech object in working memory. The time from the start of presentation until the object is placed in working memory depends on both the duration of the word and its source. The source may be external (i.e., a word said by another human or a recording), overt (i.e. a word that the model says out loud), or covert (i.e., a word that the model says to itself).

A speech object in working memory contains several pieces of information. It has a unique identifier; it has phonological content (referred to as the speech item); it has a reference to the next speech object (referred to as the speech link); it has a source (external, overt, or covert); and it has a marker which conveys prosodic and contextual information about the location of the item in a spoken sequence (beginning, middle or end). The speech object may decay out of working memory in an all or none fashion. The probability of decay is determined by the source (external, overt, or covert) of the object and the phonological similarity of the words in the active word set. Parameters are also available within the architecture for the speech link to decay independently of the speech object as a whole. The use of these parameters is discussed when relevant subsequently.

The speech item (the phonological content) is not stored directly in the speech object. To be used, it must be retrieved in a production rule action using the speech object's unique identifier as a key. In this way, the architecture remains agnostic about where the content itself is stored. When a series of words is heard by the model, the internal representation will be a chain of serially linked speech objects. There is a link to the first object in the chain, referred to as the start tag, which is not part of a speech object. The link in the last item in a chain has nothing to point to, and instead acts as an end-of-chain tag. The start tag and the end tag can decay in the same way as the speech links.

Congruent with Baddeley's conception of the phonological loop, the model can preclude the loss of speech information through decay by articulating the words either covertly or overtly using the vocal motor processor. The production of speech begins with production rules that retrieve the phonological content from the speech object and send it in a motor command to the vocal motor processor. The time from sending the command until the start of actual speech production depends on both the time for motor preparation and the time to initiate the covert or overt action. The description of what to articulate derives directly from the content of the corresponding perceptual speech object. Once the motor processor commences articulation of a word, it is perceived by the auditory perceptual processor as already described. Covert speech is sent by a privileged route from the vocal motor processor to the auditory perceptual processor.

2.3.2.1 Perceptual-Motor Implementation of Phonological Loop

The EPIC approach to modeling verbal working memory is well supported by a diverse body of literature employing multiple converging methods of investigation. One key property is the perceptual-motor nature of the phonological loop. Extensive evidence, including results from behavioral studies (e.g., Jones, Macken, & Nicholls, 2004), neuroimaging studies (e.g., Awh et al., 1996; Jonides, Lacey, & Nee, 2005; Smith & Jonides, 1999), and patient populations (e.g., M. Wilson, 2001), points to the auditory/perceptual nature of the acquisition and storage of verbal information, and the output planning and vocal motoric nature of the rehearsal process. Furthermore, recent neural-network modeling of language acquisition has emphasized the important role of a tight loop between speech perception and speech production (Rohde, 2002). In particular, this research supports EPIC's reciprocal paths from perceived speech to produced speech, and from covert produced speech directly back to perceptual mechanisms.

2.3.2.2 Decaying Episodic Verbal Memories

A second key property of the EPIC approach, closely related to the first, is the episodic nature of the memory representations for speech. Whenever rehearsal is performed, the produced speech is perceived and encoded as new speech objects. An alternative approach would be for rehearsal to strengthen an already existing representation. Our introspective experience is certainly more consistent with the episodic approach. For example, if I say "dog, dog, dog" it is clear to me that I said the word 'dog' three times and did not just yell "DOG!" once loudly. Interestingly, however, the default assumption in many models of ISR has involved the alternative strengthening approach (e.g., Anderson et al., 1998). Nonetheless, recent work has argued for the episodic approach.

Baddeley (2000; 2002) claims that an episodic buffer is a necessary component of his (formerly tripartite) theory of working memory. He states that an episodic buffer is needed to explain how people can bind together information over the short-term. Henson (1998) developed a model of immediate serial recall based on tokens that are "episodic records that a particular item occurred in a particular spatiotemporal context" (p. 84). This effort was limited to modeling immediate serial recall, but it successfully accounted for a large and diverse set of effects in the literature. Nairne (2002) takes a significantly different approach to working memory, arguing that it is primarily a cue-driven retrieval process. However, he too has suggested that "discrete rehearsals might produce copies of an item in memory" (p. 63). He distinguishes this from theories where multiple presentations strengthen a single memory trace. Consistent with these ideas of episodic working memory, speech objects in EPIC bind together phonological content and serial order information, and each time a word is perceived and encoded, a new speech object is created.

2.3.2.3 Speech Objects with Bound Content and Order Information

A third key property of EPIC's verbal-working memory is the linked-list nature of the serial-order representation. Basic findings about the effects of word length and phonological similarity on memory span (Kieras et al., 1999) and detailed modeling of serial position curves (Mueller, 2002) have been performed successfully with this assumption. However, the literature often sites Henson et al. (1996) as evidence that a linked-list model of serial order cannot account for certain aspects of ISR performance. But Henson's argument relies on an assumption that links are pointing to the content itself, which begs the question of how each link is bound to its associated content. In the framework used in this dissertation, the content and the link are bound together by both being part of a speech object. Furthermore, the links point to the speech objects and not the content. This solves the problem of how order information and item information are related to each other, and it disarms the claims of Henson et al., because content can now be lost while the integrity of the chain of speech objects is preserved.

2.3.3 Meta-Comments on Modeling

2.3.3.1 Pedagogical Method

The general approach to modeling that I use here is the pedagogical method. This involves constructing a sequence of models starting with the simplest reasonable model that performs the task and only adding complexity as it is required to deal with significant deviant features of the data. This has the advantage of helping to insure that the constructs in a model are necessary to account for human performance. A disadvantage of this approach is that it is more data-driven and exploratory than theoretically motivated and purely predictive. Given the lack of any existing model of working-memory updating, the exploratory approach is warranted at this time. As the theory and modeling matures, a shift to a more top-down predictive approach will be appropriate.

2.3.3.2 Parameters

There are a number of parameters in the EPIC architecture, each of which is associated with a particular processor. Refer to Appendix A for a list of all the parameters used in modeling. These parameters fall into general categories of standard, typical and free (Kieras et al., 1999). Standard parameters are those with well established values that are not changed across models of different tasks. For example, the 50 millisecond cognitive cycle time is a standard parameter. Typical parameters have values that can be based on past experimental and modeling work, but could potentially depend on the particular task context. For example, the time to initiate motor movements may be determined from past work, but could vary for a particular task if it required a unique movement with different demands. Free parameters are those for which the values may depend on the particular task and/or participants, or simply have not been established yet in the literature. These parameters can potentially be determined by finding the values that provide the best fit for a given model. For example, in the auditory processor, the decay parameters for speech objects are free parameters that may vary from participant to participant, and could depend on other properties of the words as well.

As a general principle, I have tried to minimize the amount of parameter fitting. This was in part a practical decision. The more free parameters being fit, the larger the search space and the more computationally intensive and time consuming the fitting process. This was compounded by the fact that there are multiple dependent variables being evaluated (response accuracy, reaction time, articulation time, serial position curves), so attempting to define the overall best fit is challenging. Also, the focus of this work is on the architecture and strategy of updating, not on the quantitative values of the parameters per se. Thus I did not to focus effort on nuances of parameter values.

However, there is a theoretically motivated side to this as well. Since I am modeling performance for a set of closely related tasks, and in many cases there are within-subject data across multiple tasks, the parameter values should be largely constant across the tasks, given the underlying claim that the same cognitive processes and structures are being used throughout. The parameter values could vary across experiments due to different participants being tested, and there are times when I accepted that this seemed to best explain the data. However, usually I kept this to a minimum, as I was testing participants from a fairly stable population, with more or less the same equipment and context.

2.3.3.3 Model Results

Through preliminary investigations, I explored the number of model runs that were necessary to achieve stable measures of predicted performance. For all of the reported model predictions, I consistently used runs of 100 trials per condition. A single trial consisted of up to 10 recall opportunities. Due to the way in which the data were aggregated, this led to between 100 and 1000 individual measurements contributing to each task-by-word-set-by-sequence-length mean data point.

CHAPTER 3

Methodological Review and New Rehearsal Tasks

In order to create models of performance, it is necessary to have clean empirical data. A number of different tasks have been used to study verbal working-memory updating in the past, but there are serious methodological concerns with them. One concern is the lack of control on the strategies used by participants. It is important to keep in mind Newell’s (1973b) classic injunctions to “know the method your subject is using to perform the experimental task” (p. 294), and to “never average over methods” (p. 295). Immediate serial recall, the running memory task, and the n-back task all pose problems that are discussed here. These past tasks raise interesting issues, but there is a need for data collected with modeling in mind. Newly developed rehearsal tasks that overcome a number of methodological issues are presented and used in this dissertation.

3.1 Immediate Serial Recall Task

In the ISR task, the participant is presented with a sequence of words (or digits or letters) either auditorily or visually, typically at a pace of about 1 word per second, followed immediately by a recall cue. At this point, the participant attempts to recall the sequence in the same order as it was presented, either verbally, in writing, or by typing. As one of the most used tasks in the study of short-term memory there are many variations on this general scheme. Typically, ISR has been considered a task of “pure storage” involving the “passive maintenance of memory loads” (Morris & Jones, 1990, p. 113). As such, its relevance here may be questioned. Nevertheless, there are at least three reasons why this task is highly relevant to the study of dynamic updating.

First, conceptually, there is a sense in which ISR does involve updating. The stimuli are often presented sequentially, not en masse, so the memory representation of the sequence must be modified in that the new words are added to the end as they are presented. It is an open empirical question whether this form of updating uses underlying processes that are the same or different from other forms of updating. As such, understanding the updating in ISR may be a basic building block to a full theory, and it provides a starting point for the series of studies presented here.

Second, recent work has argued that performance of ISR is considerably more complex, active, and strategic than has typically (but not universally, e.g., Newell, 1973a) been appreciated (Brooks & Watkins, 1990; Mueller, 2002). One implication of this is that executive control plays a significant role in ISR during both rehearsal and recall (Kieras et al., 1998; Kieras et al., 1999). A second implication is that if the challenges of modeling and interpretation that come with multiple strategies apply to ISR, then they almost certainly apply to other similar tasks that add additional updating demands. Thus the lessons and cautions that apply to understanding and modeling ISR performance need to be heeded when considering other tasks such as n-back and running memory span. In response to these issues, the tasks in this dissertation were developed with strategic control strongly in mind.

Third, a number of analyses have been developed for ISR that may pay dividends if adapted to tasks involving serial order and working-memory updating. These include proportion of complete correct recall versus list length in order to study the list length effect, multiple types of serial position functions that allow the relative roles of item and order information to be evaluated (Drewnowski & Murdock, 1980; Mueller, 2002), and response-type functions that classify what types of errors are made in each recall position (Henson, 1998; Mueller, 2002). In addition, various measures of timing have been used with ISR, including response latency (i.e., time between the recall cue and the beginning of the participant saying the first word in the response) and articulation time (i.e., the time from starting the utterance of the first word until finishing the utterance of the last word) (Cowan, 1992; Farrell & Lewandowsky, 2004). These analyses provide constraints on underlying processes that go beyond the classic memory span measure of the list length at which complete recall occurs fifty percent of the time. In this dissertation, these analyses are adapted to updating tasks for more precise evaluation of models.

3.2 Running Memory Task

The running memory task must be considered a candidate for studying dynamic memory updating, because its use for such purposes dates back 45 years. In the running memory task, the participant is presented with a sequence of words (or digits or letters) of unknown length at a steady pace, followed by a recall cue. The participant then attempts to recall the words from the end of the sequence. As originally used by Pollack et al. (1959), the participant writes down as many of the words from the end of the sequence as he/she can, with no constraint on order of recall, but with the order specified by place-ment on the page. In more recent versions of the task, the participant attempts to say a pre-specified number of the words from the end of the sequence in the correct order (Morris & Jones, 1990). For example, ten words might be presented, and the participant is asked to recall the last six. The running memory task expands upon the updating in ISR by potentially including the need to drop words from earlier in the list as well as the need to add new words to the end of the list as they are presented. However, there are a number of concerns with this task.

First, presented list length is conflated with the number of updates, but the procedure does not guarantee that this is the case, since we have no measure of what intermediate states the participant is actually in during word presentation.

Second, uncertainty plays a significant role in the running memory task. In fact, uncertainty was a primary focus of the original paper and the task was designed with this in mind (Pollack et al., 1959). If participants know the length of the full sequence, since no response is asked of the participant until the entire sequence has been presented, they can simply ignore the earlier words. For example, if they know that ten words are to be presented, and that they should recall the last six, then they can simply ignore the first four words. However, if they do not know how many words will be presented, then it is unclear how participants will cope. The difference in performance between lists of known and unknown length was tested in the original paper, and across a range of conditions, performance was better with known length lists (Pollack et al., 1959). This could be due to a cost associated with updating, or it could be due to a shift in strategy associated with the uncertainty.

Third, looking at the serial position curves raises serious doubts about how subjects perform the task. For example, in Morris and Jones' Experiment 2, with no interference and even when the length of the presented list, six items, was the same as the number of items to be recalled, there is no primacy effect, just a consistent recency slope across all items. This is significantly different from the typical u-shaped serial position curve, with both primacy and recency effects, which is found in ISR. This difference suggests that participants may not be engaging in rehearsal at all. Ruiz, et al. (2005) pursued this further and concluded that in their running memory task "participants are not updating their articulatory loops as items are being presented, even after having been instructed to do so" (p. 905). Given a difficult task, and little incentive to perform well, participants appear to be adopting a suboptimal passive strategy.

All three of these concerns lead to the conclusion that the running memory task is strategically unconstrained. As such, it is not a particularly good task for probing memory updating or for use as a starting point in developing models of performance.

3.3 N-Back Task

Another task that must be considered in the context of verbal working-memory updating is the n-back task. In the n-back task, a single item is presented (often visually) on each trial. The participant indicates (usually by pressing one of two buttons) whether the item matches the item that was presented *n* trials previously, where *n* is typically one, two, or three. The items are most often consonant letters. The letters are often presented in randomly varying case to encourage phonological, as opposed to visual, encoding. The inter-stimulus interval is typically about 2500 milliseconds. And about 33% of the trials are usually matches.

My n-back task literature review indicates that use of the n-back task to study the relationship of dynamic memory and "central organizing processes of the brain" (p. 357) dates back to the use of a recall-based variant by Kirchner (1958). In that study, which used a row of lights for stimuli, and lagged recall with a spatially congruent row of buttons for responses, Kirchner attributed poorer performance by older adults to a slowing down of central processes involved with updating. A few years later, the matching version of the task was first used under the moniker "running matching memory" to investigate the role of context in running memory (Moore & Ross, 1963). However, these promising early investigations not withstanding, there is a serious dearth of behavioral research on the n-back task, as attested by my own literature search and as asserted by others. Conway (2005) states that

> "Although the *n*-back task is arguably the current gold standard measure of working memory capacity in the cognitive neuroscience literature (for a review, see Kane & Engle, 2002), almost no behavioral research has been conducted to validate it." (p.780)

The behavioral results can be summarized as showing that accuracy decreases and response time increases as *n* increases from one to three. Period. Despite this lack of empirical foundation, n-back has been labeled "a paradigmatic case of working memory" (Jonides et al., 1997, p. 463) and the "benchmark task" for the executive control process of updating (Shimamura, 2000, p. 316). It has been used in well over 100 studies of working memory, the vast majority of which use brain imaging (e.g., fMRI as in Callicott et al., 1999; Cohen et al., 1997), electrophysiology (e.g., EEG as in McEvoy, Smith, & Gevins, 1998; Ross & Segalowitz, 2000), or patient groups (e.g., schizophrenics as in Krieger, Lis, Cetin, Gallhofer, & Meyer-Lindenberg, 2005; Thermenos et al., 2005).

While many studies have simply used n-back as a vague index of working-memory ability, a number of studies have made dubious inferences about the brain correlates of working memory and executive control processes based on confounded results from the n-back task. Given the lack of formal modeling and extensive empirical investigation, these inferences have required "intuitive" analyses of the mental processes involved in the task (e.g., Cohen et al., 1997; Jonides et al., 1997). In these studies, the change in activation of a brain area as a function of memory load and time is used to map the area to a mental process that is assumed to vary with memory load and time in the same fashion. However, there are a number of reasons to question this loose theorizing and to suggest that the n-back task is not a strong starting point for modeling memory updating.

First, the task is quite complicated in that the subject must coordinate memory maintenance and updating with the need to compare the current and *n*th previous items. Jonides et al. (1997) suggested seven different processes that might occur on each trial, and while the ordering is somewhat constrained, it is far from determined.

Second, not only is the ordering between processes an issue, but so are the details of the processes themselves. One such process is subvocal rehearsal. Perhaps the most obvious general rehearsal strategy would be to say a rolling sequence. For example, if *n* is four, say “a,b,c,d” on the first trial, followed by “b,c,d,e,” and “c,d,e,f,” and onwards (where ‘a’ is the stimulus presented on the first trial, ‘b’ on the second, etc.). However, Verhaeghen and Basak (2005) have pointed out that another strategy would be to use fixed-slot rehearsal. For example, again for an *n* of four, one would say “a,b,c,d,” followed by “e,b,c,d,” and “e,f,c,d” and so on. Different participants may use different rehearsal strategies.

Third, another complicating factor arises because a trial block in the n-back task is typically 20 or more trials in length. When the participant makes an error on a trial, they must attempt to recover while the trials keep going. Do they still have some correct items in memory? Do they start over as if it were the beginning of the block? As *n* increases and more trials are incorrect, more and more trials fall into the ambiguous recovery periods following error trials where strategy is likely to be different.

Fourth, the fact that the n-back task is a matching task has serious implications for the stimulus sequences. Typically, about 33% of the trials are matches, where the current stimulus matches the *n*th previous stimulus, and about 7% of the trials are decoys, where the current stimulus matches a recent stimulus other than the *n*th previous one. As a result, it is inevitable that there are many clustered repetitions of the same stimuli during a block, and that the patterns of these repetitions differ systematically as *n* is varied. Moore and Ross (1963), in one the first studies to use the n-back task, showed that these repetitions have a significant effect on performance. However, the reasons for this effect, the strategies that this patterning may encourage, and the differences in basic memory maintenance that these repetitions cause, are unknown.

Thus, while it is desirable to understand the memory updating that takes place in the n-back task, since it is being used so extensively in the cognitive neuroscience literature, it would be preferable to begin data collection and modeling with tasks that are more amenable to control and analysis.

3.4 New Rehearsal Tasks

In summary, the various tasks used previously to study updating in verbal working memory have desirable features but also significant shortcomings for current purposes. As a result, new tasks have been developed for this dissertation that better suit the current focus on updating of ordered sequences of words. The key desired task requirements include: (1) a need to remember serial order information, (2) a need to partially update the contents of memory, (3) careful control of strategy, (4) use of chronometric analysis, (5) localization of errors, (6) comparison of different updating operations, and (7) relevance to other memory updating tasks. The way in which the new tasks meet these requirements will be discussed once the tasks are described.

The four new rehearsal tasks all include a number of independent trials grouped into trial blocks. The same set of words is used throughout a block. Each trial consists of a startup period followed by a sequence of subtrials (see Figure 3.1). In the startup period, an initial sequence of words is presented auditorily with very short inter-stimulus intervals (ISIs), followed by an auditory go tone. The tone signals the participant to respond by recalling the initial sequence immediately and rapidly in the order presented. If the participant recalls correctly, then, after a response-stimulus interval (RSI), the first subtrial begins with the auditory presentation of a new word. Next, the participant has to respond immediately and rapidly with an updated sequence. After another RSI, the second subtrial occurs in the same way as the first. Subtrials continue until the participant makes a mistake during recall, or the maximum number of subtrials is reached.

The content of the updated word sequences depends on the particular task, as shown in Figure 3.1. In the cumulative forward rehearsal (CFR) task, each new word is added to the end of the sequence (i.e., appended). In the rolling forward rehearsal (RFR) task, each new word is appended to the sequence and the current first word in the sequence is omitted. In the cumulative backward rehearsal (CBR) task, each new word is added to the beginning of the sequence (i.e., prepended). And finally, in the rolling backward rehearsal (RBR) task, each new word is prepended to the sequence and the current last word in the sequence is omitted.

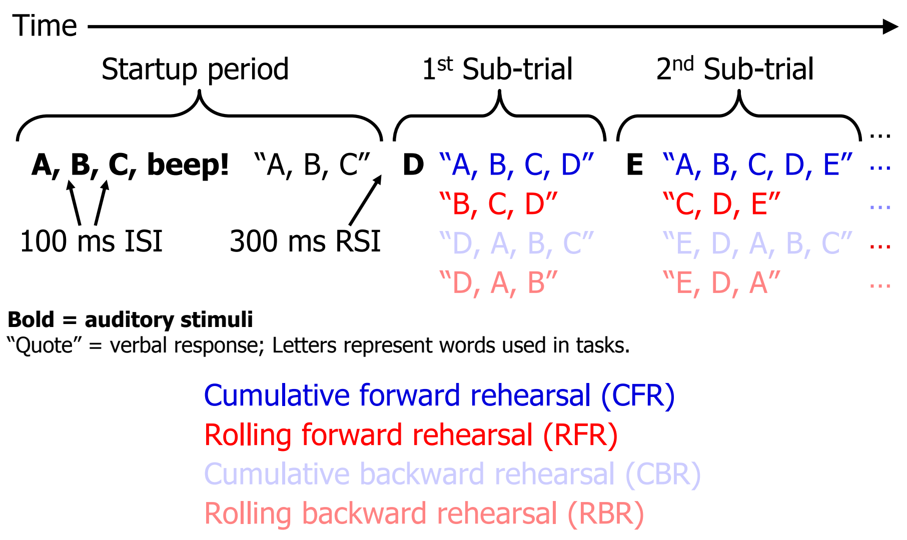


Figure 3.1 Schematic outline for a trial in each of the new rehearsal tasks. In this example, the length of the startup sequence is three words.

For each task, participants are encouraged to perform accurately, initiate responses promptly, and recite quickly through the use a point scheme that translates into a monetary reward. The latency of responses (i.e., the reaction time) is measured with an electronic voice key. The overall accuracy of the response and the duration of the response from the start to the end (i.e., the articulation time) are measured by the experimenter pressing a key. The exact sequence of words recited during the startup period and on each subtrial is transcribed from a tape-recording after the fact.

These tasks satisfy the key requirements listed previously in the following way:

1. A heavy emphasis is placed on serial order information because the participant must recall the words in the specified order to be correct. Furthermore, the same words are used repeatedly throughout a block of trials, placing the focus on the order of the words as opposed to their identity.

2. Partial memory updating is emphasized because, on each subtrial, one word is added to the sequence, and at most one word is removed from the sequence, while the rest of it remains the same.

3. Strategy is controlled by the requirement for overt rehearsal and the tight self-paced timing of the trials. During the startup period, the ISIs are only 100 ms in order to discourage covert rehearsal during the gaps between words. During subtrials, participants should be rehearsing overtly as soon as they hear each new word. And between subtrials, there is a brief 300 ms RSI that allows only enough time for breathing before the next rehearsal.

4. Chronometric analysis is possible because reaction times and articulation times are collected. These data provide an important constraint on models. All too often, models of working memory do not take realistic timing into account, divorcing them from the constraints of actual performance.

5. Errors are localized to a specific update, since the entire current sequence must be overtly recited on each subtrial. This is a substantial improvement over running memory span where memory contents are only queried at the end of a run, or the n-back task where only knowledge of the *n*th previous item is tested on each update. Furthermore, overt recall allows for the mistaken words to be analyzed in detail at each serial position.

6. Different updating operations are compared in a way that has not been possible before, since all four tasks use the same trial structure. The startup period and the presentation of stimuli on subtrials are identical in all four tasks. The tasks differ only in the correct updated response on subtrials. Accuracy and timing can be compared across updating operations while holding memory load and other aspects of the task constant.

7. The results are relevant to understanding performance of memory updating tasks in the literature. The CFR task shares the same updating requirements as ISR. RFR

has the same basic updating pattern as the running memory task and the n-back task. Insights from analyzing the CFR and RFR tasks should be applicable to the current literature.

While the design of the rehearsal tasks was not based on a particular antecedent, it turns out that there are precedents in the literature. Buschke and Hinrichs (1968) used procedures with overt rehearsal similar to the CFR and CBR tasks. Participants were presented a sequence of numbers and were asked to recite the current sequence after the presentation of each number. Like me, Buschke and Hinrichs were interested in controlling rehearsal strategy. However, they disregarded recall order in scoring responses, thus removing any emphasis on memory for serial order. As such, it is hard to relate their results to the present research.

Miyake et al. (2000) used a letter memory task with rehearsal similar to RFR. Participants heard a sequence of letters, and were asked to recite the four most recent letters after each was presented. Miyake et al. did this to insure that participants were continuously updating working memory. They measured only the proportion of letters reported correctly in the final recall at the end of the sequence. This task was one of many that they used to stress different putative executive functions. In a latent variable analysis, they found that the executive function of updating was highly correlated with performance on the letter memory task. As discussed earlier, this work is too high level to provide a mechanistic account of performance.

While the new tasks used in this dissertation have some precedent in the literature, no such tasks have ever been used to compare different verbal working-memory updating operations as reported here.

CHAPTER 4

Experiment 1: Appending Words to Sequences

The first experiment begins the investigation of verbal working-memory updating with a situation in which the participant appended words to the end of a sequence. It provides an initial test of the cumulative forward rehearsal (CFR) task, and it establishes that performance of the rehearsal tasks makes use of the same memory processes as the canonical working-memory task of immediate serial recall. A model of CFR is developed that embodies an initial theory of updating performance. In addition, a model of ISR performance is developed as a point of reference.

Appending words to the end of a sequence could be considered the most basic sort of updating. Indeed, this sort of updating is required in immediate serial recall, a canonical verbal working-memory task that is often considered a task of pure storage. In other words, this form of updating is so ubiquitous that its existence is often overlooked. The approach taken here is to start by analyzing this basic sort of updating. The proposed explanation for how it is performed will then be generalized to other updating tasks. Only if forced by the data will the theory be expanded and complicated.

The rehearsal task that requires appending words to the end of a sequence is CFR. In order to establish the generality and relevance of this task to the larger study of working memory, I show that the mental processes involved in CFR are the same as those used in other situations demanding working-memory performance. The relevance of ISR to a wide range of performance is well established (Baddeley, 1986). By showing that there are systematic relationships between ISR and CFR, I established that CFR is relevant to understanding working-memory performance more generally.

In Experiment 1, a within-subjects design was used with the ISR and CFR tasks. Individual memory performance on the two tasks should be correlated if the tasks engage the same mental processes. Furthermore, two sets of words with varying articulation times were used. A third task was used to establish articulatory durations in a principled way (Mueller, Seymour, Kieras, & Meyer, 2003). A well established result for ISR is the articulatory duration effect: Memory span decreases as the articulatory duration of the words in the sequence increases (Mueller et al., 2003). If the phonological loop plays a role in CFR that is similar to its role in ISR, then an articulatory duration effect should occur in CFR as well.

Beyond the empirical comparisons of CFR to ISR, the data from this experiment provide a basis for initial modeling. A perceptual-motor approach based on EPIC is taken here to account for the pattern of data found in CFR.

4.1 Method

4.1.1 Participants

The participants were 8 undergraduate students at the University of Michigan with normal cognitive function. They were between the ages of 18 and 30, right-handed, and spoke English as their first language. They were paid eight dollars per hour for their participation in 2 sessions of approximately 2 hours each on separate days, and earned additional monetary bonuses based on performance that typically amounted to a total of five to ten dollars.

Two additional participants (beyond the eight analyzed) were excluded from analysis and replaced with other participants due to poor, inconsistent performance and non-compliance with instructions as judged by the research assistant before results were analyzed.

4.1.2 Apparatus

The experiment was run on a Dell Pentium III personal computer with Microsoft Windows 98 using special purpose software and E-Prime 1.0 (Schneider, Eschman, & Zuccolotto, 2002). Vocal reaction times were collected using the PST Serial Response Box. Visual feedback was provided on a standard CRT display, and auditory stimuli and feedback were provided through headphones set to a comfortable volume. An experimenter sat nearby to record accuracy and articulation times, and to communicate with the participant via microphones and headphones as needed.

4.1.3 Materials

Two experimental sets and one practice set of 10 words each were used. The words in Word Set 1 were each one syllable, and those in Word Set 3 were each three syllables. The phonological similarities of the experimental word sets were equated on mean onset dissimilarity by using the PSIMETRICA technique (Mueller et al., 2003). The words in the experimental sets were all nouns selected to be similarly low in concreteness and roughly matched on familiarity and Kučera and Francis (1967) written frequency as reported in the MRC Psycholinguistic Database (M. D. Wilson, 1988). The words and relevant measurements are listed in Table 4.1. Word Set 2, which is used only in later experiments, is included in the table for ease of comparison.

Table 4.1 Words used in Experiments 1 through 4.

	Word Set			
	1 (short)	2 (medium)	3 (long)	Practice
Words	dare	belief	advantage	button
	fate	delight	behavior	candle
	hint	glory	circumstance	needle
	mood	judgment	fantasy	pencil
	oath	kindness	misery	pocket
	plea	logic	narrowness	robot
	rush	mischief	occasion	shovel
	truth	nonsense	protocol	spider
	verb	revenge	ridicule	tractor
	zeal	tenure	upheaval	whistle
Syllable numerosity				
M	1	2	3	
Phoneme numerosity				
M	3.2	7.1	9.6	
SD	0.6	1.0	1.4	
Onset dissimilarity				
M	0.376	0.375	0.381	
SD	0.101	0.095	0.076	
Nucleus dissimilarity				
M	0.249	0.196	0.221	
SD	0.123	0.080	0.068	
Coda dissimilarity				
M	0.275	0.284	0.222	
SD	0.105	0.090	0.118	
Familiarity				
M	503.7	434.9	481.2	
SD	64.2	170.3	88.1	
Concreteness				
M	298.3	259.8	296.2	
SD	38.0	96.4	27.2	
Written frequency				
M	27.5	23.3	19.0	
SD	36.4	21.7	25.3	

4.1.4 Design

Participants were tested individually with three procedures: an articulatory duration (AD) measurement, immediate serial recall (ISR), and cumulative forward rehearsal (CFR). Testing occurred in two sessions of 1 to 1.5 hours on separate days. In one session, ISR with the practice word set was followed by AD and ISR with Word Sets 1 and 3. In the other session, CFR with the practice word set was followed by AD and CFR for each of the experimental sets. AD always immediately preceded ISR and CFR for each experimental word set. The word sets were used in the same order in both sessions. The order of ISR and CFR across sessions and the order of short and long word sets within sessions were counterbalanced across participants.

4.1.5 Procedures

4.1.5.1 Articulatory Duration Measurement

Articulatory durations were measured using the procedure of Mueller et al. (2003). In each session, the articulatory duration measurements for each word set were performed in a single block of 12 trials. Four trials were performed with each of the three list lengths of 2, 3, and 4 words. The order of trials in the block was selected randomly, and the subset of words used on each trial was selected randomly without replacement from the word set.

On each trial a sequence of words was visually presented on the computer screen. The participant studied the words as long as desired. Once he/she verbally indicated readiness, the words were removed from the screen, and after 300 ms, three 100-ms tones were presented at 500-ms intervals. After the third tone, the participant articulated the sequence of words clearly and rapidly, twice without pause, in the order presented. The completion of the recital was recorded by the experimenter pressing a key. If the participant did not say the sequence correctly, then the trial was repeated. The articulation time was measured as the time from the end of the third tone until the experimenter depressed the key.

4.1.5.2 Immediate Serial Recall

For the immediate serial recall task, one block of 15 trials in a staircase procedure was used for each word set, following Mueller et al. (2003). The first trial of the block used a sequence of length four. The number of words used in each following trial depended on the accuracy of its preceding trial. After a correct trial, the next trial used one more word, while, after an incorrect trial, the next trial used one less word. The words for each trial were selected randomly without replacement from the word set.

Each trial began with the auditory presentation of the number of words to be recalled. After a 4-s pause, a sequence of words was presented auditorily, with 1-s inter-stimulus intervals. One second after the final word, a 100-ms tone was presented signaling the participant that he/she should respond. The participant attempted to recite the words in the same order that they were presented. The experimenter listened to the sequence and entered the spoken words into the computer as they occurred. The trial was considered correct only if the entire sequence was recalled in order. Feedback about accuracy was given by the auditory presentation of either the word “correct” or “incorrect”.

Participants were instructed to recall as many words as possible in the correct order. To provide incentive, a point scheme was used, and participants were paid a bonus based on the number of points they collected. Participants received one point for each word recalled in the correct position, plus ten times their memory span in additional points at the end of each block. They were paid a bonus of one dollar for every 100 points earned.

4.1.5.3 Cumulative Forward Rehearsal

There were four consecutive blocks of the CFR task for each word set. The initial number of words presented on each trial stayed constant throughout a block. Trials in the first block started with two words, in the second with three, in the third with four, and in the fourth with five. Counterbalancing the order of startup lengths was not feasible. An ascending order was chosen to gradually ramp up the difficulty of the trials across blocks. A block ended after a trial when a total of 40 subtrials or 10 trials were completed, whichever came first.

The words in the startup set were selected randomly without replacement from the word set. The word to present on each subtrial was selected randomly from those words in the word set not in the current sequence to be recalled. Thus, no word ever occurred more than once in a single recall sequence.

At the beginning of each trial, a message was visually presented indicating the number of words in the startup sequence. One second into this presentation, the number of words was also presented auditorily. After the visual message was presented for 3 s, it was removed and replaced with a 1.3-s fixation. The trial then continued with a startup period followed by zero or more subtrials (see Figure 3.1) followed by feedback.

In the startup period, the initial sequence of words was auditorily presented with 100-ms inter-stimulus intervals. Then, 100 ms after the last word of the startup sequence, a 100-ms tone was played to cue recall. The participants tried to recall the startup sequence. They were instructed to begin rehearsal as soon as they could and to recite quickly and accurately. The start time of their articulation was recorded by a voice key, and the finish time and accuracy were recorded by the experimenter pressing one of two keys. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played, and the first subtrial began after a 300-ms response-stimulus interval.

On each subtrial, a word was presented auditorily, and the subject had to respond as soon as possible by quickly and accurately reciting an updated sequence consisting of the previous sequence followed by the newly presented word. As during the startup period, reaction time, articulation time, and accuracy were recorded. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played, and the next subtrial began after another 300-ms RSI.

A trial ended after an error was made, or after a subtrial whose correct sequence contained ten words. Visual feedback was then provided for 4 s, informing the participant of the correct response, the number of subtrials completed, average reaction time, average articulation time per word, and the number of points earned on the trial.

To encourage effortful performance, participants earned points based on reaction time, articulation time and accuracy. A monetary bonus was paid based on the number of points earned. On each subtrial, the maximum points that could be earned were the number of words in the correct response multiplied by 5000. No points were earned for a subtrial if an error was made. For a correct subtrial, one point was subtracted from the maximum for each millisecond of articulation time and reaction time. Participants were paid a bonus of one dollar for every 1,500,000 points earned.

4.1.6 Data Analysis

4.1.6.1 Articulatory Duration

Data from the articulatory duration measurements were analyzed using the method specified by Mueller et al. (2003). Articulation time for each trial was predicted using the exponential function

$$t_{ijk,\text{predicted}} = I_i + 2n_{ijk} \cdot d_{ij} \cdot a_{ij}^{n-2}. \qquad \textbf{Equation 4.1}$$

Participants are indexed by *i*, word sets by *j*, and trials by *k*. *I* is an intercept parameter estimated for each participant, *n* is the number of words in the sequence, *d* is the base word length estimated for each combination of word set and participant, and *a* is an amplification factor estimated for each combination of word set and participant. The parameter values for *I*, *d*, and *a* were estimated by finding the values that minimized the sum of squared error

$$\sum_{i,j,k} \left(\frac{t_{ijk,\text{observed}} - t_{ijk,\text{predicted}}}{n_{ijk}} \right)^2. \qquad \textbf{Equation 4.2}$$

After the initial minimization, outlier trials for which the value $(t_{ijk,\text{observed}} - t_{ijk,\text{predicted}})/n_{ijk}$ was more than 2.5 standard deviations from the mean for each subject *i* were removed and the minimization was recalculated. This process was repeated iteratively until no more outlier trials are found. Typically, 2% to 5% of trials were omitted from the final fit through this process.

Once the parameter values were found, a per participant, per word-set estimate of duration was calculated as

$$D_{ij} = \sum_{n=2}^{5} \frac{d_{ij} \cdot a_{ij}^{n-2}}{4}. \qquad \textbf{Equation 4.3}$$

Mean fits for each word set were calculated by taking the arithmetic mean of each participant's value for *I*, and the geometric mean of values for each participant and word set for *d* and *a*. Finally, arithmetic means of the D_{ij} values for each word set were taken across participants to find estimated mean articulatory durations for each word set. As discussed in Mueller et al. (2003), this procedure and analysis provide a measure of articulatory duration that accurately reflects the rehearsal conditions found in serial recall experiments. Furthermore, it removes timing biases due to the equipment and the experimenter from the final estimates.

4.1.6.2 Reaction Time

Reaction times were fit with straight lines as a function of sequence length where appropriate. This was done by first fitting the individual participants' data and then averaging the slopes and intercepts to obtain mean values. The individual participant fits were done using an iterative method of fitting and outlier removal. Typically, 2% to 10% of data points were omitted from the final fit through this process.

4.1.6.3 Articulation Time

Articulation times were fit with a method similar to the one described previously for articulatory durations. Articulation time for each trial was predicted using the exponential function

$$t_{ijk,\text{predicted}} = I_i + n_{ijk} \cdot d_{ij} \cdot a_{ij}^{n-2}. \qquad \textbf{Equation 4.4}$$

Participants are indexed by *i*, word sets by *j*, and trials by *k*. *I* is an intercept parameter estimated for each participant, *n* is the number of words in the sequence, *d* is the base word length estimated for each combination of word set and participant, and *a* is an amplification factor also estimated for each combination of word set and participant. The parameter values for *I*, *d*, and *a* were estimated by finding the values that minimized the sum of squared error

$$\sum_{i,j,k} \left(\frac{t_{ijk,\text{observed}} - t_{ijk,\text{predicted}}}{n_{ijk}} \right)^2 . \qquad \textbf{Equation 4.5}$$

After the initial minimization, outlier trials for which the value $(t_{ijk,\text{observed}} - t_{ijk,\text{predicted}})/n_{ijk}$ was more than 2.5 standard deviations away from the mean for each subject *i* were removed and the minimization was recalculated. This process was repeated iteratively until no more outlier trials are found. Typically, 2% to 10% of data points were omitted from the final fit through this process.

Mean fits for each word set were calculated by taking the arithmetic mean of each participant's value for *I*, and the geometric mean of values for each participant and word set for *d* and *a*.

4.1.6.4 Memory Span

Memory span is a standard measure for ISR of the sequence length at which a participant has a 0.5 probability of correct recall (Mueller et al., 2003). For both ISR and CFR, memory span was calculated for each participant for each word set. It was calculated by performing a linear regression on the logit of the probability of correct recall as a function of sequence length, which is equivalent to performing a logistic regression on the probability. In other words, the logit of the probability of correct recall, for a given participant *i*, for word set *j*, for sequence length *n*, was predicted as

$$\operatorname{logit}(p_{ijn}) = \ln\left(\frac{p_{ijn}}{1 - p_{ijn}} \right) = a_{ij} + b_{ij} n. \qquad \textbf{Equation 4.6}$$

After the best fitting values for parameters *a* and *b* were found, the memory span was calculated as the sequence length, *n,* for which the probability of recall, *p,* was 0.5,

$$MSP_{ij} = \frac{\ln\left(\frac{0.5}{1-0.5}\right) - a_{ij}}{b_{ij}} = -\frac{a_{ij}}{b_{ij}}. \qquad \textbf{Equation 4.7}$$

Mean memory spans for each word set were then calculated by taking the arithmetic mean across participants. Mean fits for each word set were calculated using the mean memory span and the harmonic mean of the *b* values.

4.1.6.5 Serial Position Functions

While memory span provides a gross measure of recall accuracy, at the other end of the spectrum are various approaches to analyzing participants' responses on the word by word level. These can provide great insight into the types of errors participants make, and thus, by inference, the processes that underlie performance. Three classes of analysis of this type are serial position functions (SPFs), response-type functions (RTFs), and position gradient functions (PGFs). Their use here is adapted to the rehearsal tasks based on their use for ISR in Mueller (2002).

In analyses of ISR, serial position functions plot the probability of correct recall for words as a function of their presented serial position and list length. Here they are generalized to plot probability of correct recall for words as a function of their *correct recall position* and list length. Note that the relationship between order of presentation and correct recall position varies between the different rehearsal tasks.

Three different SPFs are used that differ in how correct recall is scored:

For 'position' SPFs, a word is scored as correct only if it is recalled in the correct position, otherwise it is incorrect. This is the strictest form of scoring and emphasizes order and item information.

For 'item' SPFs, a word is scored as correct if it is recalled anywhere in the sequence and incorrect if it is not recalled at all. This treats the trial as free recall instead of serial recall, and emphasizes the recall of items and not order information.

Finally, for 'relative order' SPFs, words are only scored if they are recalled. A recalled word is scored as correct if the word, if any, recalled immediately before it belongs somewhere before it in the sequence, and the word, if any, recalled immediately after it belongs somewhere after it in the sequence. If the word is recalled but not in the right relative order it is scored as incorrect. This emphasizes relative order information, since there is no penalty for words that are not recalled, but only for misorderings.

4.1.6.6 Response-Type Functions

Response-type functions are built by classifying each response into one of several categories (i.e., response types). They plot proportion of responses of each type as a function of the serial position of the response and list length. The response-type categories are the following:

1. 'No Error' responses where the correct word was said in the given position.
2. 'List Error' responses where an incorrect word was said that should have been said elsewhere in the current recall list.
3. 'Word set Error' responses where an incorrect word was said that should not have been said at all, but the word was in the current word set.

4. ‘Other Response’ responses where an incorrect word was said that does not belong to the current word set.
5. ‘No Response’ responses where nothing was said at an expected response position.

Response-type functions primarily provide information about the origin of words said in a sequence. They can provide insight into the processes that are determining what words to say during recall of a sequence.

4.2 Results

Results are first presented for each of the three tasks, and then comparisons between performance on ISR and CFR are presented.

4.2.1 Meta-Comments on Graphs

Error bars indicating 95% confidence intervals are shown for the observed data in all graphs of accuracy, reaction time, and articulation time throughout the dissertation. These were calculated by first finding the standard error per participant for each data point and then finding the standard error of the mean. These error bars are not representative of statistical reliability for particular tests of differences between conditions.

A consistent scheme of line and point color, point shape, point fill, and line type is used throughout the dissertation, as outlined in Table 4.2. These visual properties are used to differentiate the data based on task, word set, observed versus predicted, and mathematical fit. Color, which is always consistent for points and their accompanying line, is used to indicate the task. For the rehearsal tasks, hue indicates cumulative versus rolling, and saturation indicates forward versus backward, providing easy perceptual grouping along both dimensions. The shape of the points is used to indicate the word set, with a distinct shape reserved for data averaged across word sets. Filled points are used for observed data, and open points are used for model predictions. Connecting lines for the observed data are solid, while connecting lines for the model predictions are dashed. Mathematical fits to the observed data are drawn as smooth curves using dotted lines, while mathematical fits to the model predictions are drawn as smooth curves using dot-dashed lines.

Table 4.2 Formatting scheme for graphical presentation of data.

Category represented	Graphical element
Task	Color
AD	🟪
ISR	🟩
CFR	🟦
RFR	🟥
CBR	🟦
RBR	🟥
Word set	Point shape
1	●
2	■
3	▲
Average	✳
Source	Point fill and line type
Observed data	—●—
Mathematical fit	·····●·····
Model prediction	– – ○– – ·

4.2.2 Articulatory duration

Articulatory duration measurements for each word set for each participant were aggregated across the two sessions. Figure 4.1 shows the observed mean articulation times for each word set as a function of sequence length,[1] as well as the exponential functions fit to the data to calculate the articulatory durations. The mean estimated articulatory durations for each word set are shown in Table 4.3. I found that articulatory durations for Word Set 1 were reliably shorter than those for Word Set 3, $t(7) = 22.68$, $p < .001$. This is useful because it establishes that the word sets differ in articulatory duration, and so they should lead to a word length effect in tasks that use the phonological loop.

Table 4.3 Experiment 1: Articulatory durations and memory spans per word set.

	Word set	
Task and variable	1 (short)	3 (long)
AD		
Duration (ms)	261	424
ISR		
Memory span (words)	6.07	5.55
CFR		
Memory span (words)	7.25	6.02

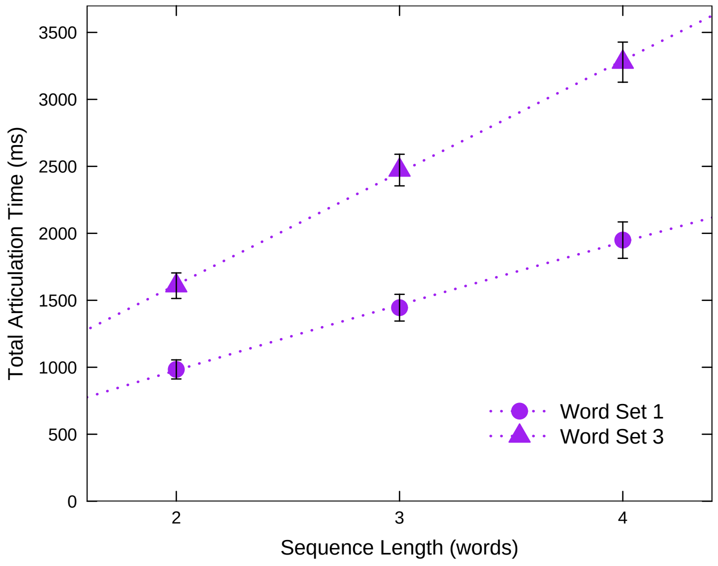


Figure 4.1 Experiment 1 AD: Observed mean articulation times and mean exponential fits (dotted lines) as a function of word set and sequence length.

[1] Observed mean articulation times for each word set for each sequence length are calculated by first subtracting each participant's estimated intercept value, *I*, from their mean articulation times. The geometric mean of the resulting values is then taken for each word set at each length. Finally, these values are added to the arithmetic means of the intercept values.

4.2.3 ISR

Figure 4.2 shows the observed mean probability of correctly recalling the entire presented sequence as a function of word set and sequence length in the ISR task.[2] It also shows the mean best fitting logistic regression functions used to calculate memory span. Table 4.3 lists the mean estimated memory spans for each word set obtained from the fit functions. As expected based on past findings of the word length effect, I found that memory span for the shorter words of Word Set 1 was reliably greater than memory span for the longer words of Word Set 3, $t(7) = 2.01$, $p = 0.042$ (two-tailed).

The patterns of errors made in the ISR task are typical of this task. Consequently, presentation of the error analysis in the form of SPFs and RTFs is postponed until the later section dealing with modeling of the ISR results.

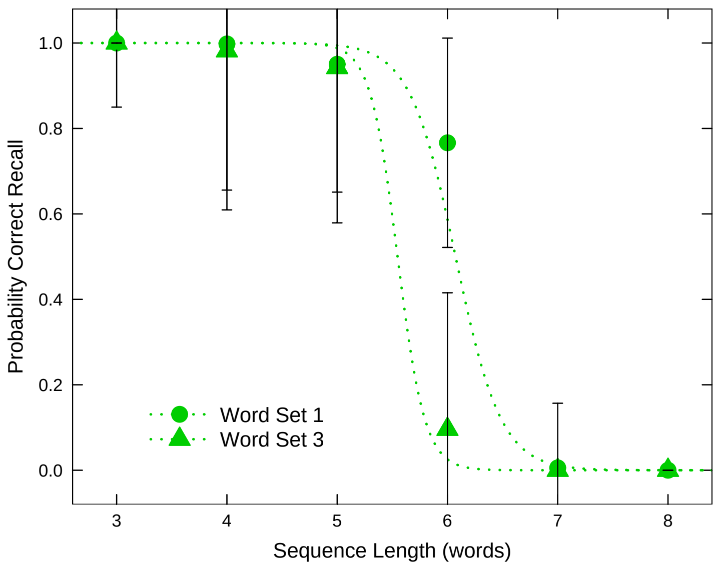


Figure 4.2 Experiment 1 ISR: Observed mean probabilities of correct recall of the entire sequence and estimated mean logistic regression functions (dotted lines) as a function of word set and sequence length.

[2] Observed mean probabilities of correct recall are calculated by taking the arithmetic mean of individual participants' probabilities of correct recall for each word set and sequence length.

4.2.4 CFR

For the CFR task, data were collected from two distinct parts of each trial. First there was the startup period, where participants recalled the initial sequence of two to five words. Then there was the series of subtrials where they had to update the previous sequence by adding a newly presented word to the end. Since this is the first use of a rehearsal task, basic results for both the startup period and the subtrials will be presented.

Figure 4.3 shows the observed mean probability of correct recall during the startup period of trials as a function of word set and sequence length. In a two-way within-subjects ANOVA for the probability of correct recall, the main effects of word set, $F(1, 7) = 16.87$, $p = 0.004$, and sequence length, $F(3, 21) = 31.26$, $p < .001$, and their interaction, $F(3, 21) = 18.43$, $p < .001$, were all reliable. These effects were driven largely by the decrease in accuracy at sequence length five, which is considerably larger for the longer words of Word Set 3.

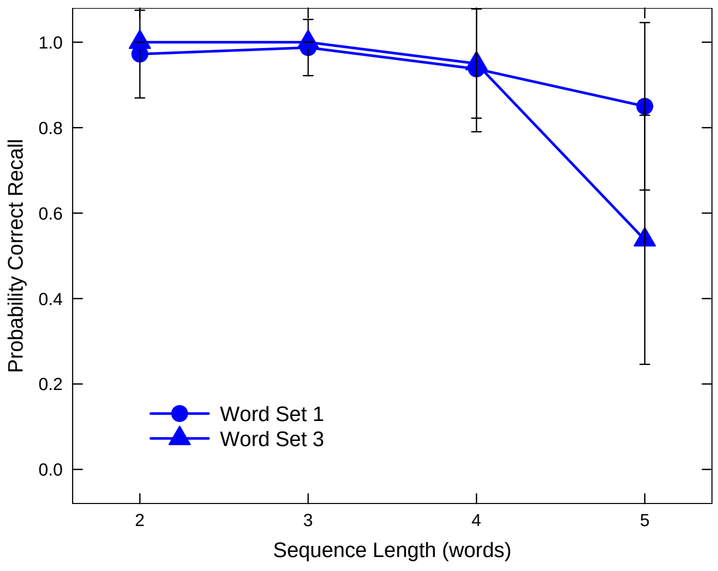


Figure 4.3 Experiment 1 CFR startup period: Observed mean probabilities of correct recall of the entire sequence as a function of word set and sequence length.

Subtrials could be grouped both by the number of words to be recalled and by the number of words in the startup sequence for that trial. In other words, a subtrial in which six words were recalled could be the first in a trial with a startup sequence of five, or the fourth in a trial with a startup sequence of two. However, in a three-way within-subjects ANOVA for probability of correct recall, with word set, sequence length, and startup sequence length as factors, the effect of startup sequence length was not reliable, $F(3, 21) = 0.186$, $p = .90$.[3] As a result, in subsequent analyses the data have been collapsed across startup sequence length.

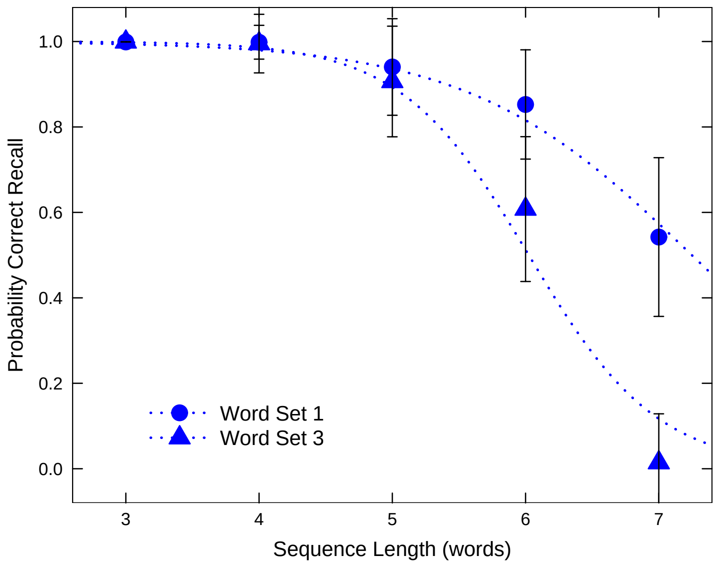


Figure 4.4 Experiment 1 CFR subtrial: Observed mean probabilities of correct recall of the entire sequence and estimated mean logistic regression functions (dotted lines) as a function of word set and sequence length.[4]

Figure 4.4 shows the observed mean probability of correctly recalling the entire updated sequence as a function of word set and sequence length during the subtrials of

[3] Only sequences lengths of six and seven were included in this analysis, since at shorter lengths not all startup lengths were represented, and at longer lengths there was a paucity of data. However, an inspection of the data at lengths four and five showed a consistent pattern.

[4] A small amount of data was available for sequence lengths of eight or more, but since few participants contributed to it, and very few subtrials occurred, it is not shown here since it is not considered representative.

the CFR task. It also shows the mean best fitting logistic regression functions used to calculate memory span. In a two-way within-subjects ANOVA for the probability of correct recall, the main effects of word set, *F*(1, 7) = 25.43, *p* = 0.001, and sequence length, *F*(4, 28) = 113.88, *p* < .001, and their interaction, *F*(4, 28) = 18.02, *p* < .001, were all reliable.[5] Table 4.3 lists the mean estimated memory spans for each word set obtained from the fit functions. I found that memory span for the shorter words of Word Set 1 was reliably greater than memory span for the longer words of Word Set 3, *t*(7) = 6.17, *p* < 0.001.

The reliable effect of word length on accuracy for CFR supports the assertion that CFR uses the phonological loop as does ISR. A word length effect is typically found in ISR (Mueller et al., 2003), and was replicated here. For ISR, this effect has been interpreted as evidence for the role of the phonological loop in memory maintenance. The longer it takes to rehearse a sequence of words, the more likely that information will decay before it gets refreshed by the rehearsal process. The evidence here supports the conclusion that this same process is occurring in CFR.

During each startup period in the CFR task, reaction time (RT) was measured as the time between the onset of the recall cue and the triggering of the voice key by the participant's vocal response. Figure 4.5 shows the observed mean startup period RT as a function of word set and sequence length.[6] Using a two-way within-subjects ANOVA, I found that the main effect of sequence length was reliable, *F*(3, 21) = 10.14, *p* < 0.001, but the main effect of word set, *F*(1, 7) = 0.50, *p* = 0.50, and the interaction, *F*(3, 21) = 0.50, *p* = 0.68, were not.

[5] Only sequence lengths of three through seven were used in this analysis, because not all participants were able to reach sequences of length eight or more.

[6] Observed mean reaction times and articulation times were calculated by taking the mean of participants' median times on correct recalls for each word set and sequence length. Medians were used for individual participants to mitigate the effect of outliers.

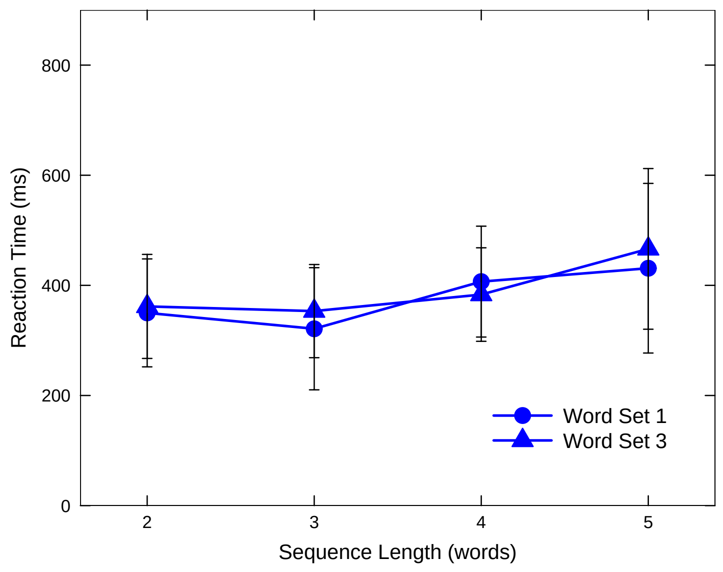


Figure 4.5 Experiment 1 CFR startup period: Observed mean reaction time as a function of word set and sequence length.

On subtrials in the CFR task, RT was measured as the time between the onset of the presented word and the triggering of the voice key by the participant's response. Note that the use of the word onset as the starting point is not without its problems, since the contents of the word are not yet available to the participant at that time. However, since I do not know in this context how much of the word must be presented to the participants before they are able to perceive it, the onset provides a well-defined, unbiased starting point. At a later point, the relationship between differences in presented word length and RT as a function of word set will be taken up.

Figure 4.6 shows the observed mean subtrial RTs on correct trials as a function of word set and sequence length.[7] Using a two-way within-subjects ANOVA, I found a reliable difference between word sets, $F(1, 7) = 32.04$, $p < .001$, but the effect of sequence length, $F(3, 21) = 1.18$, $p = 0.34$, and the interaction, $F(3, 21) = 0.73$, $p = 0.55$, were not reliable.

[7] Only data for sequence lengths three through six were used in subtrial RT and AT analysis, because some participants had no correct recalls at longer lengths.

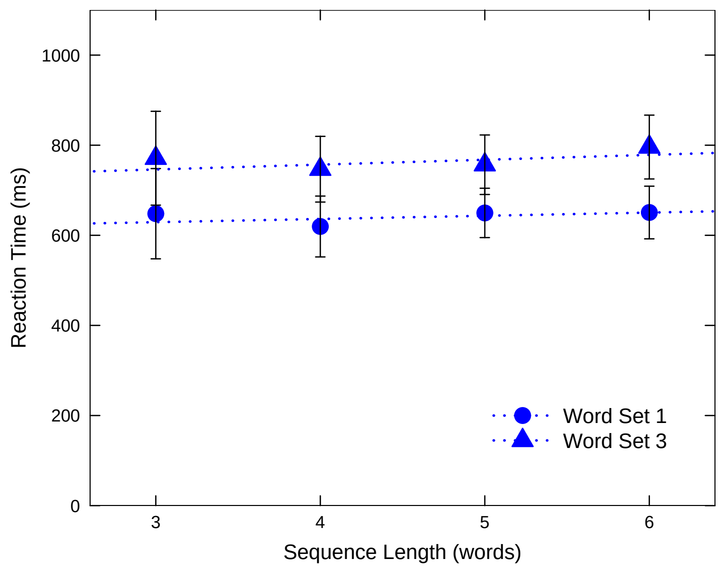


Figure 4.6 Experiment 1 CFR subtrial: Observed mean reaction time as a function of word set and sequence length with linear fits (dotted lines).

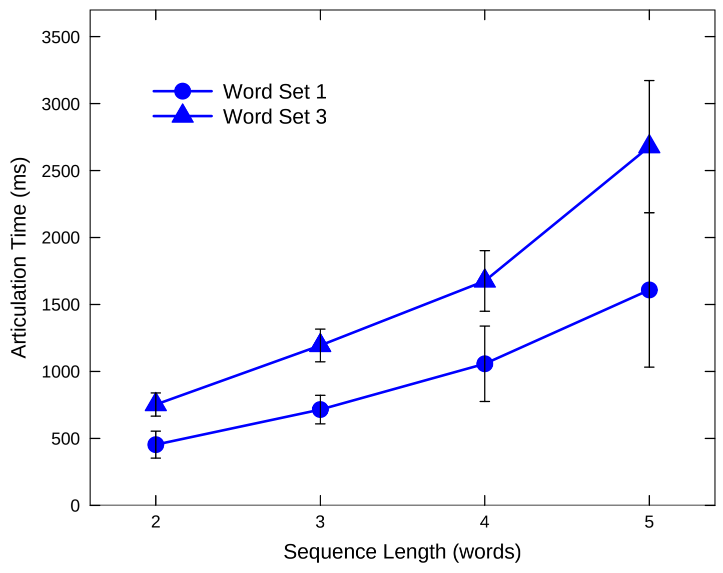


Figure 4.7 Experiment 1 CFR startup period: Observed mean articulation time as a function of word set and sequence length.

In addition to RT, articulation time (AT) was also measured. For both startup periods and subtrials, AT was measured as the time from when the voice key triggered at the beginning of recall until the experimenter pressed a key indicating that recall was complete. Figure 4.7 shows the observed mean startup period ATs on correct trials as a function of word set and sequence length. Using a two-way within-subjects ANOVA, I found reliable effects of word set, $F(1, 7) = 157.5$, $p < .001$, and sequence length, $F(3, 21) = 222.9$, $p < .001$, as well as a reliable interaction, $F(3, 21) = 40.5$, $p < .001$. Looking at individual contrasts for the effect of sequence length, I found reliable linear, $F(1, 7) = 643.4$, $p < .001$, and quadratic, $F(1, 7) = 23.0$, $p < .001$, trends.

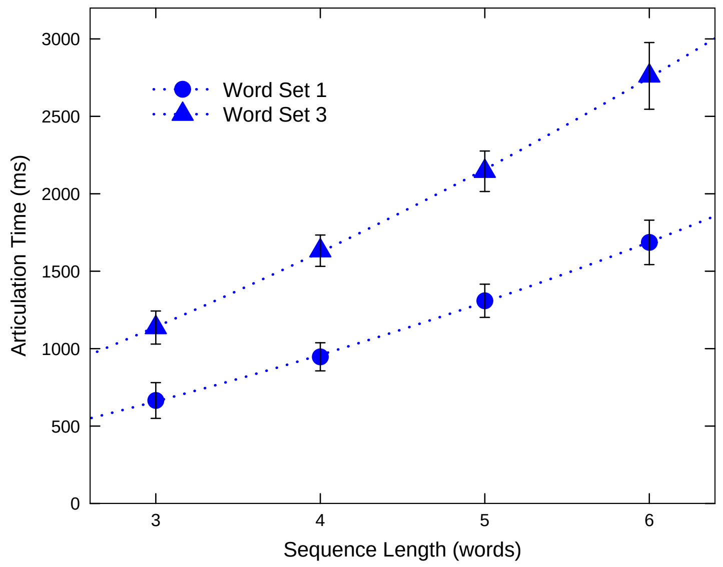


Figure 4.8 Experiment 1 CFR subtrial: Observed mean articulation time as a function of word set and sequence length with mean exponential fits (dotted lines).

Figure 4.8 shows the observed mean subtrial ATs on correct trials as a function of word set and sequence length. Using a two-way within-subjects ANOVA, I found reliable effects of word set, $F(1, 7) = 259.5$, $p < .001$, and sequence length, $F(3, 21) = 404.0$, $p < .001$, as well as a reliable interaction, $F(3, 21) = 35.2$, $p < .001$. Looking at individual contrasts for the effect of sequence length, I found reliable linear, $F(1, 7) = 1204$, $p < .001$, and quadratic, $F(1, 7) = 82.9$, $p = .009$, trends.

The presentation of error analysis in the form of SPFs, RTFs, and PGFs is deferred until the later section dealing with modeling of the CFR results.

4.2.5 ISR versus CFR

The prior analyses have assessed ISR and CFR separately, but for the purposes of assessing shared mental mechanisms, comparison of performance on the two tasks is important. Figure 4.9 shows the relationship between memory span for ISR and memory span for CFR averaged across word sets for each participant. The correlation between the two measures was found to be large and reliable, $r(6) = 0.81$, $p = .01$. A regression analysis found that the memory span for CFR can be expressed as a linear function of memory span for ISR with an intercept of 3.62 and a slope of 0.52.

This high correlation for individual participants between memory span for ISR and memory span for CFR provides further support for the claim that CFR uses the phonological loop as ISR does. This type of correlation has been used to implicate the same underlying working-memory system in a variety of situations including various tests of general intelligence that have also correlated highly with memory span in ISR.

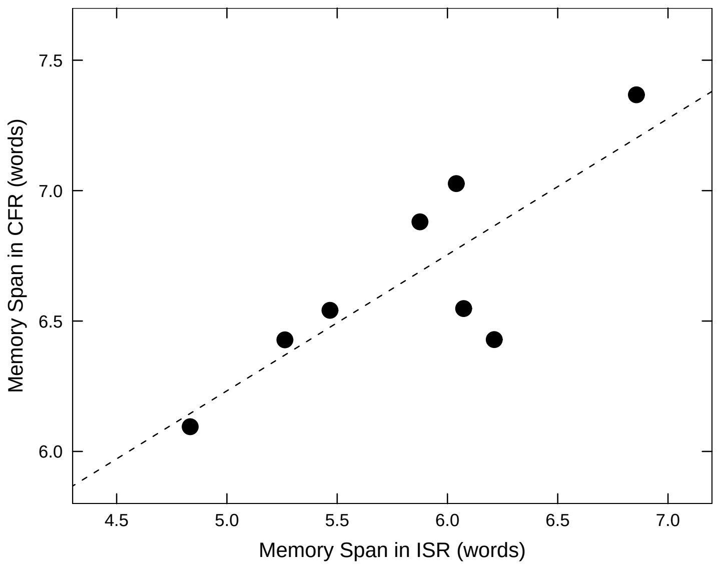


Figure 4.9 Experiment 1 ISR and CFR: Per participant mean memory span across word sets for CFR plotted against ISR (circles), with fitted regression line (dashed line). The regression line slope is 0.52 and intercept is 3.62. The correlation between ISR span and CFR span is 0.81.

4.3 Modeling of CFR

A model of CFR was developed that embodies an initial theory of working-memory updating performance. This model and theory were then generalized to other updating situations in later chapters. The effort to model the CFR task has a firm foundation because a number of systematic and interesting results were found. While accuracy decreased monotonically as a logistic function of sequence length, reaction time did not change reliably as a function of sequence length. This is significant because it provides evidence that the time duration of the cognitive processes required to support this type of updating is not dependent on the number of items in memory.

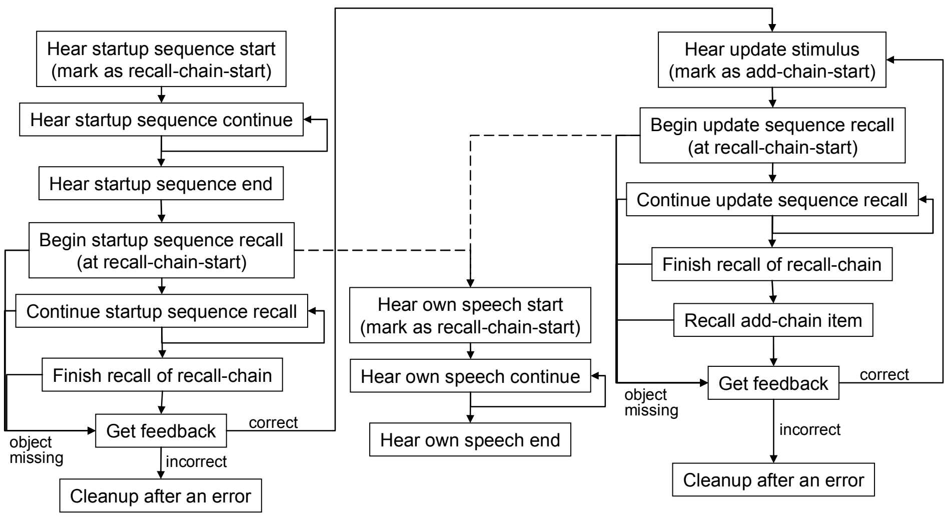
The use of shared mechanisms in ISR and CFR is explored further in the modeling work reported in the next few sections as well. For example, the correlation data just discussed is analyzed in more detail by considering the slope of the relationship. If the operation of a shared mechanism in the different task contexts can account for the slope, this will provide further support for shared mechanisms. For this reason, ISR is modeled as well as CFR.

4.3.1 Basic Model

The initial minimal Basic Model of CFR is based on the 'Boulder' model of ISR (Kieras et al., 1998; Kieras et al., 1999). This model uses a simple articulation strategy to perform the task. Under the model, decay only occurs for speech objects as a whole. When a speech object decays, both the phonological content and the order information are lost. When the model can not find the next speech object during articulation of a sequence, it gives up without trying to recover. Due to the task design and task demands, all rehearsal is overt. This model consists of a number of methods that implement the main sub-tasks required for performance. The main sub-tasks of the model are:

1. Encode startup sequence as recall-chain
2. Recall the startup sequence
3. Encode own articulation as new recall-chain
4. Get feedback on accuracy of performance
5. Encode subtrial stimulus as add-item
6. Recall current recall-chain followed by add-item
7. Clean-up at trial end

Including housekeeping rules for starting a block of trials and cleaning up after a trial, there are about 40 rules in this model. The sequencing of sub-tasks is represented schematically in Figure 4.10. As the startup sequence of words is presented, the model listens to the sequence, tagging each new word as having been heard. A speech tag pointing to the first word in the sequence is created, labeling it as the recall-chain. Upon hearing the initial recall cue, the model walks down the recall-chain, retrieving the content of each speech object and overtly articulating each word in turn until the end of the chain is reached. As this is occurring, the model is also hearing itself recite the sequence and again tagging each word as it is perceived. This new sequence is tagged as the new recall-chain. The auditory perceptual processor and the vocal motor processor are operating in parallel. In addition, methods consisting of collections of rules are managing the processes of articulation and perception in parallel. If at any time in the rehearsal process the model is unable to find the next speech object because of decay, it stops rehearsal. Once the model finishes rehearsing, either because the entire sequence has been rehearsed, or because a missing object was encountered, it waits for feedback. If feedback indicates an error, then the trial is over, and the model cleans up and waits for the next trial to begin. If the feedback indicates correct recall, then the sequence of subtrials begins.



Hear startup sequence start (mark as recall-chain-start)
Hear startup sequence continue
Hear startup sequence end
Begin startup sequence recall (at recall-chain-start)
Continue startup sequence recall
Finish recall of recall-chain
Get feedback
correct
object missing
incorrect
Cleanup after an error
Hear own speech start (mark as recall-chain-start)
Hear own speech continue
Hear own speech end
Hear update stimulus (mark as add-chain-start)
Begin update sequence recall (at recall-chain-start)
Continue update sequence recall
Finish recall of recall-chain
Recall add-chain item
Get feedback
correct
object missing
incorrect
Cleanup after an error

Figure 4.10 A schematic showing important steps in the initial CFR model. The solid lines indicate flow of control within a thread of execution, while the dotted lines indicate the spawning of a separate thread of execution.

When the model hears the new word on a subtrial, it tags this as the add-item and immediately begins to recite the current recall-chain. When the end of the chain is reached, the add-item is recited. As during startup, if the next item is not available during recall, the rehearsal stops and the model waits for feedback. Crucially, the model is again hearing itself recite the sequence and building up an internal representation of it for the new recall-chain. This new internal representation will have the add-item included at the end. In other words, this new internal representation is an updated sequence that has the new word incorporated into it.

The processes of articulation and perception themselves provide an updated internal memory representation. This idea is crucial. The modified representation of the sequence is generated through a literal process of "re-presentation."

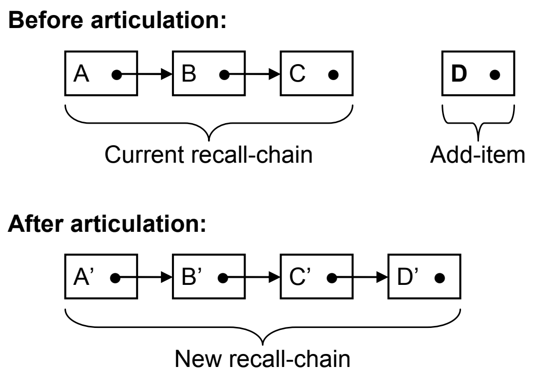


Figure 4.11 Schematic showing memory representations before and after articulation of an updated sequence. The primes (') indicate that the speech objects in the new recall-chain are different objects than those in the current recall-chain, but they do have the same phonological content.

An example of this process is represented schematically in Figure 4.11. Before articulation of the updated sequence, the recall chain contains three words, 'A,' 'B,' and 'C'. Each box represents a speech object, with a letter representing the phonological content of the speech item, and a dot with extending arrow representing the order information in the speech link. The presented word is represented independently as the add-item. The recall chain is then articulated followed immediately by articulation of the add-item. These articulations are perceived as the new recall-chain. The resulting representation is of a single integrated sequence of words. The primes (') indicate that the speech objects in the new integrated sequence are different objects from those in the original representation, but they include the same phonological content.

This model can be evaluated both for its performance during the startup period and during the subtrials within each trial. Since this is the first model presented, I will include the startup performance. However, since the primary focus of this dissertation is on the subtrials where the participant must update their memory representation, and since startup performance was similar across tasks and experiments, I will not focus on it in evaluating further models.

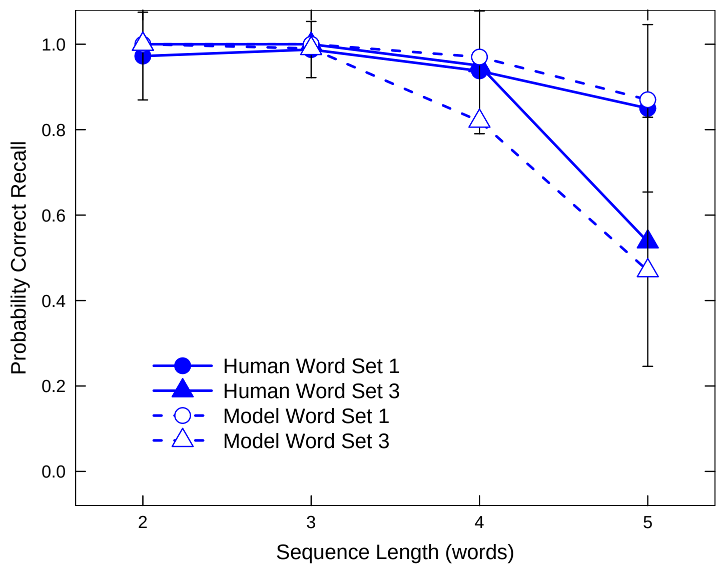


Figure 4.12 Comparison of startup accuracy for CFR Basic Model and Experiment 1 data.

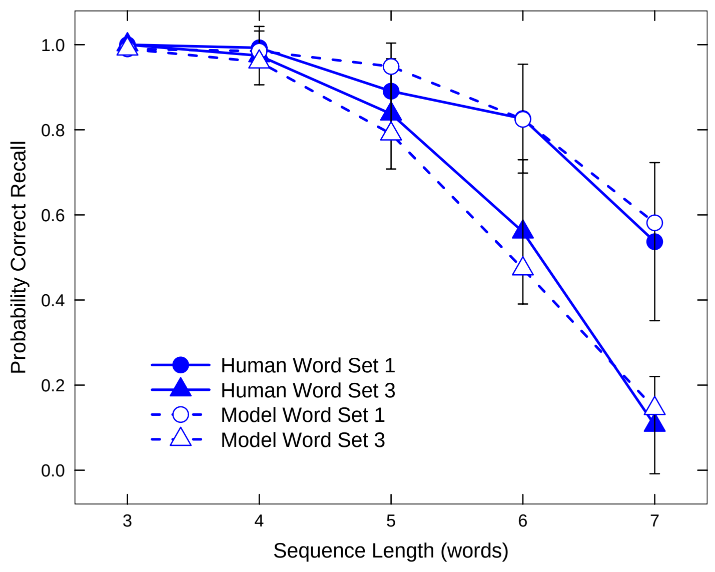


Figure 4.13 Comparison of subtrial accuracy for CFR Basic Model and Experiment 1 data.

As shown in Figure 4.12 and Figure 4.13, this model captures two key features of the accuracy data: decrease in accuracy with increasing set size, and lower accuracy for words having longer articulatory duration. These aspects of the model follow from two key features in interaction with the strategy just described. First, speech objects decay with increasing likelihood over time according to a log-normal distribution. Second, the articulatory durations of the model's speech are specified by the mean articulatory durations calculated from the empirical data in the separate articulatory duration task. The more words that need to be articulated, the more likely that one of them will decay before it is said. Furthermore, since it takes longer to articulate a given number of words having longer duration, it is more likely that one or more of the words remaining to be articulated will have decayed. Since this was the first model to be created, the decay parameters for speech objects were selected to provide a good fit to the data. In this case, the mean decay time was 4000 milliseconds and the spread of the decay-time distribution was 0.3. These values were used for both external and overt speech. Refer to Appendix A for a full listing of parameter values.

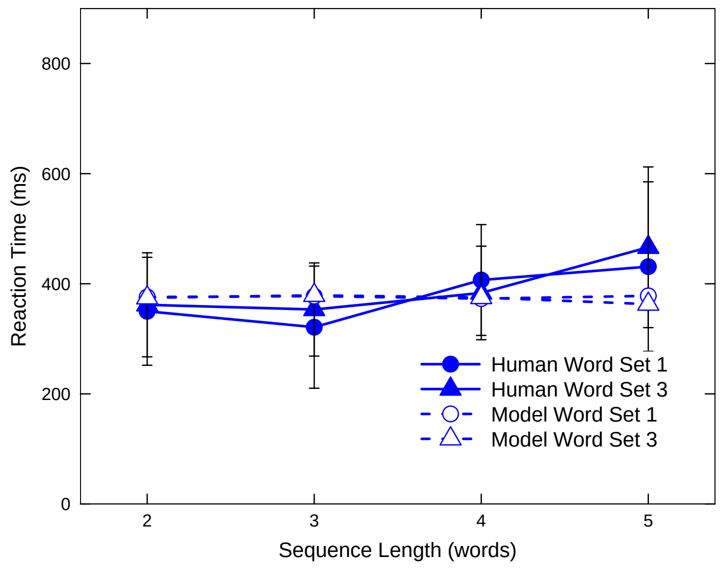


Figure 4.14 Comparison of startup reaction times for CFR Basic Model and Experiment 1 data.

This model also exhibits the essentially unchanging RTs with set size for startup and subtrials and the effect of articulatory duration on subtrial RT seen in Figure 4.14 and Figure 4.15. The lack of an articulatory duration effect on startup RT is due to the fact that recall is initiated by a presented tone which is the same regardless of word set. However, for subtrials, the model initiates the method to respond as soon as the new word is entered into working memory. The longer the duration of the word, the longer it takes to reach working memory. Since RT is measured from the onset of the word presentation, this leads, in turn, to longer reaction times.

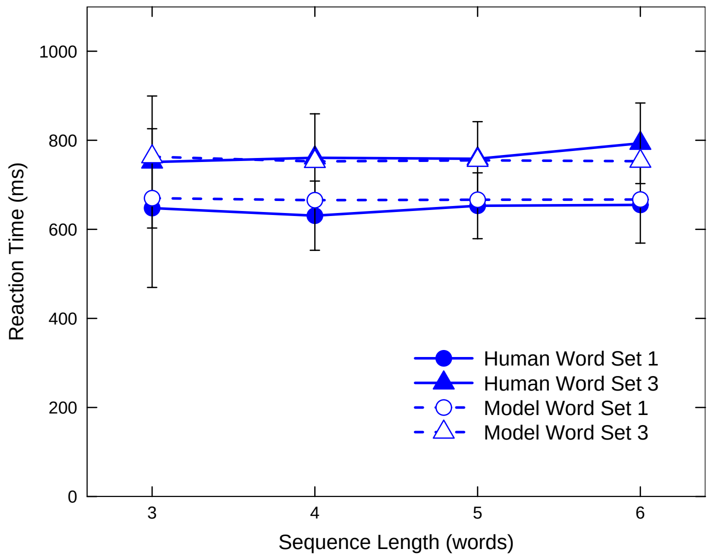


Figure 4.15 Comparison of subtrial reaction times for CFR Basic Model and Experiment 1 data.

Interestingly, the small absolute values of the reaction times, and the small difference between reaction times for the shorter and longer words, imply that a word does not need to be processed in its entirety to enter working memory. In fact, based on these data, it was determined that a content delay of fifty percent of the total word duration best accounted for performance in this model. This value was used for all subsequent models. This rapid processing is reasonable because the words came from small closed sets with which the participants had considerable practice and familiarity.

The model's successes suggest that it is capturing some critical aspects of CFR performance. However, a look at articulation times and response-type functions indicate that the model does not reflect performance in important ways. For both startup (see Figure 4.16) and subtrials (see Figure 4.17), the model does show increases in articulation times with greater sequence lengths and articulatory durations, and it accounts for their absolute values reasonably well at smaller sequence lengths, particularly for the shorter words, but it fails to exhibit the acceleration in slope found at longer sequence lengths. The model exhibits a linear increase with sequence length because the additional articulation time for each word is just the time to retrieve the phonological content, send it to the vocal motor processor, and prepare and articulate it. All of these times are constant per word with increasing sequence length. In order to show the accelerating slope of the observed data, a process would need to occur that increases in either frequency or duration per word as the sequence gets longer. Further insight into what this process might be comes from looking at the response-type functions.

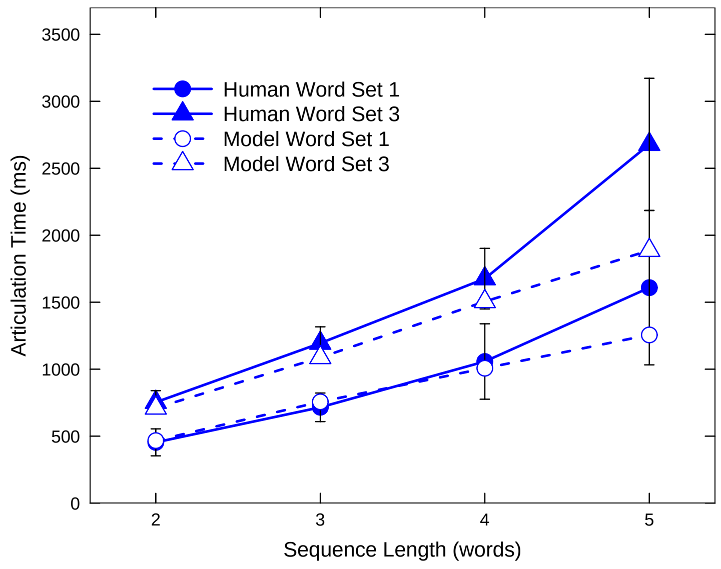


Figure 4.16 Comparison of startup articulation times for CFR Basic Model and Experiment 1 data.

The response-type functions classify each word in a response based on whether an error occurred and what type it was. They show this separately for the words at each serial position in a response, and for each sequence length. In Figure 4.18, we can see that the observed pattern of correct (i.e., 'No Error') responses is matched qualitatively by the model – the percentage of correct responses declines with serial position and with sequence length at each serial position, with a flattening out of the decline for the last few words in the sequence. However, the model predicts the occurrence of no errors where a word in the sequence is said at the incorrect position (i.e., 'List Error'), where as this is a primary type of error made by the participants. In addition, the model predicts many errors where no response is made (i.e., 'No Response') following a decelerating curve whereas the participants make far fewer of these errors, and they follow an accelerating curve with increasing recall position.

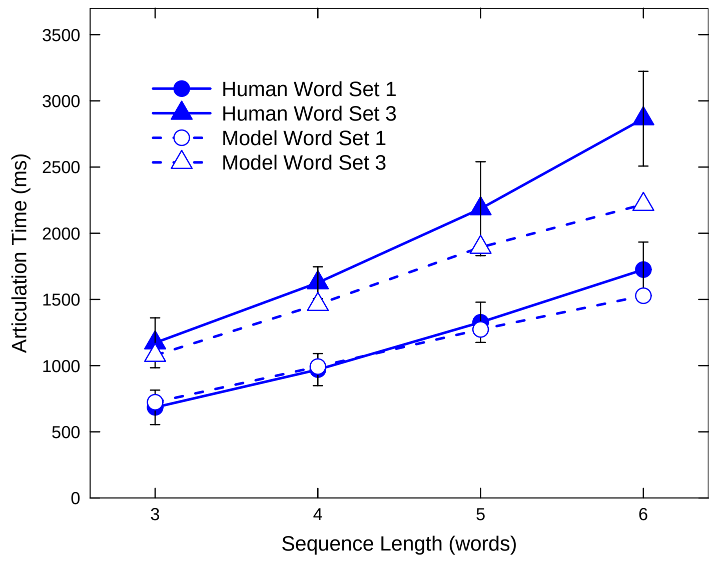


Figure 4.17 Comparison of subtrial articulation times for CFR Basic Model and Experiment 1 data.

The model shows this pattern of errors due to its lack of a recovery strategy. It will always get the sequence correct until the point at which it encounters a missing speech object, and then it will say nothing for the remaining serial positions.

The Basic Model of CFR has established that the EPIC approach to modeling verbal working memory can be extended to tasks other than immediate serial recall. The model shows the canonical effects of articulatory duration and sequence length. However, it fails to exhibit other important aspects of performance. In particular, the model fails to account for the exponential slope of articulation times, and the types of word-by-word errors that are made. A possible explanation for this is that the model lacks some of the processes that participants use in performing the task.

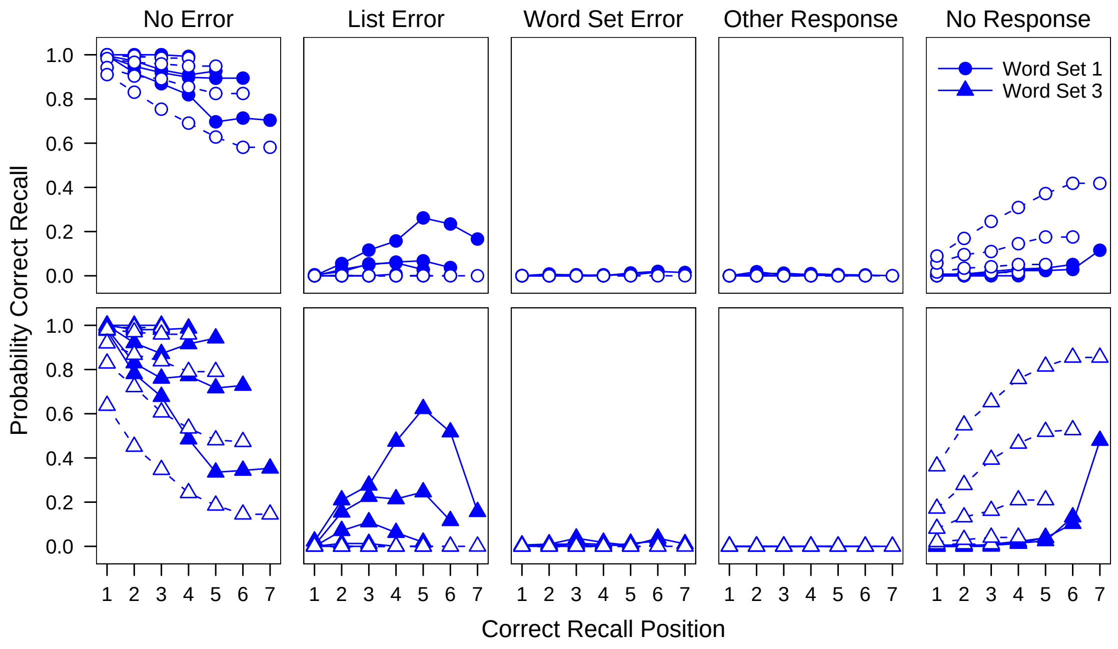


Figure 4.18 Comparison of response-type functions for CFR Basic Model (open points) and Experiment 1 data (filled points).

A reasonable hypothesis is that both of these problems have a single source in the absence of a strategy to recover when information is lost from memory. Past work has shown that people do use recovery strategies in verbal working-memory tasks such as immediate serial recall (Mueller, 2002). A particular recovery strategy is tested in my next model, a 'guessing' model of CFR performance.

4.3.2 Guessing Model

Before describing the strategy used here, it is worth noting why the implementation of such a strategy is suggested by the data. First of all, the response-type functions indicate that participants do carry on with recall beyond the point where they make an initial mistake. Whether they are aware of this or not is a separate question, but clearly the mental machinery performing the recall does not immediately abort. This is indicated by the fact that most serial-position errors are errors of position ('List error's), not errors of omission ('No response'). If this only impacted incorrect trials, perhaps it could be set aside for present purposes. However, the accelerating slopes of the articulation time functions suggest otherwise.

Since recovery would both take longer and be more likely to happen for longer sequences, this could explain the accelerating slopes. This highlights the importance of accounting for this aspect of performance. In order to get at the underlying updating processes that are occurring in this task, it is necessary to understand the time course of other processes that are contributing to task performance. This is particularly true given that the current approach argues that rehearsal and recall processes are inextricably linked to updating processes.

Mueller (2002) discusses a number of different strategy components that participants may use to recover when information is lost during recall in the immediate serial recall task. The general idea is for the participant to make an educated guess about the rest of the sequence based on the information remaining in memory. Based on a preliminary evaluation of various guessing strategies adapted to the CFR task, a particular strategy combining some of these components was developed for implementation here.

Certain differences between ISR and CFR informed this choice. In ISR, participants are sometimes encouraged to say "blank" when they do not know a word in the sequence, in order to maintain the accuracy of the following words. This is handled by a 'fill-in' process in some guessing strategies. However, this was not done in the present rehearsal tasks. In fact, participants almost never said a word that was not in the current word set, so fill-in was not used in the guessing strategy here. This was also supported by the fact that at longer sequence lengths, there were omissions at the very end of the sequence. However, as in ISR, the same word was never included twice in a recall sequence, so participants could use this knowledge by never guessing a word that they had already said in the current recall.

The guessing strategy described here is used during recall when the model checks for the next speech object to recall, and the object is not available. This could involve either the next object in the recall chain, or it could involve the add object, depending on where in recall the loss arises. Instead of aborting recall, the model goes through a series of steps to guess what to recall next. The goal is to identify all of the continuous chains of objects remaining, and then to resume recall at the start of one of these remaining chains. The chain that terminates with the last object in the original recall chain (the so-called 'end-chain') is recalled after all other chains, except for the add object, which is recalled last. A flowchart of this guessing strategy is shown in Figure 4.19.

This strategy makes use of the add object and the speech objects in the current recall chain that are yet to be recalled. They are all considered to be 'not found' when the process starts. First the add object is tagged as 'found' and the last object in the recall chain is tagged as 'found' and as the 'end-chain head'. This is done because the model knows that these two items will be recalled last, and so any other items in memory should be recalled before them. Previous work modeling ISR has indicated that specific tags pointing to these items are available (Kieras et al., 1999; Mueller, 2002).

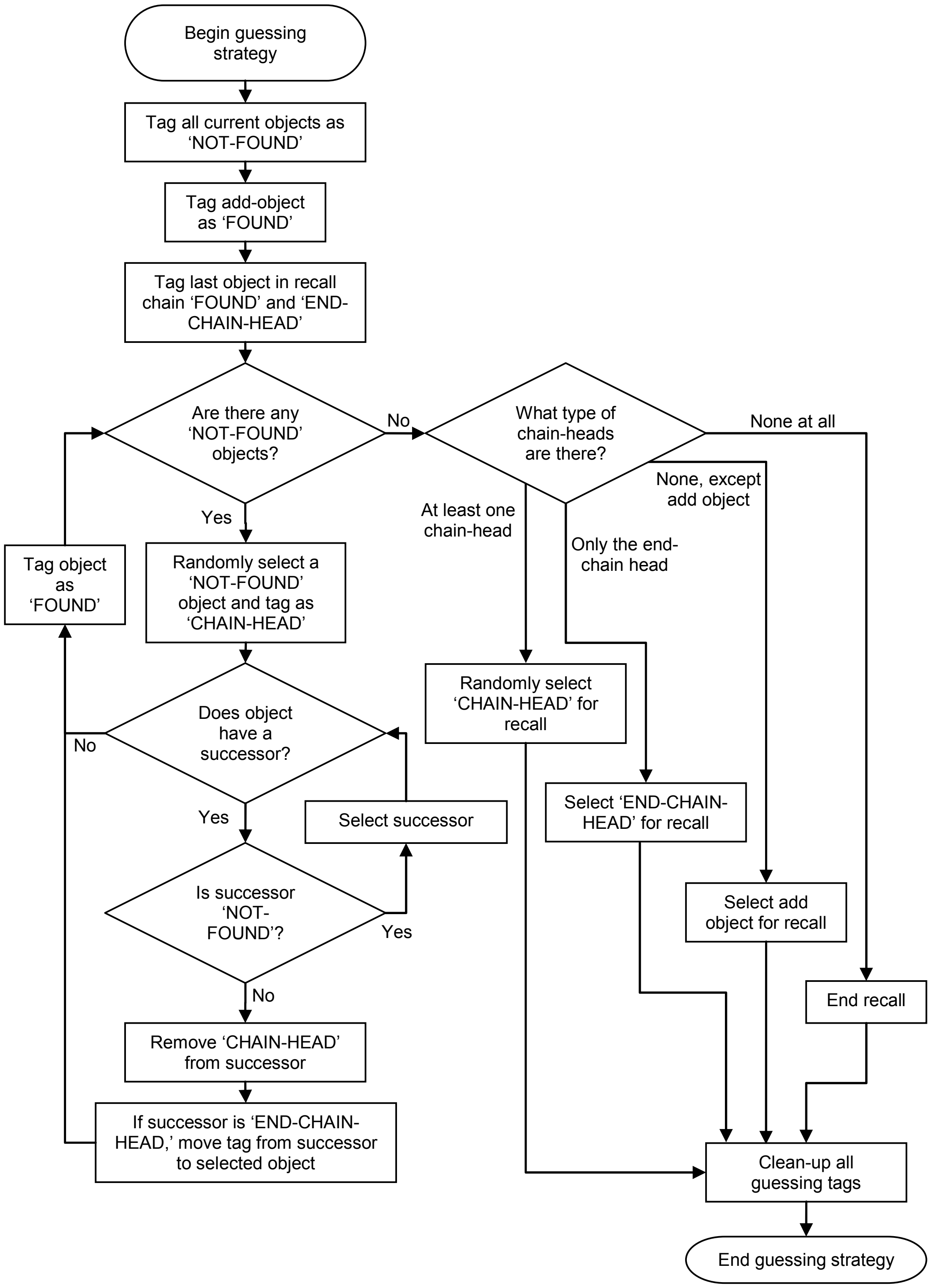

Begin guessing strategy
Tag all current objects as 'NOT-FOUND'
Tag add-object as 'FOUND'
Tag last object in recall chain 'FOUND' and 'END-CHAIN-HEAD'
Are there any 'NOT-FOUND' objects?
No
What type of chain-heads are there?
None at all
None, except add object
At least one chain-head
Only the end-chain head
Yes
Tag object as 'FOUND'
Randomly select a 'NOT-FOUND' object and tag as 'CHAIN-HEAD'
Randomly select 'CHAIN-HEAD' for recall
Does object have a successor?
No
Yes
Select successor
Select 'END-CHAIN-HEAD' for recall
Is successor 'NOT-FOUND'?
Yes
Select add object for recall
No
End recall
Remove 'CHAIN-HEAD' from successor
If successor is 'END-CHAIN-HEAD,' move tag from successor to selected object
Clean-up all guessing tags
End guessing strategy

Figure 4.19 Flowchart describing the guessing strategy used by the CFR Guessing Model when the next speech object to recall is not known. Based on similar flowcharts in Mueller (2002).

The chain-building process then begins. One of the remaining ‘not found’ objects is randomly selected as the current item and tagged as a ‘chain head’. The model attempts to follow the link from this item to its successor. If the next item is also ‘not found,’ then it is selected as current, and the model attempts to find its successor. If the successor to an item had already been tagged ‘found’, then its ‘chain head’ tag is removed, and the current item is marked ‘found’. If the successor was actually the ‘end-chain head,’ then this tag is removed and assigned to the current item. If the successor to the current item can not be found, then the current item is just marked ‘found’. After either an already found object is reached, or the next object cannot be found, the model stops, selects a new ‘not-found’ object and repeats this process. This continues until there are no more ‘not-found’ objects left. The result of this iterative process is that each contiguous chain of objects has its first object tagged as a ‘chain head’ or as the ‘end-chain head’, and the add object is marked as such.

At this point, the model randomly selects a ‘chain-head’ to recall next. If no chain heads exist, then the ‘end-chain head’ is selected. If it does not exist either, than the add object is selected. The normal recall process is resumed while all of the tags associated with this guessing strategy are removed. If later in recall another gap in the chain is encountered, then the guessing strategy is used again.

The CFR Guessing Model was run with the same parameter settings as the CFR Basic Model. Refer to Appendix A for a full listing of parameter values. In Figure 4.20, the response-type functions for this model can be seen in comparison to the Experiment 1 data. Unlike the prior model, which showed no errors of position within the sequence (i.e., ‘List Error’s), this model shows a pattern of such errors qualitatively similar to the observed data. It also shows a pattern of omissions (i.e., ‘No Response’) with an accelerating slope similar to the observed data. Both of these improvements are a direct result of the guessing strategy. When a single object decays from the recall chain before it is recalled, the chain containing the rest of the objects to recall will be discovered as the end-chain by the guessing strategy and recalled. This will lead to a number of list errors followed by a lack of response in the last serial position.

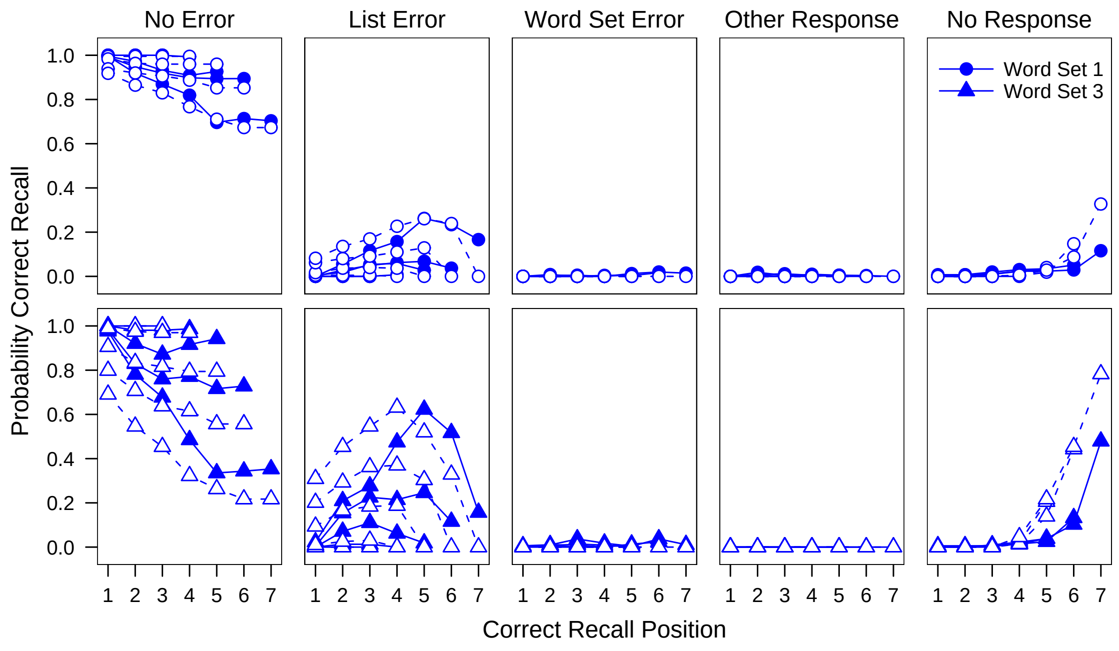


Figure 4.20 Comparison of response-type functions for CFR Guessing Model (open points connected by dashed lines) and Experiment 1 data (filled points connected by solid lines).

Based on this approximate analysis of the data, I might declare victory and move on. However, there is a second important way to look at the serial-position data that provides further insight into the underlying processes involved in task execution. These are the serial position functions (SPFs) shown in Figure 4.21. The observed ‘Relative Order’ SPFs show that many of the participants’ errors occur when they say the sequence of words in the wrong order. The model predicts almost no errors of this nature. This prediction stems from the interaction of the guessing strategy with the decay process. Only entire objects decay, while the remaining chain stays intact, so the recall is wrong because a word is skipped and the remaining words are shifted in serial position. However, as just described, this type of shift does not lead to an error of relative order. That is, the later words in the sequence are still said after the earlier words in the sequence. This clear difference between the observed and predicted data implies that the model is missing a fundamental process – the independent decay of serial order information. Thus, a third model of CFR involving link decay was developed to address this shortcoming.

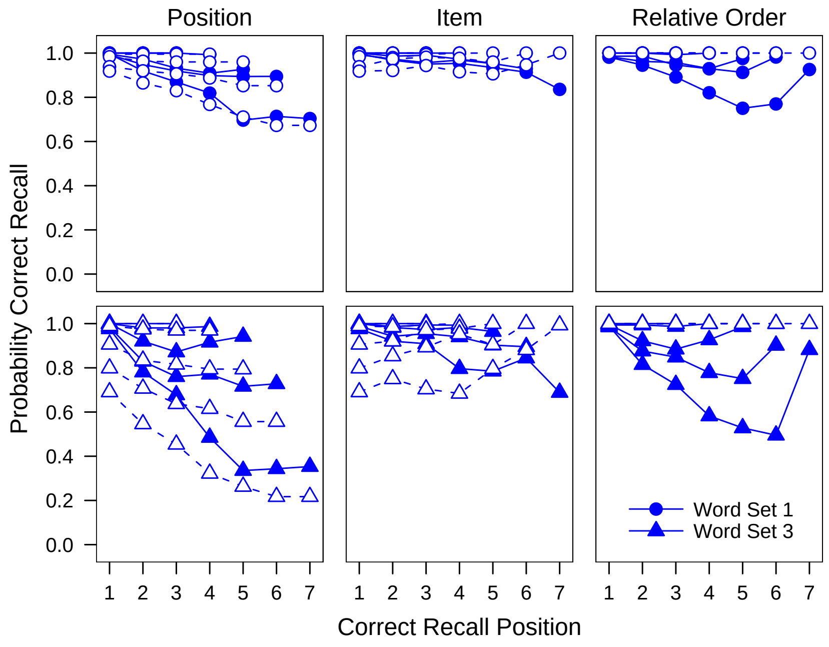


Figure 4.21 Comparison of serial position functions for CFR Guessing Model (open points) and Experiment 1 data (filled points).

4.3.3 Link-Decay Model

As already observed, the data clearly point to their being independent decay of serial-order information. On a theoretical level, such decay is supported in serial verbal working memory by a line of work dating back at least to Moore and Ross (1963). In their paper, which used an early version of the n-back task, they examined item repetitions to show that memory for item and order information is separable. More recently, McElree and Dosher (1993) used speed-accuracy trade-off analyses of recency judgments and item recognition to argue that item and order information are dissociated in verbal working memory. Also, Mueller (2002), in modeling ISR, found that independent decay of speech links was needed to account for serial position errors. In a review of converging evidence from multiple methodologies, Marshuetz (2005) argues that both behavioral and neural evidence support this distinction.

From an architectural point of view, the modified EPIC architecture used here lends itself to this independent decay. Serial-order information is represented as pointers within the speech objects. By setting the median and spread of a decay time distribution for these speech links, the loss of serial-order information due to decay can be modeled. When a link decays, the former value of the link, which was the ID of the next speech object in the sequence, is replaced with a value of 'GONE'.

This CFR Link-Decay Model took the CFR Guessing Model, added decay parameters for link information, and added new rules for additional error handling. Refer to Appendix C for a listing of these production rules. The algorithms that the model uses are identical when no information is missing, but they now have additional rules to handle the new types of failures that can occur. Whereas previously the only way that the recall chain could be broken was by a missing speech object, it is now possible that all of the speech objects remain, but the link pointing from one object to the next is lost. This is important because it enables the recall to be correct while still requiring the use of a guessing strategy during performance.

The particular values of the object and link decay parameters used here were selected to provide the best overall fit to the data. Refer to Appendix B for goodness-of-fit measures. Interestingly, once link decay was added into the model, it was necessary to reduce the speed of object decay significantly. Indeed, the data suggest that many of the errors that the earlier models were attributing to the decay of speech objects were in fact due solely to the decay of order information. A look at the observed serial position functions in Figure 4.22 shows the relative importance of order errors over item errors. Speech objects decayed with a mean of 6 seconds and a spread of 0.3, while speech links decayed with a mean of 3 seconds and a spread of 0.6. Refer to Appendix A for a full listing of parameter values.

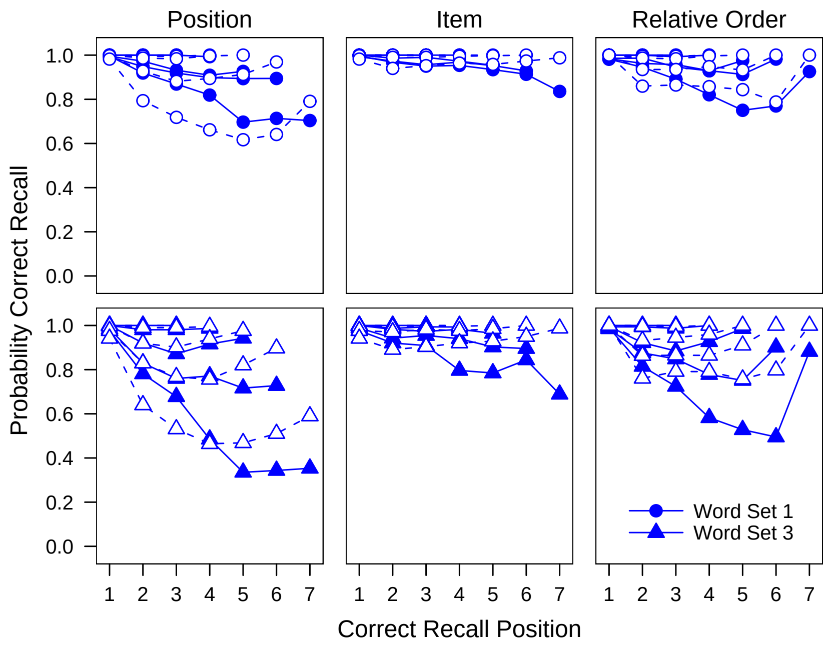


Figure 4.22 Comparison of serial position functions for the Link Decay Model of CFR (open points connected by dashed lines) and Experiment 1 data (filled points connected by solid lines).

The CFR Link-Decay Model provides a reasonable qualitative fit to the ‘Relative Order’ serial position functions. Accuracy is high at the beginning and end of the sequence, and is lower in the middle, and this effect is greater for longer sequences. The ‘Position’ serial position functions also show a reasonable fit, with high accuracy at the beginning of the list, and declining accuracy further down the list. There is a deviation at the longest sequence lengths where the model predicts a ‘recency’ effect that is absent in the observed data. This is because the recency effect due to order errors in the observed data is mitigated by item errors at the end of the sequence, as seen in the ‘Item’ SPFs. Investigations with various model parameters show that this increase in item errors with increasing serial position cannot be due to decay processes alone. Looking back at the performance of the CFR Guessing Model in Figure 4.21, it can be seen that the model predicts more item errors at earlier serial positions, not later ones. This is a subtle consequence of the timing of recall.

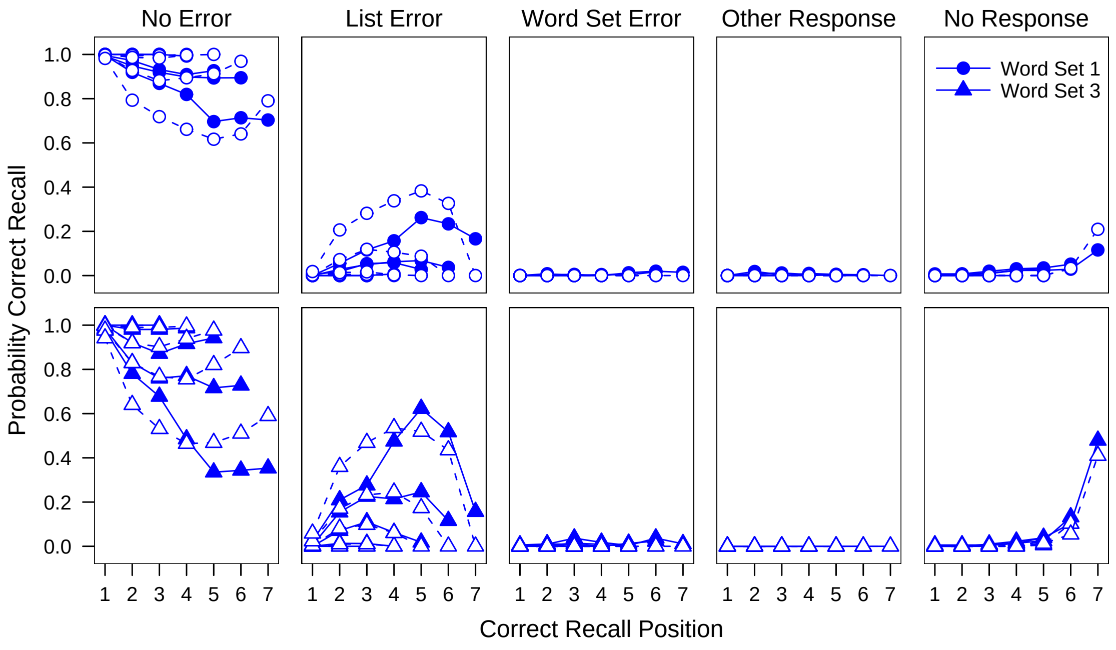


Figure 4.23 Comparison of response-type functions for CFR Link Decay Model (open points connected by dashed lines) and Experiment 1 data (filled points connected by solid lines).

It has been argued elsewhere that this pattern of worse recall for later items in conjunction with a list length effect can most parsimoniously be explained by a capacity limitation with overwriting of earlier items by new items (Mueller, 2002). However, since the magnitudes of these item effects are relatively small, and since the addition of a capacity limitation would be a significant departure from the decay approach taken heretofore, this avenue was not pursued in the current work.

In addition to improving the fit to the serial position functions, the CFR Link-Decay Model also fits the response-type functions fairly well (see Figure 4.23). In particular, the balance between list errors and omissions is in line with the observed data. Furthermore, the fit to articulation times has been improved (see Figure 4.24). This is a result of the guessing strategy in combination with link decay. Articulation times, particularly at longer list lengths, are longer than in the CFR Basic Model. Note that these articulation times are only for correct subtrials. Interestingly, this indicates that guessing does occur on correct subtrials. This is possible because links decay faster than objects, and so it is often the case that a link will decay, but the guessing strategy can correctly determine the sequence of the rest of the list. In the simplest case, a single lost link can be handled because the rest of the recall chain still forms a continuous chain to the end of the sequence.

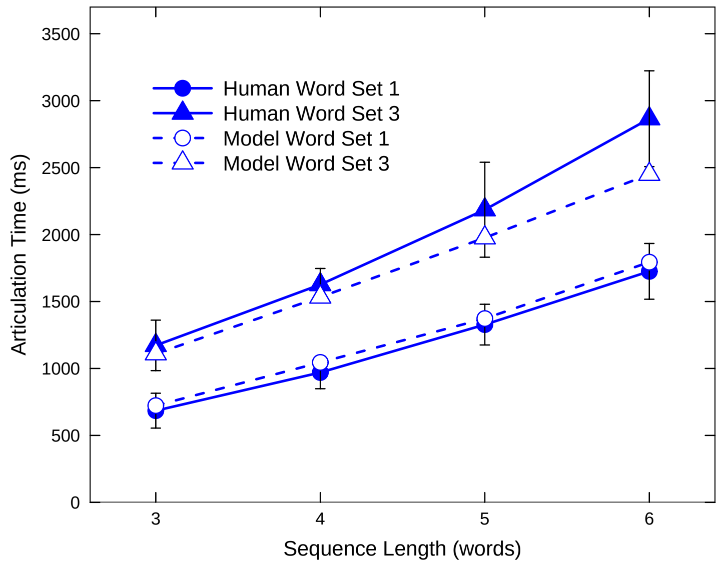


Figure 4.24 Comparison of subtrial articulation times for CFR Link-Decay Model and Experiment 1 data.

The CFR Link-Decay model successfully accounts for many important aspects of human performance in the CFR task. In this model, updating of the internal memory representation when a new word is added to the end of a sequence is performed through a process of articulation and perception. Careful analysis of the observed data from this task in conjunction with exploration of architectural and strategic components of the model has revealed important additional insights. Architecturally, serial-order information in the representation of the sequence can be lost independently of the information about the words themselves. Strategically, participants make use of a fairly sophisticated recovery strategy that completes recall in a reasonable way in the face of decaying information.

This model accounts for updating in the CFR task using a perceptual-motor mechanism. A further goal of this experiment is to show that CFR performance is not exceptional, but rather is making use of the same underlying verbal working-memory systems as other such performance. Toward that end, the principles and mechanisms used here are applied in a model of ISR. If these are indeed shared mechanisms, then the ISR model should be successful at predicting performance as well.

4.4 Modeling of ISR

The purpose of this model was to establish that performance on ISR can be accounted for within the same architecture and using the same parameters and general strategies as the present new rehearsal tasks. More specifically, the purpose was to better understand the relationship between performance on ISR and CFR in Experiment 1.

A great deal of previous work has been done on modeling ISR within the EPIC framework (Kieras et al., 1998; Kieras et al., 1999; Mueller, 2002). This work was used as the basis for the present ISR model. There are a couple of reasons for why this was the most appropriate route to take. The primary focus of my dissertation is to develop a detailed understanding of performance in the new rehearsal tasks, so the majority of effort was focused on modeling performance in those tasks. Secondly, as discussed already, ISR is not very strategically constrained. Thus, instead of spending a great deal of time accounting for this in modeling, it was dealt with behaviorally through the methodology of CFR. And finally, the prior work, in combination with the lessons learned in the CFR modeling described here, was found to be quite applicable to ISR.

The ISR Model performs covert rehearsal as the sequence of stimuli is presented. This is different than the CFR task, but is a phenomenon that is well supported in the ISR literature (e.g., by the effect of articulatory suppression during stimulus presentation), and has been modeled previously using EPIC (Kieras et al., 1999). During covert rehearsal, the ISR Model attempts to say the entire sequence of words that have been presented so far. Because words are continuing to be presented during this rehearsal, they are maintained in an 'add-chain' until they can be appended to the current recall-chain through articulation. When the recall cue occurs, the current rehearsal cycle is completed, and then the model begins overt recall. Overt recall includes the guessing strategy used in CFR to deal with information that has been lost through decay. The parameters are set such that speech links can decay independently of speech objects. The pattern of data requires this, since participants are much more likely to say the correct words in the wrong order than they are to say a word that is not in the rehearsal set. Also, as discussed previously, the dissociation of item and order information is well established in the literature.

For the ISR Model, refer to Appendix A for a full listing of parameter values, and Appendix B for goodness-of-fit measures.

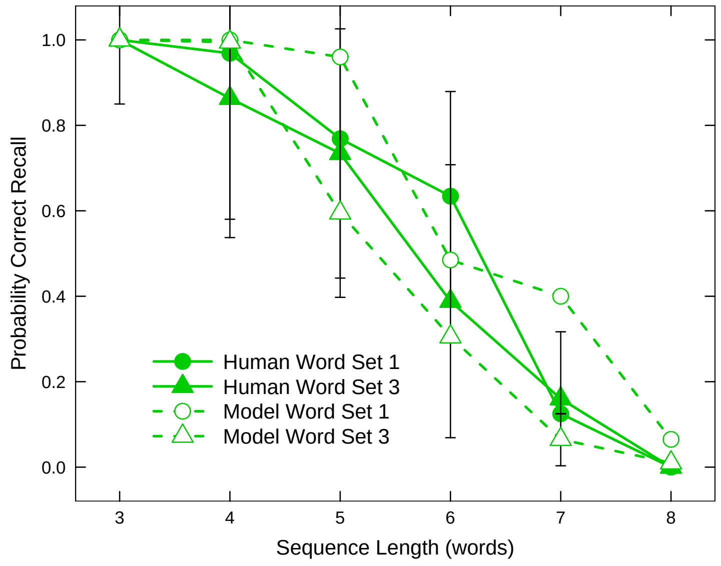


Figure 4.25 Comparison of accuracy for ISR Model and Experiment 1 ISR data.

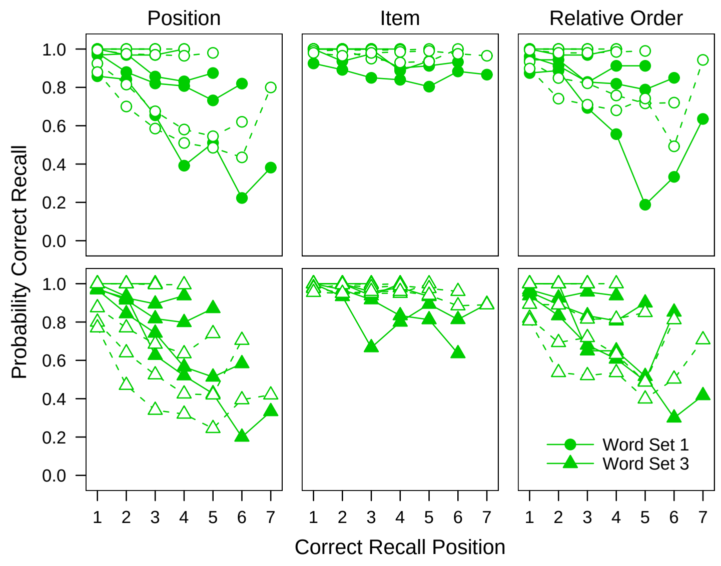


Figure 4.26 Comparison of serial position functions for ISR Model and Experiment 1 ISR data.

In accounting for the probability of correctly recalling the entire sequence of words on a trial (see Figure 4.25), the ISR Model matches the observed data in two crucial respects: accuracy decreases with sequence length, and accuracy is lower for words having longer articulatory duration. In addition, the overall performance envelope is matched; predicted performance is at ceiling for sequences of length three, and it is at floor for sequences of length eight, with a monotonic decrease over that range.

Serial position functions are also qualitatively matched by the ISR Model (see Figure 4.26) in much the same way that the CFR Link-Decay model fit the CFR data. The ‘Position’ SPFs and ‘Relative Order’ SPFs show the characteristic position and sequence-length effects, including recency effects in the last serial positions. Furthermore, the ‘Item’ SPFs predicted by the ISR Model fail to show the increase in errors with serial position as in CFR, again attributable to the lack of capacity limitations. These similarities in both the observed data and the models for ISR and CFR support the claim that the same underlying mechanisms are employed by participants in both tasks.

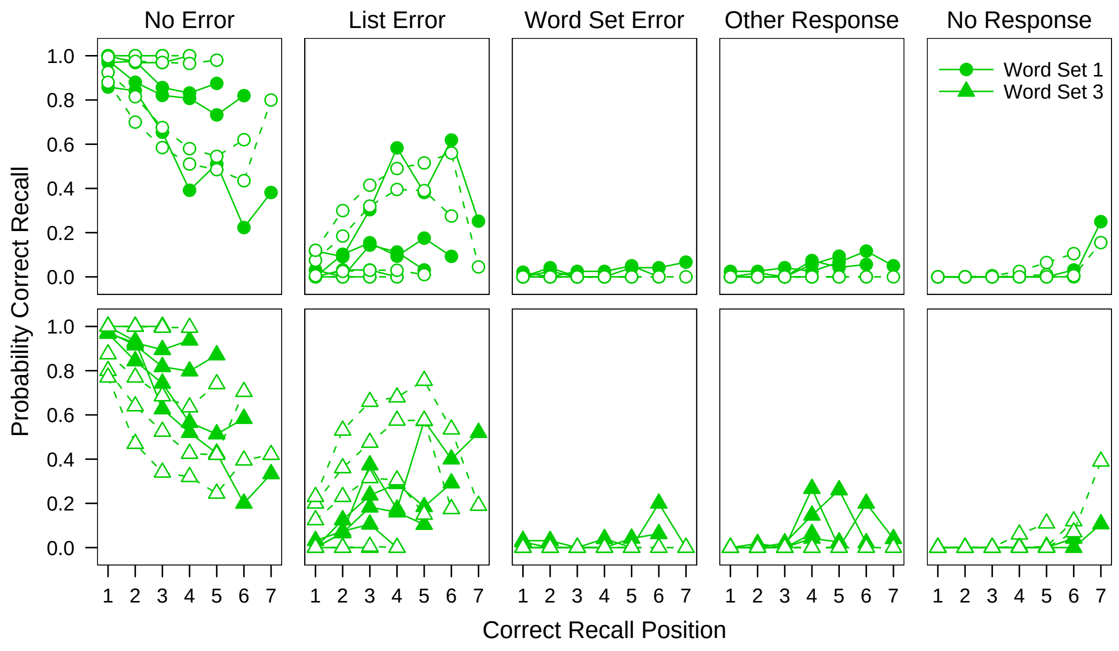


Figure 4.27 Comparison of response-type functions for ISR Model and Experiment 1 ISR data.

4.4.1 Further Comparison of CFR Link-Decay and ISR Models

While the ISR and CFR Link-Decay models fit their respective data sets in similar ways, there are a couple of additional issues to address about the relationship between these tasks and models. One, driven by the models, concerns the parameters used in the best fits; the other, driven by the data, concerns the relative level of performance in the two tasks.

The best-fit parameters for the ISR Model were different from those for the CFR Link-Decay Model. In particular, the mean object decay time for the ISR Model was twelve seconds whereas for the CFR model it was six seconds. This is a substantial difference; however, it should be noted that with equivalent compromise parameter values, reasonable fits are possible for both models. Object decay plays a relatively small role in the models compared to link decay, so performance is not particularly sensitive to the rate of object decay within this range of values. This outcome is certainly related to the use of small closed sets of words in the tasks, which places much more emphasis on memory for order information than for item information. Manipulations such as increasing the size of the word sets or even using novel words on each trial might shift the emphasis considerably toward item information, and then the model fits might be more sensitive to object decay parameters.

Yet, the difference in object decay time for the CFR and ISR models should not be entirely dismissed. It may be indicative of various differences in requirements and performance on the two tasks. In ISR, participants are not rewarded for rapid responses, which may encourage the introduction of additional processes, currently unaccounted for, during task execution. Also, in ISR, participants may say “blank” to indicate words that they do not know in the response. The fact that there were a noticeable number of ‘Other

Responses' in ISR (Figure 4.27) but not in CFR (see Figure 4.23) shows that this is taking place and is not accounted for by the ISR model used here.

In analyzing observed individual performance in the two tasks, a strong linear relationship was found between memory span for ISR and CFR (see Figure 4.9). Interestingly, the slope of this relationship was not one. In comparing different participants, they differed more in ISR span than in CFR span. Participants no doubt vary in many ways, but two dimensions of variation relevant to this issue involve mean articulation times and mean decay times.

To simulate this variation, the ISR Model and the CFR Link-Decay Model were run with articulation times higher or lower than the mean values across participants, and they were also run with decay parameters that were higher or lower than the best fit values. I found that the change in predicted memory span due to these changes was greater for ISR than for CFR, mirroring the relationship found for individual participants.

While individual differences are not a primary focus of this dissertation, the ability to account for this relationship between ISR and CFR memory spans strengthens the argument that these models are capturing performance in a meaningful way.

4.5 Discussion

One goal of Experiment 1 was to determine if there are shared mechanisms for performance of a canonical working-memory task, ISR, and the new CFR task. This was strongly supported both empirically and through modeling. Empirically, it was supported by the shared effects of sequence length and articulatory duration, and the correlation between memory spans. Through modeling it was supported by the success of models based on the same underlying mental processes accounting well for the data from both tasks.

The second main goal of this experiment was to test whether the perceptual-motor approach to verbal working-memory updating could account for performance in a basic form of updating, namely situations where the participant must append words to the end of a sequence. The success of the CFR model in accounting for human performance supports the idea that, at least for updating involving addition of items to the end of a list, a perceptual-motor treatment of updating can be successful. Note that this model posits no separate primitive updating operations beyond the articulatory and perceptual processes necessary for maintenance of a static sequence. This is not to say that executive control plays no role in updating, but simply that its role in updating is not fundamentally different from its role in maintenance.

Given this account of updating performance, can the same type of model be applied straightforwardly to performance in other rehearsal tasks that demand additional updating operations? The next experiment, with subsequent modeling, is intended to address this question.

CHAPTER 5

Experiment 2: Removing Words from the Beginning of Sequences

In Experiment 2, my exploration of updating operations was expanded to include removing words from the beginning of a sequence along with adding words at the end of a sequence. This type of updating has relevance to tasks where the participant is presented a continuous stream of information and has to take action based on a moving window of recent items. The study of this sort of updating has a long empirical tradition dating back to running memory tasks in research by Pollack et al. (1959).

More recently, the study of this type of updating has regained prominence through the popularity of the n-back task in brain imaging work (refer to Chapter 3 for more discussion of this). In this context, the n-back task has been considered a canonical test of working-memory function. As such, if the approach taken here can account for performance where appending and dropping words from the beginning of a sequence are both required, then it will be relevant to a large body of other recent work.

In some sense, this is a more significant test for the claims made here than were their applicability to immediate serial recall. The sort of updating involved in ISR, where words are added to the end of a sequence, has often been considered a part of "maintenance" in the loose theorizing done about verbal working memory. However, deleting words from the beginning of a list takes us fully into the realm of executive control and updating as typically conceived. Of course, this dichotomous view of appending and deleting runs contrary to the claims made here that maintenance and updating actually use the same processes and representations.

In Experiment 2, participants were tested with both the CFR and RFR tasks. Three word sets were used, and articulatory duration measurements were made, so that the articulatory duration effect could be evaluated across tasks.

This experiment had a number of aims:

First, it provides a replication of the results found for appending words to the end of a sequence in the CFR task. Since the final model of CFR performance is taken as a starting point for generalizing the perceptual-motor account of updating to a wider range of operations, it is worthwhile to confirm that those findings are reliable and reproducible.

Second, Experiment 2 provides the first empirical comparison of updating operations within the common framework of my new rehearsal tasks. Since both CFR and RFR have the same startup period, performance on the two types of updating can be compared head-to-head during subtrials.

Third, Experiment 2 expands the current investigation to address questions specific to the deletion operation in memory updating. As noted in Chapter 2, the field is unresolved about whether dropping words from a sequence requires active suppression/inhibition of information, or whether passive neglect might be sufficient.

Finally, fourth, in this experiment, I will explore whether the explanation of updating performance embodied in the CFR Link-Decay Model can be generalized and applied to the RFR task. Is the idea that updating is part and parcel of the same mechanisms that are involved in maintenance, which are themselves mechanisms of perception and action, sufficient to explain deletion operations as well as append operations in verbal working memory? And, furthermore, is the current representation of serial order sufficient to account for these data?

5.1 Method

5.1.1 Participants

The participants were 6 undergraduate students at the University of Michigan with normal cognitive function. They were between the ages of 18 and 30, right-handed, and spoke English as their first language. They were paid eight dollars per hour for their participation in 2 sessions of approximately 2 hours each on separate days, and earned additional monetary bonuses based on performance that typically amounted to a total of five to ten dollars.

Two additional participants (beyond the six analyzed) were excluded from analysis and replaced with other participants because of their inability to perform the tasks across the full range of conditions.

5.1.2 Apparatus

The same apparatus was used as in Experiment 1, except that the entire experiment was run using a single E-Prime 1.0 script.

5.1.3 Materials

Three experimental sets and one practice set of 10 words each were used. Word Sets 1 and 3 were the same as those in Experiment 1. Word Set 2 consisted of words having two syllables (see Table 4.1). These words were selected in the same way as those for the other word sets. The phonological similarities of the experimental word sets were equated on mean onset dissimilarity by using the PSIMETRICA technique (Mueller et al., 2003). The words in the experimental sets were all nouns selected to be similarly low in concreteness and roughly matched on familiarity and Kučera and Francis (1967) written frequency as reported in the MRC Psycholinguistic Database (M. D. Wilson, 1988). Table 4.1 lists the words and relevant measurements for each set.

5.1.4 Design

Participants were tested individually with three procedures: an articulatory duration (AD) measurement, cumulative forward rehearsal (CFR), and rolling forward rehearsal (RFR). Testing occurred in two sessions of 1.5 to 2 hours on separate days. In one session, CFR with the practice word set was followed by AD and CFR for each of the experimental word sets. In the other session, RFR with the practice word set was followed by AD and RFR for each of the experimental word sets. AD always immediately preceded CFR and RFR for each experimental word set. The word sets were used in the same order in both sessions. The order of CFR and RFR across sessions was counterbalanced across participants. The order of short, medium and long word sets within sessions was counterbalanced across participants using a Latin square.

5.1.5 Procedures

Articulatory duration measurements and CFR were performed in the same way as in Experiment 1.

5.1.5.1 Rolling Forward Rehearsal

The organization of RFR trials was the same as CFR trials. The startup period of each trial was identical. The difference was in the expected response on each subtrial. While in CFR the sequence length increased on each subtrial by adding the new word to the end of the sequence, in RFR the sequence length remained constant on each subtrial, because the new word was added to the end but the first word in the previous sequence was dropped.

There were four consecutive blocks of the RFR task for each word set. The initial number of words presented on each trial stayed constant throughout a block. Trials in the first block started with two words, in the second with three, in the third with four, and in the fourth with five. A block ended after a trial when a total of 40 subtrials or 10 trials were completed, whichever came first.

The words in the startup set were selected randomly without replacement from the word set. The word to present on each subtrial was selected randomly from those words in the word set not in the current sequence to be recalled. Thus, no word ever occurred more than once in a single recall sequence.

At the beginning of each trial, a message was visually presented indicating the number of words in the startup sequence. One second into this presentation, the number of words was also presented auditorily. After the visual message was presented for 3 s, it was removed and replaced with a 1.3-s fixation. The trial then continued with a startup period followed by zero or more subtrials (see Figure 3.1) followed by feedback.

In the startup period, the initial sequence of words was auditorily presented with 100-ms inter-stimulus intervals. Then, 100 ms after the last word of the startup sequence, a 100-ms tone was played to cue recall. The participants tried to recall the startup sequence. They were instructed to begin rehearsal as soon as they could and to recite quickly and accurately. The start time of their articulation was recorded by a voice key, and the finish time and accuracy were recorded by the experimenter pressing one of two keys. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played and the first subtrial began after a 300-ms response-stimulus interval.

On each subtrial, a word was presented auditorily, and the subject had to respond as soon as possible by quickly and accurately reciting an updated sequence. This sequence consisted of the previous sequence without the first word followed by the newly presented word. As during the startup period, reaction time, articulation time, and accuracy were recorded. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played, and the next subtrial began after another 300-ms RSI.

A trial ended after an error was made, or after the ninth subtrial. Visual feedback was then provided for 4 s, informing the participant of the correct response, the number of subtrials completed, average reaction time, average articulation time per word, and the number of points earned on the trial.

To encourage effortful performance, participants earned points based on reaction time, articulation time and accuracy. A monetary bonus was paid based on the number of points earned. On each subtrial, the maximum points that could be earned were the number of words in the correct response multiplied by 5000. No points were earned for a subtrial if an error was made. For a correct subtrial, one point was subtracted from the maximum for each millisecond of articulation time and reaction time. Participants were paid a bonus of one dollar for every 1,500,000 points earned.

5.1.6 Data Analysis

The data analysis was the same as in Experiment 1, with the addition of one new analysis specific to RFR.

5.1.6.1 Word-Set Error Functions

In response-type functions, one type of error is a word-set error, where a word is said that is in the current word set, but is not in the current response sequence. For CFR, these were always words that had not been presented at any point on the current trial, because all words presented on the current trial were in the current sequence to be recalled. However, in RFR, since a word is dropped from the sequence on each subtrial, word set errors can either be words that were never presented on the current trial, or they can be words that were presented and were previously part of the recall sequence, but have since been dropped. In the latter case, the word can be classified in terms of how long ago it was dropped from the recall sequence.

To create the word-set error functions (WSEF), the incidence of these different types of word-set errors on subtrials was analyzed. The total number of word-set errors for which the erroneous word was last included *n* recalls previously (*n*-back) was counted for each value of *n*. The total number of opportunities for each of these types of errors was then counted.

Note that the number of opportunities varies with *n*, because, on earlier subtrials within a trial, there are only recently dropped words. For example, on the first subtrial in a trial, exactly one word has been dropped from the sequence, and so the recall of each word in the sequence is an opportunity for a one-back error. On the second subtrial, two words have now been dropped, and so the recall of each word is an opportunity for either a one-back error or a two-back error. This explains why it is necessary to divide the number of errors for each *n* by the number of opportunities for that type of error, to get a measure of the prevalence of that type of error as a percentage of total opportunities. Finally, an additional value provides the rate of word-set errors where the erroneous word was never in the recall sequence during the trial.

The WSEFs are potentially interesting in that they provide an indication of the time course of likelihood that a word dropped from the recall set will erroneously reappear in the recalled sequence.

5.2 Results

The articulatory duration measurements are reported first, as they establish that the materials indeed provided a manipulation of articulatory duration. The CFR and RFR results are then organized by their relevance to each of the stated aims for this experiment.

Table 5.1 Experiment 2: Articulatory durations and memory spans per word set.

	Word set		
Task and variable	1 (short)	2 (medium)	3 (long)
AD			
Duration (ms)	271	352	419
CFR			
Memory span (words)	7.11	6.50	5.99
RFR			
Memory span (words)	5.46	4.68	4.63

5.2.1 Articulatory Duration Measurements

Articulatory duration measurements for each word set for each participant were aggregated across the two sessions. The mean estimated articulatory durations for each word set are shown in Table 5.1. In a one-way within-subjects ANOVA for the articulatory durations, the main effect of word set was reliable, $F(2, 8) = 136.2$, $p < 0.001$, with mean duration for Word Set 2 greater than Word Set 1, and Word Set 3 greater than the mean of Word Sets 1 and 2. Thus there is a reliable difference in mean articulatory duration across the three word sets used in this experiment.

5.2.2 Replication of CFR Results

One of the aims of Experiment 3 was to confirm that the results found in Experiment 1 for CFR are replicable. This is important both to establish the stability of the procedure used for the new rehearsal tasks, and to increase confidence in the conclusions drawn from modeling the results. Qualitatively, the pattern of results for CFR in Experiment 2 looks the same as in Experiment 1. Quantitatively, the key findings from Experiment 1 held here as well.

The accuracy data followed the same pattern as in Experiment 1. In a two-way within-subjects ANOVA for the probability of correct recall, the main effects of word set, $F(2, 8) = 13.15$, $p = .003$, and sequence length, $F(4, 16) = 67.42$, $p < .001$, and their interaction, $F(8, 32) = 4.87$, $p < .001$, were all reliable. Table 5.1 lists the mean estimated memory spans for each word set obtained from the fit functions. In a one-way within-subjects ANOVA for the memory spans, the main effect of word set was reliable, $F(2, 8) = 15.86$, $p = .002$, with memory span for Word Set 2 less than Word Set 1, and Word Set 3 less than the mean of Word Sets 1 and 2.

The reaction time data also showed the same important findings of a word set effect but no sequence length effect. In a two-way within-subjects ANOVA for the reaction times, there was a reliable effect of word set, $F(2, 8) = 13.27$, $p = .003$, but no reliable effect of sequence length, $F(3, 12) = .19$, $p = .9$, and no reliable interaction, $F(6, 24) = 1.05$, $p = .42$.

Finally, the articulation times also largely mirrored the results from Experiment 1. Using a two-way within-subjects ANOVA, I found reliable effects of word set, $F(2, 8) = 23.99$, $p < .001$, and sequence length, $F(3, 12) = 28.98$, $p < .001$, as well as a reliable interaction, $F(6, 24) = 3.79$, $p = .008$. Looking at individual contrasts for the effect of sequence length, I found a reliable linear trend, $F(1, 4) = 85.5$, $p < .001$, but, unlike in the previous experiment, the quadratic trend was not reliable, $F(1, 4) = 1.39$, $p = .262$.

Overall, performance on CFR was very similar in the two experiments.

5.2.3 Comparison of Updating Operations

A second aim of this experiment was to empirically compare performance for the two types of updating associated with CFR and RFR. The first step is to compare performance during startup. Startup is procedurally identical in the tasks, and is intended as a way to create the same initial state before updating begins. As such, performance during the startup period should be very similar for the two tasks in order to provide an equivalent starting point for the execution of updating operations.

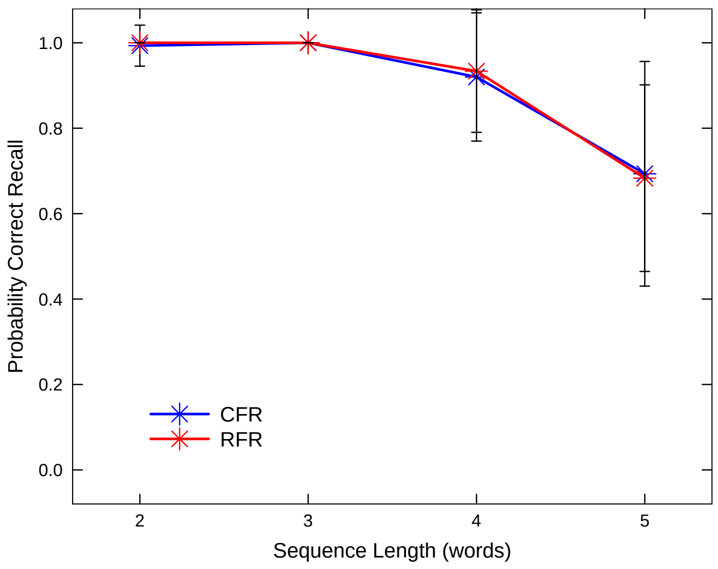


Figure 5.1 Experiment 2 startup period: Observed mean probabilities of correct recall of the entire sequence as a function of task and sequence length.

As seen in Figure 5.1, the startup accuracies as a function of sequence length are extremely similar for CFR and RFR. This was supported by a three-way within-subject ANOVA with sequence length, word set, and task as factors. The difference in accuracy between CFR and RFR was not reliable, $F(1, 4) = .05$, $p = .83$. Similar results were found for reaction times and articulation times during startup. Three-way within-subject ANOVAs with sequence length, word set, and task as factors showed no reliable effect of task for reaction time, $F(1, 4) = .59$, $p = .49$, or articulation time, $F(1, 4) = .006$, $p = .94$.

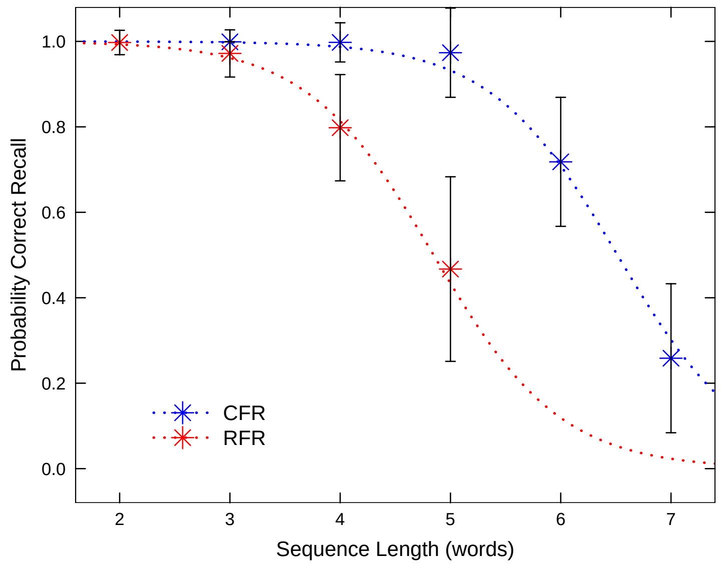


Figure 5.2 Experiment 2 subtrial: Observed mean probabilities of correct recall of the entire sequence and estimated mean logistic regression functions (dotted lines) as a function of task and sequence length.

With a reasonable match between startup performance on CFR and RFR confirmed, the tasks can be compared during subtrials when the different types of updating are occurring. Looking first at the accuracy of performance, it is clear that accuracy drops down at a smaller sequence length for RFR than CFR (see Figure 5.2). This is confirmed by a two-way within-subject ANOVA which shows that memory span for RFR is reliably less than memory span for CFR, $F(1, 4) = 88.85$, $p < .001$. In addition, the main effect of word set is reliable, *F*(2, 8) = 26.33, *p* < .001, while the interaction of word set and task is not, *F*(2, 8) = 2.23, *p* = .17. Furthermore, an ANOVA for accuracy in RFR shows that, as in CFR, there is a reliable effect of word set on memory span, *F*(2, 8) = 26.46, *p* < .001, with memory span for Word Set 2 less than Word Set 1, and Word Set 3 less than the mean of Word Sets 1 and 2 (see Table 5.1). Thus, this indicates that performance of RFR is harder than CFR, but the use of the articulatory loop is critical to performance for both tasks.

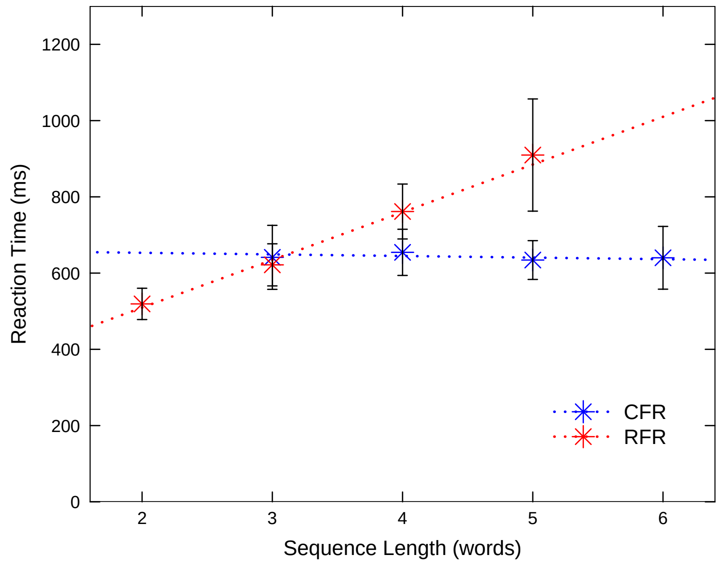


Figure 5.3 Experiment 2 subtrial: Observed mean reaction time as a function of task and sequence length with linear fits (dotted lines).

A comparison of the two tasks on reaction time performance reveals a clear qualitative difference (see Figure 5.3). While RTs in CFR do not change as a function of sequence length, there is a clear linear increase in RT with sequence length for RFR. This was confirmed by fitting lines to the per-word-set reaction times for each task. For CFR, the slopes of the lines were not reliably different from zero, *F*(1, 4) = .068, *p* = .81, while, for RFR, they were, *F*(1, 4) = 11.256, *p* = .028. The implications of this finding will be explored in the course of modeling. At an abstract algorithmic level, it suggests that in the RFR task there is an additional process before recall begins whose time course increases linearly with the number of items in the recall sequence.

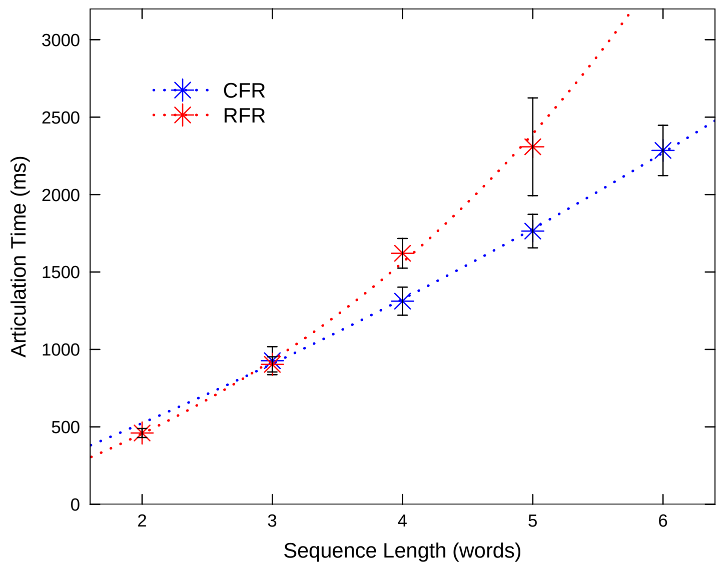


Figure 5.4 Experiment 2 subtrial: Observed mean articulation time as a function of task and sequence length with mean exponential fits (dotted lines).

In terms of articulation time, performance was slower for RFR than for CFR at longer sequence lengths (see Figure 5.4). As described in Experiment 1, exponential fits were made to the articulation time data. These fits have two parameters related to the curves, *d*, the base duration of articulating a single word, and *a*, the acceleration that describes how much longer each word takes to say as the number of words to say gets larger. Two-way within-subject ANOVAs were run for each of these parameters with task and word set as factors. For *d*, the effect of word set was reliable, $F(2, 8) = 32.68$, $p >$ .001, with the longer words taking longer to articulate, while the effect of task was not reliable, $F(1, 4) = 1.93$, $p = .24$. For *a*, the effect of task was reliable, $F(1, 4) = 12.5$, $p =$ .024, with RFR having more acceleration than CFR, while the effect of word set was not reliable, $F(2, 8) = 2.75$, $p = .12$. This supports the idea that the basic articulation time for a word was a function of the word itself and was unaffected by the task. However, the additional time to say the sequence on top of simple articulation of individual words (i.e., the gaps between words) was not affected by word set, but was a function of the task being performed. This may indicate that in RFR more processing is being done on average to determine each word to recall. For example, the use of a guessing strategy may be required more frequently in RFR than in CFR.

In summary, reliable differences in performance for CFR and RFR were found across all three major dependent variables. In all cases, deleting words from the beginning of the list in addition to adding words to the end of the list led to worse performance. The implications of these findings for generalizing the CFR Link-Decay Model will be considered subsequently.

5.2.4 Removing Words from the Beginning of Sequences

A unique aspect of RFR performance is the need to drop a word from the previous recall sequence each time a new word is added. The processes necessary to account for how participants are able to do this will be considered in detail by modeling the data. However, to better understand the deletion operation, it makes sense to consider what happens to the ostensibly deleted words. In particular, do these words intrude into rehearsal in later sub-trials? Insight into these processes may be possible by looking at the error data in particular ways. The word set error function (WSEF), described in the methods section, provides such a tool.

The WSEF for RFR in Experiment 2 is shown in Figure 5.5. There are a couple of key features in these data. First, word-set errors are not very common. Out of all the words that should have been uttered in performing RFR, those utterances classified as word-set errors were less than three percent of the total responses. Second, when a word-set error did occur on a subtrial, the erroneous word was very likely to have previously been in the recall sequence during that trial. Third, the erroneous word was highly likely to have been the most recently dropped word. And finally, the likelihood that a dropped word shows up in recall declines rapidly with the number of intervening subtrials.

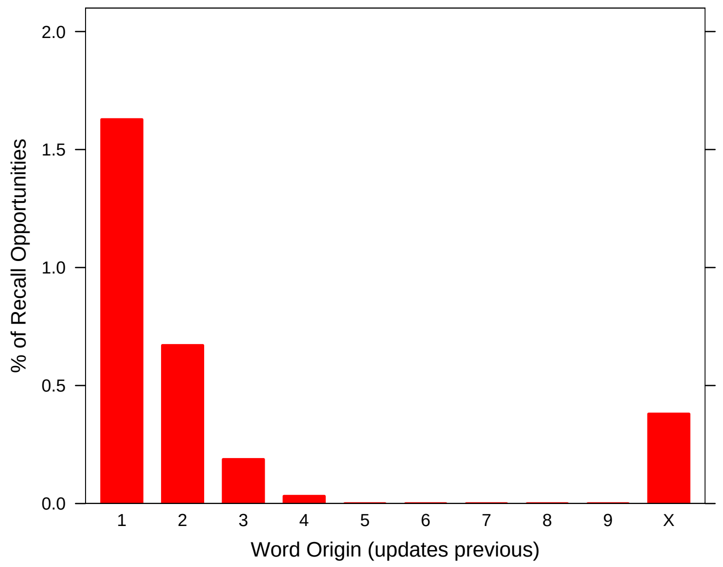


Figure 5.5 Experiment 2 RFR: Word set error function. The rightmost bar, labeled 'X,' indicates errors where the erroneous word was not in the recall sequence earlier in the trial.

This is exactly what we would expect if words are passively neglected when dropped from a sequence. The most recently used words would be the least likely to have decayed away, and as found, this likelihood would increase rapidly with increasingly distant deletion. On the other hand, if deletion primarily depended upon active inhibition or suppression, then we would expect the most recently deleted word to be the least likely to intrude, because the effect of inhibition would be greatest on it.

While the WSEF tells us where the erroneous words originated, a look at the response-type function for RFR shows us where the word-set errors went (see Figure 5.6). More word set errors occurred at intermediate serial positions than at the start or end of the sequence.

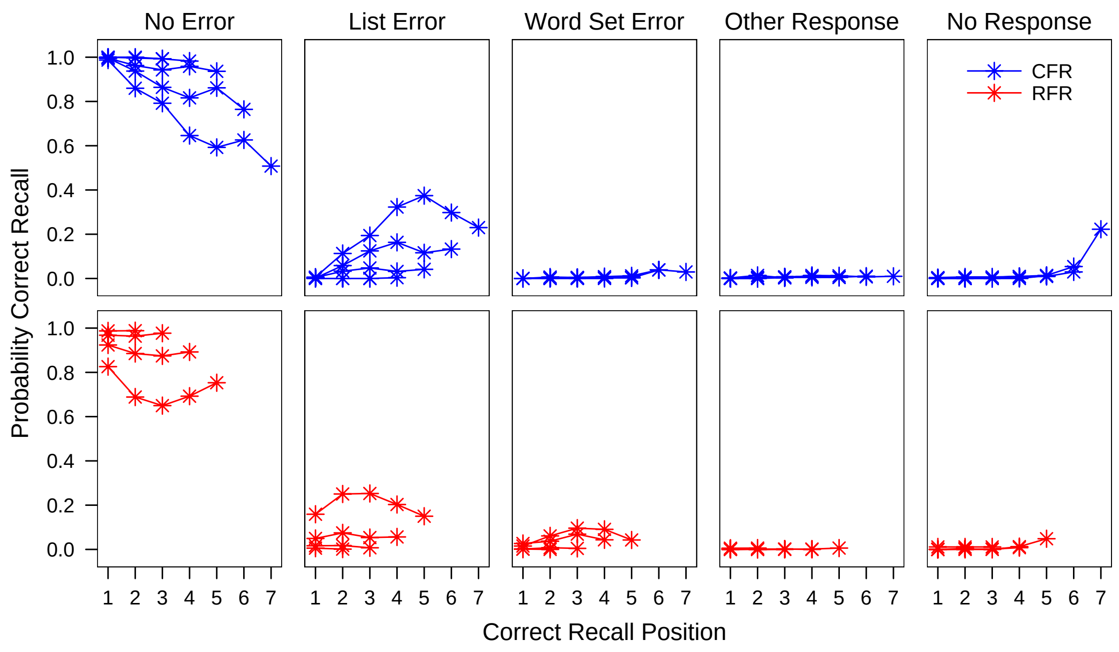


Figure 5.6 Experiment 2 subtrial: Response-type functions for each task.

Again, this is what we would expect if the words are passively neglected, and are used mistakenly during a strategic guessing process. They would be expected to appear in the positions where guessing most often takes place, namely near the end of a sequence, but not at the very end, since the last item is the new item and acts as a known anchor. On the other hand, in an inhibition account that relies on strength for serial-position representation, a failure to inhibit would be expected to lead to the word occurring close to where it previously occurred in the list, namely near the beginning.

Thus the intrusions in RFR, both in terms of where they come from and where they occur, are more consistent with deletion by passive neglect than deletion by active inhibition/suppression. Yet, the question remains whether a passive approach to deletion can account for the larger patterns of accuracy and time course in the RFR task. In order to evaluate this, modeling is necessary.

5.3 Modeling of RFR

The fourth and most critical aim of Experiment 2 was to test the generalizability of the idea that updating of serial verbal working memory is a perceptual-motor process that makes use of the same machinery as basic processes of perception, action, and memory maintenance.

5.3.1 Least-Change Model

The most straightforward way to test the generalizability of the CFR Link-Decay Model was to use the ideas captured in it to make a model of the RFR task. I did this by taking the CFR Link-Decay Model and making the minimal number of changes that were necessary for it to properly perform the RFR task. The changes were constrained only by what was allowable in the EPIC architecture. If there are not any further constraints on how the resources available are deployed, then this approach would lead to a reasonably accurate model. However, if there are other constraints on the use of the underlying architecture which have yet to be revealed, then the resulting model may deviate from the observed data in significant ways. At this point (that is to say, before considering the actual results of the RFR task) we have no reason to insert other constraints on the operations that are possible.

The initial model of RFR, the RFR Least-Change Model, was constructed by taking the CFR Link-Decay Model and making the minimal number of changes that were necessary for it to properly perform the RFR task. Specifically, at the start of subtrial recall, after the new stimulus has been heard, the model simply follows the link from the first item in the chain to the second item in the chain without articulating the first. In this way, the item is dropped from the articulated sequence, and thus does not appear in the new chain created from the perception of recall. In this RFR model, the process of deletion is passive in that the word is simply not said and thus no new representation of the word will be created and the old representation will decay over time. Furthermore, as in the CFR models, the ‘updated’ memory representation is created through the process of articulation and perception, not through direct manipulation of the memory representation itself.

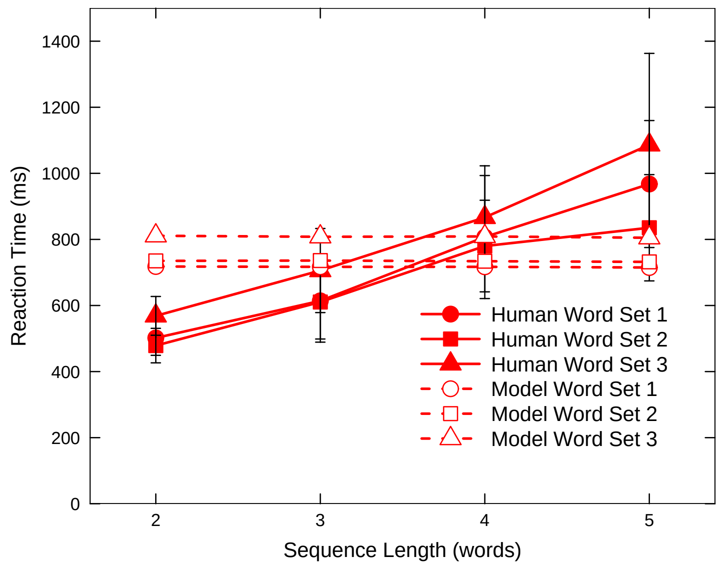


Figure 5.7 Comparison of subtrial RTs for RFR Least-Change Model and Experiment 2 RFR data.

Since the RFR Least-Change Model is so similar to the CFR Link-Decay Model, the predicted performance is also very similar. The RFR reaction times are slightly slower than CFR times, because of the need to skip the first item in the chain before recall begins. However, as in CFR, RFR reaction times are unchanging with sequence length (see Figure 5.7). This disagrees with the empirical results. As reported in the results section, the observed RFR data have a reliable upward slope in RT as a function of sequence length. The failure of the RFR model to predict this slope is a significant shortcoming and indicates that there are processes unanticipated by the CFR model alone.

The next step is to determine what processes could account for the reaction time pattern found in the observed RFR data. Once a candidate is determined, it will be appropriate to return to more fundamental issues, including the constraints on the architecture that make this seemingly less efficient approach necessary, and the reasons that the particular solution to those constraints is employed.

5.3.2 Covert-Articulation Model

In theorizing about what could be occurring during performance of the RFR task, two critical features of the RTs need to be considered. The obvious feature is the upward slope of approximately 150 milliseconds per word. A second feature is the occurrence of very low RTs at the shortest sequence length. At length two, the RTs for RFR are less than they are across sequence lengths for CFR. This indicates that the process leading to the RFR RT slope has a very small intercept or else it starts before the new stimulus is presented. Given that modeling of past VWM tasks, such as ISR, has supported the claim that covert rehearsal of verbal material takes place (Kieras et al., 1999), this was an immediate candidate for consideration here. Perhaps participants were covertly rehearsing the sequence before producing it overtly during recall.

Preliminary investigations indicated that for this strategy to be viable, the covert rehearsal must start before the presentation of the new stimulus. This follows from the fact that the RTs at sequence length two are around 500 milliseconds. Since the RT ‘clock’ starts at the onset of word presentation, there simply is not time for any covert rehearsal starting when the new stimulus is perceived. In fact, there is hardly time to initiate overt articulation.

The first event prior to the presentation of the new stimulus is the feedback signal indicating correctness on the previous subtrial. This auditory signal not only lets the participants know that they have answered the previous subtrial correctly, but also confirms that another subtrial is about to occur, and is thus a reasonable point in time for the participants to begin preparation. Furthermore, since the time between the feedback signal and the presentation of the new stimulus is a constant response-stimulus interval, the participants can begin preparation confident in the timing of the subsequent subtrial.

Preliminary investigations also indicated that the covert rehearsal must consist of only the previous words in the sequence minus the word to be dropped on the current subtrial. In other words, neither the word to be dropped, nor the new stimulus, is included in the covert rehearsal. There simply is not sufficient time for more rehearsal given the timing of the observed responses.

The timing constraints of this situation also indicate that after covert rehearsal has been performed, if the new stimulus has not yet been perceived, the first word for overt recall is sent to the vocal motor system for preparation only. This is so that when the new stimulus is perceived, the first word can be produced as quickly as possible.

Based on these findings, the RFR Least-Change Model was modified to form a new RFR Covert-Articulation Model. As soon as the feedback signal is given, this new model begins covert rehearsal of the recall chain that it has just finishing hearing. This covert rehearsal begins by passively skipping the first word in the recall chain, as was done overtly in the Least-Change Model. The rest of the words in the recall chain are then covertly articulated one after another as quickly as possible.

If a break in the recall chain is encountered due to either object decay or link decay, the model restarts rehearsal by randomly selecting one of the remaining objects to recall. This is a considerably less sophisticated recovery strategy than used for overt recall. The primary impetus for this choice was that rehearsal is clearly done very quickly while the full guessing strategy takes considerable time. However, given how soon after the previous overt recall the covert rehearsal occurs, it is unlikely that decay will have occurred, and so this recovery strategy is used much less frequently than the overt guessing strategy. As such, the impact of the particular strategy chosen for this subcomponent may not play a large role in the overall performance.

Once the covert rehearsal has finished, if the RFR Covert-Articulation Model has not yet perceived the new stimulus, then the first word to be overtly recalled (i.e., the first word in the covert recall chain) is sent to the vocal motor processor to be prepared. Once the model has perceived the new stimulus, then overt recall begins. Overt recall is done using the memory representation created from the covert rehearsal. This relies heavily on the ability of the EPIC architecture to fire multiple rules simultaneously, as the internal perception of the covert rehearsal may very well be underway as overt recall begins. Note that overt recall will start with the first item in the covert chain, and will end by tacking the new stimulus on after the recall chain. Thus the overt recall for RFR (once covert rehearsal has been done) is quite similar to recall in the CFR task.

In summary, subtrial performance in the RFR Covert-Articulation Model starts immediately after feedback from the previous subtrial with covert rehearsal of the previous sequence minus the word to be dropped. Once this is complete and the new stimulus has been perceived, overt recall begins based on the covert sequence. The covert sequence is repeated and followed by the 'add' item.

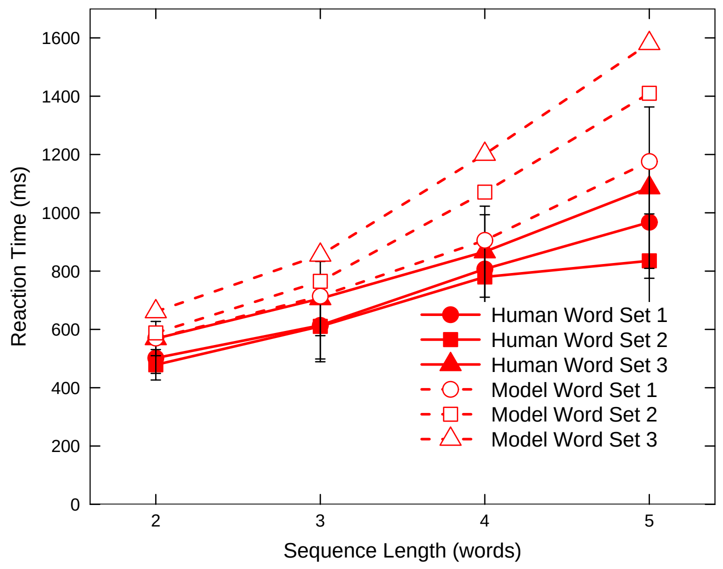


Figure 5.8 Comparison of subtrial RTs for RFR Covert-Articulation Model and Experiment 2 RFR data.

The RT performance of this model is shown in Figure 5.8. This model is an improvement over the Least-Change Model in that it predicts large RT slopes, and the RT slope for Word Set 1 is quite close to the observed value. However, the model fails to fit the RFR RT data in two main ways. First, the slopes are different for the different word sets, while in the human data the differences were smaller and not reliable. And second, the slopes for Word Sets 2 and 3 are too steep.

At this point, we have the RFR Least-Change Model which provides a lower bound on performance with an RT slope of zero, and the RFR Covert-Articulation Model which provides an upper bound on performance with RT slopes that are too steep. The RT slopes of the veridical RFR model must lie between these two extremes.

5.3.3 Mixture-Based Approaches

One approach to reconciling the two models developed so far would be to consider a mixture model. In such a model, the strategies of each of the two previous models would be used on a percentage of trials. By manipulating this percentage, any slope between the slopes of the two individual strategies could be attained. The distribution of RTs at the higher sequence lengths generated by this model would have a distinctly bimodal distribution, since, on each individual trial, the covert rehearsal either did or did not occur.

In order to assess the relevance of such a model to the observed data, a preliminary investigation of observed RT distributions was performed. This was considered at two levels of analysis: between-subjects and within-subjects. At the between-subjects level, it was possible that some participants used one strategy while other participants used the other. In this case, the bimodal distribution would show up in the distribution of the RT slopes calculated separately for individual participants. However, inspection of the distribution of slopes for all participants for all word sets showed a unimodal distribution centered on the mean slope. This suggested that qualitatively different strategies were not being used by subpopulations of the participants.

At the within-subjects level, it was possible that each participant was using one strategy on some trials and a different strategy on other trials. In this case, the bimodal distribution would be expected in the distribution of individual subtrial RTs at the higher sequence lengths for each participant considered separately. Inspection of distributions for individual participants with individual word sets revealed no evidence for such bimodality. In particular, there were no clusters of fast RTs at high sequence lengths near to the values found for much lower sequence lengths.

Since the observed data did not suggest a mixture of two qualitatively different strategies either between or within participants, mixture models were not considered further for RFR.

5.3.4 Memory Search Approaches

Another approach to finding a model with an intermediate RT slope is to consider other processes that may depend on the sequence length, but be smaller in magnitude. One class of such processes involves memory search. Search processes, which would not engage motor production, could be significantly faster than the articulatory processes of the Covert-Articulation Model.

The role of search processes in verbal working memory has been investigated extensively. The initial work in this area was done with the item recognition task, where a serial search through memory has been claimed to take place (Sternberg, 1966). Later work linked the findings from the item recognition task to memory span in immediate serial recall (Cavanagh, 1972). Cavanagh (1972) included a meta-analysis of search rates in which he found the mean search time per word to be 47 ms. A more recent paper found a memory search time per one-syllable word of 76 ms (Hulme, Newton, Cowan, Stuart, & Brown, 1999). This paper also linked memory search to immediate serial recall, arguing that a search process occurs during the pauses in recall during ISR. These values fit well with the EPIC architecture, where 50 ms is the cognitive cycle time. Indeed, the 'natural' rate of a search process implemented in EPIC that walks down a chain of words would be one cycle period or 50 ms per item. However, a slope of 50 to 75 ms will be too shallow to account for the observed data in the RFR task. In other words, the type of search process found in the item recognition task is too fast to account for the RT slopes here.

Another type of memory search that has been related to recall of word sequences involves the processes of speech preparation and production. It has been claimed that initiating the production of a word sequence uses a rapid self-terminating serial search (Sternberg, Monsell, Knoll, & Wright, 1978). However, the rate of these search processes is even faster – on the order of 10 ms per item. Furthermore, there is no reason to think that these preparatory processes would be any more or less evident in RFR compared to CFR.

Memory search of yet another sort has been suggested by speed-accuracy tradeoff analyses for judgment of recency (JOR) and n-back tasks (McElree, 2001; McElree & Dosher, 1993). Through mathematical modeling, McElree has obtained estimated search rates of 250 to 300 ms or more per item. In these tasks, a probe item is compared to a sequence of items held in memory. In the n-back task, the probe is compared to the item *n* back in the sequence, while in the JOR task, two probes are assessed to determine which came later in the sequence (i.e., more recently). McElree argues that, on at least some percentage of trials, participants perform both of these tasks by using a serial self-terminating search to find the relevant items in the sequence.

It is not clear how the need for explicit comparison in these tasks relates to the recall requirements of RFR. However, the per-word search times for JOR and the n-back task are in the same ball park as those for RFR, and thus it is worth considering this similarity in more detail. Since McElree uses only mathematical models, he is able to remain agnostic about the nature of the search processes involved. In other words, he can avoid specifying what processes are occurring during the several hundred milliseconds per word of search time. Nevertheless, from the point of view of the perceptual-motor approach taken here, the obvious hypothesis is that a process of covert articulation may occur. In order to walk down the sequence of items making word to word comparisons along the way, McElree's participants might be covertly articulating the sequence held in memory. This brings us back to the idea of covert rehearsal that I dismissed previously because it predicted RT slopes that were too steep. Covert articulation is now reconsidered.

5.3.5 Truncated-Articulation Model

In reconsidering the role that covert articulation might play in RFR, let us return momentarily to the Covert-Articulation Model. A key point is that the predicted reaction time function for Word Set 1, the one syllable words, was quite reasonable both in terms of slope and intercept (see Figure 5.8). The major deviations were for Word Sets 2 and 3, the two and three syllable words, where the model predicted significantly steeper slopes due to the longer articulation times for these words, while the data show essentially the same slope. This suggests that covert articulation could be occurring if it was performed rapidly and with truncation of later syllables for the longer words.

Given that a number of other seemingly more likely options have been ruled out, it is worth considering this option seriously. In terms of task demands, there was certainly a strong incentive for participants to begin overt articulation as quickly as possible. Participants were encouraged to respond quickly in the task instructions. Participants also earned monetary bonuses by collecting points for fast reaction times (as well as accurate responses and rapid articulation times).

There is also evidence suggesting that covert rehearsal may be truncated under speeded conditions. Dell and Repka (1992) specifically addressed this question by comparing speech errors in speeded overt and covert rehearsal. Previous research had already found that covert speech is often faster than overt speech. They found that speech errors in covert speech tended to occur in word onsets. They concluded that participants are truncating their covert rehearsal to include only the first parts of words.

Additional support for this possibility is found in work on the phonological similarity effect in immediate serial recall. In quantifying phonological similarity, Mueller et al. (2003) found that syllable onsets are much more important than the nucleus or coda in determining the size of the phonological similarity effect. This could very well be due to truncation during the covert rehearsal that is known to take place during presentation of stimuli in ISR.

Since the current observed data and past work are both consistent with truncated covert articulation during serial verbal working-memory tasks, a new RFR Truncated-Articulation Model was developed that makes use of it. This model was based on the RFR Covert-Articulation Model described previously. The prior model was modified by truncating the covert rehearsal to the first syllable of each word. This was accomplished by using the same short articulation times for covert speech for all three word sets. (Note that the data show some suggestion that the longer words had slightly longer RTs, but since this was not a reliable difference, it is not considered further here.)

Overt recall was also modified in the RFR Truncated-Articulation Model. It uses a more conservative approach to production. In the CFR models, the production of speech was maximally pipelined. This meant that the production rules would send the next word to the vocal motor system as soon as the prior word was prepared for articulation. The actual production of the prior word would then occur in parallel with the preparation of the next word. In this way, the next word would be ready for articulation as soon as the prior word was completed. The ability to perform in this way is built into the EPIC architecture. This is supported by abundant evidence that speech is often produced such that pipelined production is necessary. Indeed, looking at the waveform of natural speech, it is not uncommon for there to be no discernable gaps between consecutive words.

Pipelining was scaled back in the RFR Truncated-Articulation Model, so that preparation of the next word did not begin until the articulation of the previous word was complete. Initial investigations indicated that this was necessary to explain the reliable slow down in articulation times reported for RFR compared to CFR.

One possible reason for this more conservative strategy is that generating overt recall from the truncated covert recall requires a more effortful and time-consuming process. Another possibility is that the RFR task is perceived as being harder by the participants, and as a result they take a more cautious approach to task execution. If this is the case, the adoption of this more conservative approach would itself contribute to worse performance by making it more likely that words decay before they are articulated. This seeming conservatism during the overt recall process contrasts with the apparent aggressiveness of the strategy employed before overt recall. The truncated rehearsal is an aggressive approach, as is the anticipatory preparation of the first word in the recall chain (as discussed in the prior description of the Covert-Articulation Model). At this point, the reason for these contrasting periods of aggressive and conservative strategy remains speculative.

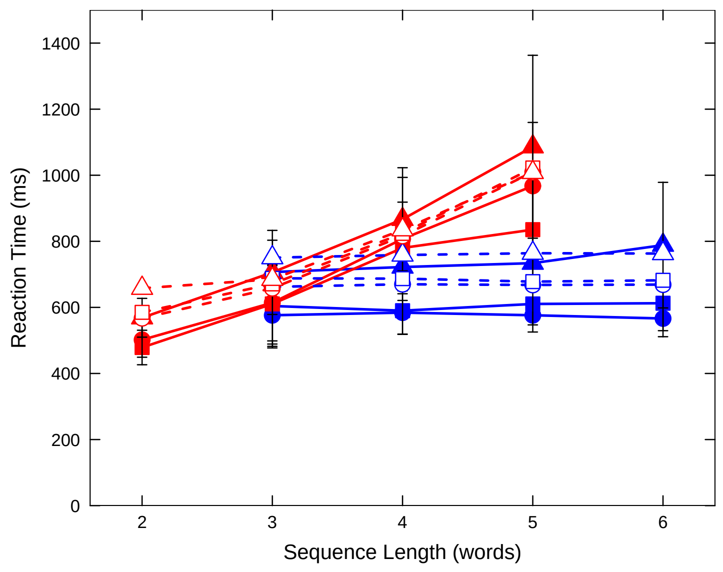


Figure 5.9 Comparison of subtrial RTs for RFR Truncated-Rehearsal Model (open red points) and observed Experiment 2 RFR data (filled red points), and CFR Link-Decay Model (open blue points) and observed Experiment 2 CFR data (filled blue points). Word sets are indicated by shape (Set 1 by circles, Set 2 by squares, and Set 3 by triangles).

The results of running the Truncated-Articulation Model are now compared to the observed RFR data from Experiment 2. The CFR Link-Decay model and the observed CFR data from Experiment 2 are also included for comparison. Both models were run with the same decay parameters used for the CFR model in fitting the Experiment 1 data. Refer to Appendix A for a full listing of parameter values, Appendix B for goodness-of-fit measures, and Appendix C for production rules.

The CFR Link-Decay Model fits the CFR RT data fairly well, particularly for Word Set 3 (see Figure 5.9). More importantly, reaction times for the RFR Truncated-Articulation Model provide a good fit to the upward slope seen in the observed RFR data. This is a direct result of the truncated covert-rehearsal process that is occurring before overt articulation on each subtrial. The largest deviation from the observed data occurs at the shortest sequence length, where the model predicts longer than observed RTs. This happens because, even with anticipatory preparation of the first word in the recall chain, the model cannot account for the rapid response at sequence length two. This may indicate that overt recall, like covert rehearsal, is starting before the new stimulus is perceived.

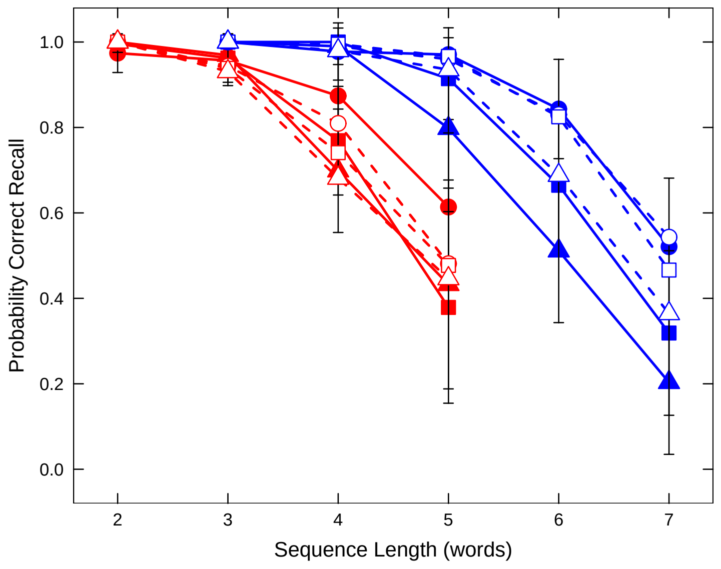


Figure 5.10 Comparison of subtrial accuracies for RFR Truncated Rehearsal Model (open red points) and observed Experiment 2 RFR data (filled red points), and CFR Link-Decay Model (open blue points) and observed Experiment 2 CFR data (filled blue points). Word sets are indicated by shape (Set 1 by circles, Set 2 by squares, and Set 3 by triangles).

For the accuracy data (see Figure 5.10), the most important feature predicted by the models is that the accuracy curves are lower for RFR than CFR with the same decay parameters. It is not obvious that this follows from the way the CFR model was modified into the RFR model. Part of this effect is due to the fact that the word dropped in RFR, the first word in the sequence, is the word most likely to be recalled correctly in CFR. As a result, the overall likelihood of recalling all four words in an RFR subtrial for sequence length four is less than the likelihood of recalling all four words in a CFR subtrial of sequence length four. In some sense, an RFR subtrial of sequence length four is more like a CFR subtrial of sequence length five, since there are five words to handle in memory even though the first one is not articulated.

However, this cannot be the whole story. Comparing CFR and RFR accuracies shows that, for both the observed and the predicted data, accuracy on RFR subtrials of length *n* is less than accuracy for CFR subtrials of length *n* + 1. This residual shortfall in RFR accuracy is related to the slower RTs at large set sizes, the slower rate of articulation, and the processes of covert rehearsal.

The effect of word set is seen in both the models and the data for RFR and CFR, but the effect is larger in the data than in the models. This shortcoming is related to the recovery strategy. Since the recovery strategy does not directly involve rehearsal processes, but takes a considerable amount of time, it dilutes the effect of differences in articulatory duration between the word sets. Perhaps participants incorporate covert rehearsal into recovery, which would increase the word set effect on both accuracy and articulation time.

The predicted serial position functions for the RFR Truncated-Articulation Model show a number of key effects seen in the data (see Figure 5.11). The 'Position' and 'Relative Order' SPFs both show a sequence length effect. They also show that, unlike for CFR, the first item in the sequence has a substantial chance of being recalled incorrectly, especially at larger sequence lengths. In addition, the 'Relative Order' SPF shows a recency effect for the last item.

However, as in CFR, the lack of a capacity limit precludes the 'Item' SPFs from capturing the increase in item errors with serial position. This is particularly true for the last serial position, which is the new word appended to the sequence. Since this word is presented later and has the same decay function as other words, it is remembered particularly well by the RFR model, while in the observed data, it is left out of recall more often than any other word. This suggests that, in this context, external speech may decay faster than internal speech. While different decay rates for different speech sources were not pursued here, this may be necessary for optimal model fitting.

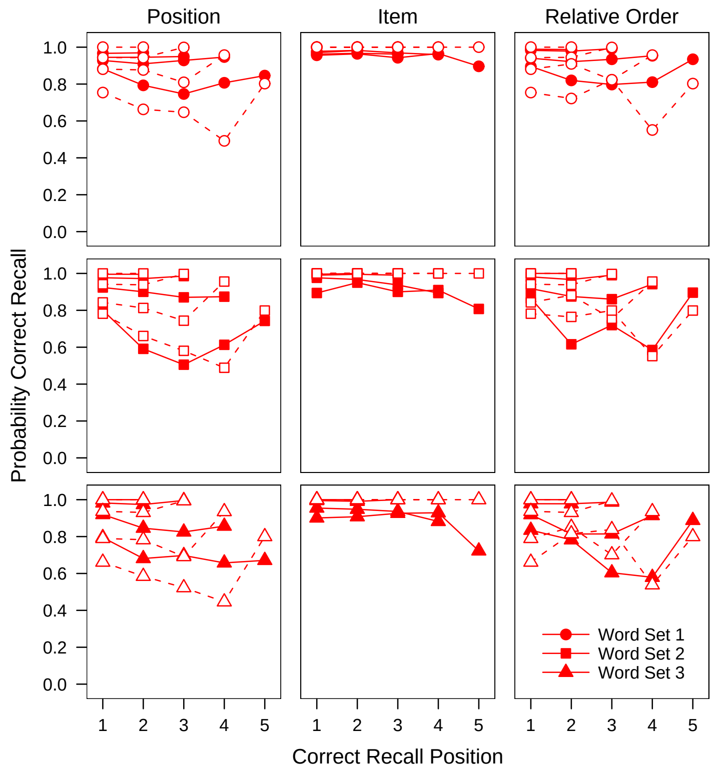


Figure 5.11 Comparison of SPFs for RFR Truncated Rehearsal Model (open points) and observed Experiment 2 RFR data (filled points).

The RFR Truncated-Articulation Model is based on the most successful model of CFR, the Link-Decay Model. The initial model of RFR failed to account for the surprising linear increase in RT with sequence length. In order to account for this effect, a truncated covert rehearsal process was added that performs the updating operation of dropping a word from the beginning of the sequence. The new word is then added through the overt articulation process. While various residual deviations of the model from the data suggest various further avenues to pursue, overall, this model accounts for many major features of the observed RFR data. The implications of this model for the larger theoretical questions about working-memory updating are addressed next.

5.4 Discussion

An initial aim of Experiment 2 was to replicate the Experiment 1 CFR findings about appending words to the end of sequences. This was done both empirically and through modeling. Empirically, the observed data for CFR in Experiment 2 showed the same overall pattern as in Experiment 1. The CFR Link-Decay Model was run with the same parameters, except for the use of new articulation times from the Experiment 2 participants. As was the case for Experiment 1, the model's predictions captured many important aspects of the Experiment 2 data. This replication raises confidence in the Experiment 1 conclusions about processes involved in verbal working-memory updating.

A second aim of Experiment 2 was to compare performance with the append operation to performance with another updating operation. This experiment introduced the operation of omitting words from the beginning of a sequence. This deletion operation has been considered indicative of memory tasks that require executive control as well as maintenance (Jonides et al., 1997). Empirically, a number of differences were found between performance on CFR, where only appending operations were required, and RFR where both appending and dropping from the beginning of a sequence were required. For a given sequence length, accuracy is lower and articulation time is greater when both updating operations are required. However, for both accuracy and articulation time, an effect of articulatory duration is found in both updating situations. While reaction times did not change as a function of sequence length when words were appended, they increased linearly with sequence length when both operations were performed. This set of differences indicates that performance of the RFR task differs qualitatively from performance of the CFR task.

A third aim of Experiment 2 was to evaluate the generalizability of the model developed for CFR to RFR in order to better understand the processes responsible for the differences in performance. The most straightforward adaptation of the CFR model failed to account for the surprising linear increase in reaction time with sequence length found in RFR. Based on a consideration of different alternatives, it was determined that the best explanation involves a process of truncated covert rehearsal. This process performs the working-memory updating operation of deleting a word from the beginning of a sequence by passively omitting it from a cycle of covert rehearsal. A new word is then appended to the sequence through an overt articulation process as it was in CFR. While various deviations of the model from the data suggest further avenues to pursue, overall this model accounts for many major features of the observed RFR data.

Critically, the RFR Truncated-Articulation Model accomplishes appending and deletion with a set of perceptual-motor processes which use the same perceptual and articulatory machinery and the same types of control as in CFR, and as would be used in any verbal working-memory task that involves active maintenance of information. This expands to a new operation the claim that verbal working-memory updating is not a distinct process of executive control, but rather a flexible redeployment of perceptual and motoric resources to meet task demands.

The representation used here is one of decaying speech objects with bound phonological content and pointer-like serial order information that can decay independently. While decay alone may not account for all aspects of the data, it is sufficient to explain significant features of the serial position curves. Evidently, this form of representation can provide the substrate for performance on a task that demands operations of deletion as well as appending.

A fourth aim of Experiment 2 was to further understand the mechanisms of deletion. The approach to deletion used in the RFR model is one of passive neglect. There is no active process of suppression or inhibition of information in order to 'delete' the dropped item. By simply not articulating a word, it is not propagated to the newly created memory representation. The old representation is eventually lost through decay over time. This passive approach is sufficient to account for key differences between RFR and CFR performance, including lower accuracy, which might otherwise be attributed to the impact of effortful processes of suppression or inhibition.

A unique analysis of the origin of intrusions into recall was also consistent with a passive account of deletion. The most recently dropped words were the most likely to intrude, with the rate of intrusion dropping off rapidly with decreasing recency. This is consistent with the dropped words remaining passively in memory and decaying over time. It is less consistent with an alternative account where the words are actively suppressed when they are dropped and then the suppression is released over time. According to this alternative, we would expect the most recently dropped words, which are also the most strongly inhibited, to be among the least likely to intrude.

5.4.1 Constraints on Updating

The successes of the RFR Truncated-Articulation Model beg the question of why participants perform the RFR task in the way that they do. Why do they perform covert rehearsal when dropping and appending words, but not when they are only appending words? Why do they rehearse in the specific pattern used?

To approach these questions, recall how the initial, failed, RFR model was constructed: it was a minimal modification of the CFR model. The only constraints placed on the adaptation were those of the EPIC architecture. This model failed by overpredicting performance. This suggests that there are in fact additional constraints on human updating performance that need to be understood.

In the initial model of RFR, the updating strategy involved skipping articulation of the first word in the sequence, saying the rest of the recall chain, and then saying the new stimulus item at the end. Both updates (dropping and appending) occurred in a single overt cycle of rehearsal. In the final model of RFR, the updating strategy involved skipping articulation of the first word in the sequence and then saying the rest of the recall chain during a cycle of covert rehearsal. Then the covert sequence was recalled overtly followed by appending the new stimulus. The two updates occurred in separate cycles of rehearsal. These two strategies are just two cases from of a wider family of possible rehearsal strategies for working-memory updating in RFR.

To understand the full space of updating strategies, it is necessary to identify the dimensions along which they may vary. One is clearly the parallelism of the updating operations – do they occur on the same cycle of rehearsal or different cycles of rehearsal. A second deals with the types of rehearsal performed. A cycle of rehearsal takes a memory representation of a sequence as its input and generates a new representation as its output. We can specify for both the input and the output whether the representation is based on overt or covert rehearsal. A third dimension deals with the way in which deletion occurs.

In the models used here, deletion occurs by skipping a word during recall, but another possibility would be to articulate the word covertly instead of overtly (or vice versa). This could cause the word to be represented separately in memory from the recall chain, effectively removing it from the sequence.

Table 5.2 lists the possible articulation strategies for updating in RFR based on the dimensions just described. There are additional strategies that can be generated within this space of possibilities, but they all involve at least three cycles of rehearsal. Since three cycles of rehearsal would take an unreasonably long time, these possibilities are not considered further. The table indicates not only the pattern of articulation used in each strategy but also where it falls along each dimension, the general pattern of reaction times it would predict, and the number of words that must be rehearsed covertly on each subtrial to implement the strategy. The initial RFR Least-Change strategy is Strategy 1. The successful RFR Truncated-Articulation strategy is Strategy 8.

Since participants were encouraged to make rapid responses, and were given feedback about reaction times, I assume that they were trying to begin overt articulation in an efficient manner. To a reasonable approximation, the fewer words rehearsed covertly by a strategy before overt articulation begins, the shorter the RTs will be. With no other constraints, Strategy 1 is the clear winner, since it requires no words to be covertly rehearsed. However, if there is one additional constraint that append and delete operations cannot both occur on the same cycle of rehearsal (i.e., updates must be sequential, not simultaneous), then Strategy 8 is the best remaining strategy, because it requires $n - 1$ words to be covertly rehearsed.

Table 5.2 Articulation strategies for updating in RFR.

Strategy	Articulation	Parallelism	Deletion Method	Input & Output Modalities	RT Pattern	Words Rehearsed Covertly
1.	_, B, C, $\underline{D}$	Simult.	Skip	O → O	—	0
2.	$\underline{a}$, B, C, $\underline{D}$	Simult.	Switch	O → O	—	1
3.	a, b, c | _, B, C, $\underline{D}$	Simult.	Skip	c → O	∕	n
4.	a, b, c | $\underline{a}$, B, C, $\underline{D}$	Simult.	Switch	c → O	∕	n + 1
5.	_, b, c, $\underline{d}$ | B, C, D	Simult.	Skip	O → c	∕	n
6.	a, b, c, $\underline{d}$ | _, B, C, D	Sequent.	Skip	Add: O → c Del.: c → O	∕	n + 1
7.	a, b, c, $\underline{d}$ | $\underline{a}$, B, C, D	Sequent.	Switch	Add: O → c Del.: c → O	∕	n + 2
8.	_, b, c | B, C, $\underline{D}$	Sequent.	Skip	Del.: O → c Add: c → O	∕	n - 1

Note. Clarifications described by column. **Articulation:** Just prior to the shown sequence, the participant has said "A, B, C" and then heard 'D.' Letters represent articulated words. Uppercase indicates overt articulation. Lowercase indicates covert articulation. Underlines indicate deletion and addition operations. Vertical bars indicate boundaries between rehearsal cycles. **Parallelism:** Simult. = addition and deletion in a single rehearsal cycle. Sequent. = addition and deletion in sequential cycles. **Deletion Method:** Skip = deletion by not articulating a word. Switch = deletion by switching output modality. **Input & Output Modalities:** O = overt articulation. c = covert articulation. Add = addition operation. Del. = deletion operation. "→" = input modality to output modality. **RT Pattern:** "—" = no change in RT with rehearsal set size. "∕" = linear increase in RT with rehearsal set size. **Words Rehearsed Covertly:** The total number of words rehearsed covertly in the strategy. **n** = sequence length.

This provides a clear reason why Strategy 8 would be used given the need for sequential operations – but why is sequentiality needed? One possibility, suggested by recent research with variants of the Garavan memory updating paradigm, is that working-memory updating is non-selective (Kessler & Meiran, 2006). Kessler and Meiran argue that when multiple items are held in working memory and all are potentially subject to updating, all items are updated whenever the task requires the updating of a single item. In the memory updating paradigm used by Kessler and Meiran, the updating was always the modification of an item already in the sequence. However, this principle can be generalized to RFR. In RFR, the operation of dropping the first word from the sequence would necessitate the updating of the entire sequence through covert rehearsal. Only then could the second operation, appending the new word to the end of the sequence, be performed by again rehearsing the entire sequence, this time overtly.

CHAPTER 6

Experiments 3 and 4: Prepending Words to Sequences and Removing Words from the End of Sequences

By moving from appending words to removing words from the beginning of a sequence, significant new lessons were learned about working-memory updating. The new, more informed ideas account for performance on both CFR and RFR. One way to evaluate the generality of these ideas is to further expand the range of updating operations under study. Natural next operations are prepending a word onto the beginning of a sequence, and removing a word from the end of a sequence. I studied these with two more new procedures, cumulative backward rehearsal (CBR) and rolling backward rehearsal (RBR).

In CBR, participants prepend each new stimulus item presented during a subtrial onto the beginning of the sequence recited during the previous subtrial. This does not include the sequence presented during the startup period, which is recalled in the same order as it is presented (as in CFR and RFR). As a result, performance of CBR may or may not be similar to performance of the backward immediate serial recall task with which it shares a surface similarity. In backward ISR, the entire sequence is presented before overt recall, so the internal representation constructed and maintained by the participant could very well be the typical forward list. In this case, the 'reversal' of the sequence would be performed all at once as the final overt recall is produced. On the other hand, participants could instead build up the sequence in reverse order by prepending each word as it is presented. In CBR, the requirement of overt rehearsal on each subtrial guarantees that the prepending is done incrementally.

While the specific details of generalizing the CFR Link-Decay Model to predict CBR performance will be saved for the subsequent discussion of modeling, it is worth considering them at a high level here in order to further motivate the next experiments. According to the perceptual-motor account which uses a linked list of speech objects, prepending is not much different than appending, in that the difference is just a matter of when a new item is articulated in relation to the previous recall chain. By saying the new item first followed by the recall chain, the resulting memory representation will have the new item prepended. On this account, the reaction times for CBR should remain constant across sequence length as they did for CFR.

In contrast, other approaches to verbal working memory would make different predictions. One significant class of models represents serial order as a set of ordered slots holding items (e.g., O'Reilly & Soto, 2002). Another class uses ordinal tags attached to items (e.g., Burgess & Hitch, 1999; Henson, 1998). In models like these, prepending an item would require putting it in the first slot or giving it the first tag, and thus shifting the rest of the words down a slot or tag. The time for this process would presumably increase linearly with the number of items in the sequence, suggesting a linear increase in RT with increasing sequence length. Given these divergent predictions, the results from CBR will potentially provide strong support or refutation for the representation used in the present models.

Another new task, rolling backward rehearsal (RBR), was also used in these experiments. During RBR, words are not only prepended to the current sequence but also dropped from its end. Analogically speaking, RBR is to RFR what CBR is to CFR. A key hypothesis introduced at the end of the previous chapter was that updating is non-selective (i.e., a cycle of rehearsal is required for each updating operation). Since RBR requires two updating operations on each subtrial, it provides a test of the non-selective updating hypothesis. In other words, RBR, like RFR, should show a linear increase in RT with sequence length due to the need for a cycle of covert rehearsal before overt recall begins.

Since RBR requires the deletion operation, it also provides a second task context in which to evaluate the evidence for passive versus active deletion processes. If the reason for intrusions in RFR is indeed the passive neglect of words, then the same intrusion pattern would be expected in RBR.

Two experiments were run to investigate prepending and deletion from the end of a sequence. In Experiment 3, all four tasks (CFR, RFR, CBR, and RBR) were run in a within-subject design. While this allowed for within-subject comparison of all of the working-memory updating operations, it required that only one word set be used in order to keep the experiment to two sessions. In Experiment 4, CBR and RBR were run in a within-subject design with two word sets. This provided replication of results from Experiment 3, and also allowed for investigation of the articulatory duration effect in CBR and RBR.

Experiments 3 and 4 and the subsequent modeling had a number of aims. The first aim was to replicate the RFR and CFR results from Experiment 2. In particular, for RFR, I was interested in confirming the critical linear increase in reaction time with sequence length. The second aim was an empirical comparison of performance across four different tasks having different updating demands within the common framework provided by the rehearsal tasks. The third aim was to evaluate deletion processes in a new task, namely RBR. And the fourth aim was to test the generalization of the perceptual-motor account of working-memory updating, as embodied in the CFR Link-Decay Model and the RFR Truncated-Articulation Model, to the operations of prepending and deleting from the end of a sequence.

6.1 Experiment 3 Method

6.1.1 Participants

The participants were 12 undergraduate students at the University of Michigan with normal cognitive function. They were between the ages of 18 and 30, right-handed and spoke English as their first language. They were paid eight dollars per hour for their participation in 2 sessions of approximately 2 hours each on separate days, and earned additional monetary bonuses based on performance that typically amounted to a total of five to ten dollars.

6.1.2 Apparatus

The same apparatus was used as in Experiment 2.

6.1.3 Materials

One experimental set and one practice of 10 words each were used. The experimental set used was Word Set 1 (see Table 4.1).

6.1.4 Design

Participants were tested individually with five procedures: an articulatory duration (AD) measurement, cumulative forward rehearsal (CFR), rolling forward rehearsal (RFR), cumulative backward rehearsal (CBR), and rolling backward rehearsal (RBR).

Testing occurred in two sessions of 1.5 to 2 hours on separate days. On each day, AD was performed first followed by two of the rehearsal tasks. CFR and RFR (the forward rehearsal tasks) were always tested on the same day, and CBR and RBR (the backward rehearsal tasks) were always tested on the other day. For each rehearsal task, first it was run with the practice word set and then with the experimental word set. The order of tasks across days (forward and backward tasks) was counterbalanced across participants, as was the order of tasks within days (cumulative and rolling tasks). The within-day ordering was always the same for both sessions for a participant.

6.1.5 Procedures

The AD task was performed in the same way as in Experiment 2. The CFR and RFR tasks were performed in the same way as in Experiment 2, except that the order of blocks was different. In this experiment, there were still four blocks for each rehearsal task, with startup sequence lengths of two, three, four and five words, but the order of sequence lengths was counterbalanced across participants using a Latin square. This is the only experiment where the order of startup sequence lengths was counterbalanced. This was done in case the ascending order of sequence lengths used in Experiments 1 and 2 was creating a significant confound between order of performance and sequence length. However, since the prior CFR and RFR results replicated in this experiment, this manipulation is not discussed further.

6.1.5.1 Cumulative Backward Rehearsal

The organization of CBR trials was the same as CFR and RFR trials. The startup period of each trial was identical. The difference was in the expected response on each subtrial. While in CFR the sequence length increased on each subtrial by adding the new word to the end of the sequence, in CBR the sequence length increased on each subtrial by adding the new word to the beginning of the sequence.

There were four consecutive blocks of the CBR task. The initial number of words presented on each trial stayed constant throughout a block. Startup sets of two, three, four and five words were used. The order of startup lengths was counterbalanced across participants using a Latin square as for CFR and RFR. A block ended after a trial when a total of 40 subtrials or 10 trials were completed, whichever came first.

The words in the startup set were selected randomly without replacement from the word set. The word to present on each subtrial was selected randomly from those words in the word set not in the current sequence to be recalled. Thus, no word ever occurred more than once in a single recall sequence.

At the beginning of each trial, a message was visually presented indicating the number of words in the startup sequence. One second into this presentation, the number of words was also presented auditorily. After the visual message was presented for 3 s, it was removed and replaced with a 1.3-s fixation. The trial then continued with a startup period followed by zero or more subtrials (see Figure 3.1) followed by feedback.

In the startup period, the initial sequence of words was auditorily presented with 100-ms inter-stimulus intervals. Then, 100 ms after the last word of the startup sequence, a 100-ms tone was played to cue recall. The participant tried to recall the startup sequence. They were instructed to begin rehearsal as soon as they could and to recite quickly and accurately. The start time of their articulation was recorded by a voice key, and the finish time and accuracy were recorded by the experimenter pressing one of two keys. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played and the first subtrial began after a 300-ms response-stimulus interval.

On each subtrial, a word was presented auditorily, and the subject had to respond as soon as possible by quickly and accurately reciting an updated sequence. This sequence consisted of the new word followed by the previous sequence. As during the startup period, reaction time, articulation time, and accuracy were recorded. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played and the next subtrial began after another 300-ms RSI.

A trial ended after an error was made, or after the ninth subtrial. Visual feedback was then provided for 4 s, informing the participant of the correct response, the number of subtrials completed, average reaction time, average articulation time per word, and the number of points they earned on the trial.

To encourage effortful performance, participants earned points based on reaction time, articulation time and accuracy. A monetary bonus was paid based on the number of points earned. On each subtrial, the maximum points that could be earned were the number of words in the correct response multiplied by 5000. No points were earned for a subtrial if an error was made. For a correct subtrial, one point was subtracted from the maximum for each millisecond of articulation time and reaction time. Participants were paid a bonus of one dollar for every 1,500,000 points earned.

6.1.5.2 Rolling Backward Rehearsal

The organization of RBR trials was the same as for the other rehearsal tasks. The startup period of each trial was identical. The difference was in the expected response on each subtrial. While in RFR the first word in the previous sequence was dropped and the new word was added at the end, in RBR the last word in the previous sequence was dropped and the new word was added to the beginning.

There were four consecutive blocks of the RBR task. The initial number of words presented on each trial stayed constant throughout a block. Startup sets of two, three, four and five words were used. The order of startup lengths was counterbalanced across participants using a Latin square as for the other rehearsal tasks. A block ended after a trial when a total of 40 subtrials or 10 trials were completed, whichever came first.

The words in the startup set were selected randomly without replacement from the word set. The word to present on each subtrial was selected randomly from those words in the word set not in the current sequence to be recalled. Thus, no word ever occurred more than once in a single recall sequence.

At the beginning of each trial, a message was visually presented indicating the number of words in the startup sequence. One second into this presentation, the number of words was also presented auditorily. After the visual message was presented for 3 s, it was removed and replaced with a 1.3-ms fixation. The trial then continued with a startup period followed by zero or more subtrials (see Figure 3.1) followed by feedback.

In the startup period, the initial sequence of words was auditorily presented with 100-ms inter-stimulus intervals. Then, 100 ms after the last word of the startup sequence, a 100-ms tone was played to cue recall. The participant tried to recall the startup sequence. They were instructed to begin rehearsal as soon as they could and to recite quickly and accurately. The start time of their articulation was recorded by a voice key, and the finish time and accuracy were recorded by the experimenter pressing one of two keys. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played and the first subtrial began after a 300-ms response-stimulus interval.

On each subtrial, a word was presented auditorily, and the subject had to respond as soon as possible by quickly and accurately reciting an updated sequence. This sequence consisted of the new word followed by the previous sequence less its last word. As during the startup period, reaction time, articulation time, and accuracy were recorded. If the rehearsal was incorrect, then an error signal was played and the trial ended with feedback. If it was correct, then a confirmation tone was played and the next subtrial began after another 300-ms RSI.

A trial ended after an error was made, or after the ninth subtrial. Visual feedback was then provided for 4 s, informing the participant of the correct response, the number of subtrials completed, average reaction time, average articulation time per word, and the number of points they earned on the trial.

To encourage effortful performance, participants earned points based on reaction time, articulation time and accuracy. A monetary bonus was paid based on the number of points earned. On each subtrial, the maximum points that could be earned were the number of words in the correct response multiplied by 5000. No points were earned for a subtrial if an error was made. For a correct subtrial, one point was subtracted from the maximum for each millisecond of articulation time and reaction time. Participants were paid a bonus of one dollar for every 1,500,000 points earned.

6.1.6 Data Analysis

The data analysis was the same as in Experiment 2.

6.2 Experiment 4 Method

6.2.1 Participants

The participants were 12 undergraduate students at the University of Michigan with normal cognitive function. They were between the ages of 18 and 30, right-handed and spoke English as their first language. They were paid eight dollars per hour for their participation in 2 sessions of approximately 2 hours each on separate days, and earned additional monetary bonuses based on performance that typically amounted to a total of five to ten dollars.

6.2.2 Apparatus

The same apparatus was used as in Experiments 2 and 3.

6.2.3 Materials

Two experimental sets and one practice of 10 words each were used. The experimental sets used were Word Set 1 and Word Set 3 (see Table 4.1).

6.2.4 Design

Participants were tested individually with three procedures: articulatory duration (AD), cumulative backward rehearsal (CBR), and rolling backward rehearsal (RBR). Testing occurred in two sessions of 1.5 to 2 hours on separate days. On each day, one rehearsal task was used. First, the rehearsal task was performed with the practice set, and then AD was performed followed by the rehearsal task for each of the two experimental word sets. The word sets were used in the same order in both sessions. The order of tasks across days and the order of word sets within days were fully counterbalanced across participants.

6.2.5 Procedures

The AD task was performed in the same way as in the previous experiments. The CBR and RBR tasks were performed in the same way as in Experiment 3, except that the order of blocks was different. In this experiment, the four blocks for each task were always presented in increasing order of startup sequence length from two to five.

6.2.6 Data Analysis

The data analysis was the same as in Experiments 2 and 3.

6.3 Results

The results are arranged by the aims discussed earlier in the chapter. Articulatory duration measurements are reported first for reference. Results from Experiments 3 and 4 are intermixed (but appropriately labeled) and reported where relevant in order to provide a succinct and goal-oriented presentation of the data.

Table 6.1 Experiment 3: Articulatory durations and memory spans per task and group of participants for Word Set 1.

	Participant group		
Task and variable	All	Subset 1	Subset 2
AD			
Duration (ms)	297	329	274
CFR			
Memory span (words)	7.20	7.63	6.89
RFR			
Memory span (words)	4.89	5.14	4.63
CBR			
Memory span (words)	6.12	6.78	5.69
RBR			
Memory span (words)	4.38	4.74	4.08

Table 6.2 Experiment 4: Articulatory durations and memory spans per task and word set.

	Word set	
Task and variable	1 (short)	3 (long)
AD		
Duration (ms)	362	568
CBR		
Memory span (words)	6.33	5.49
RBR		
Memory span (words)	4.69	4.10

6.3.1 Articulatory Duration

For each experiment, articulatory duration measurements for each word set for each participant were aggregated across the two sessions. The mean estimated articulatory durations for each word set are shown in Table 6.1 and Table 6.2. In Experiment 3 there was only one word set and thus no articulatory duration effect to evaluate. In Experiment 4, the mean articulatory duration for Word Set 1 was reliably less than that for Word Set 3, $t(11) = 18.69, p < .001$.

6.3.2 Replication of RFR and CFR Results

The first aim in this chapter was to replicate the Experiment 2 findings for RFR and CFR. Results for RFR and CFR in Experiment 3 were compared. Performance during the startup period was compared to insure that there were no major differences in the initial state before working-memory updating began. For startup accuracy, there was no reliable effect of task in a two-way within-subject ANOVA with task and sequence length as factors, $F(1, 11) = 0.5146, p = .49$. For startup reaction time, there unexpectedly was a reliable effect of task in a two-way within-subject ANOVA with task and sequence length as factors, $F(1, 11) = 6.39, p = .028$. However, the magnitude of this effect, about 31 ms, was small. For startup articulation time, there was a marginally reliable difference between the tasks for *d*, the base duration of articulating a single word, *t*(11) = 2.178, *p* = .052. However, this difference only had a magnitude of about 31 ms. For *a*, the acceleration, the effect of task was not reliable, *t*(11) = 0.82, *p* = .43. The time differences between startup for RFR and CFR are undesirable, but their small size makes them be of marginal importance for present purposes. They may be indicative of minor differences in fatigue in the two conditions, since RFR is significantly harder than CFR.

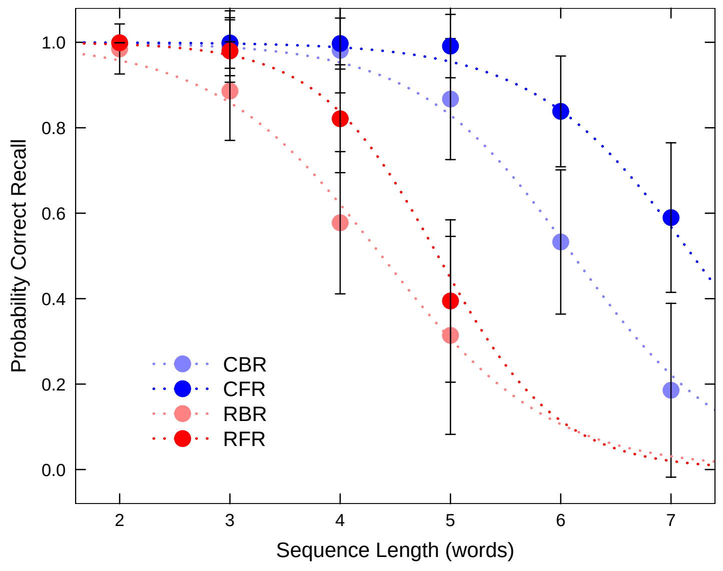


Figure 6.1 Experiment 3 subtrial: Observed mean probabilities of correct recall of the entire sequence and estimated mean logistic regression functions (dotted lines) as a function of task and sequence length.

Performance during updating was then compared by looking at the subtrials for RFR and CFR in Experiment 3. As in Experiment 2, accuracy dropped off at a smaller sequence length for RFR than for CFR (see Figure 6.1). The memory span for RFR was reliably less than the memory span for CFR (see Table 6.1), *t*(11) = 13.81, *p* < .001. Articulation times also followed the same pattern as in Experiment 2, with longer times for RFR at the longer sequence lengths. When the parameters for AT were compared between tasks, the base duration, *d*, was not reliably different, *t*(11) = 1.59, *p* = .14, but the difference for the acceleration, *a*, was reliable, *t*(11) = 2.50, *p* = .029.

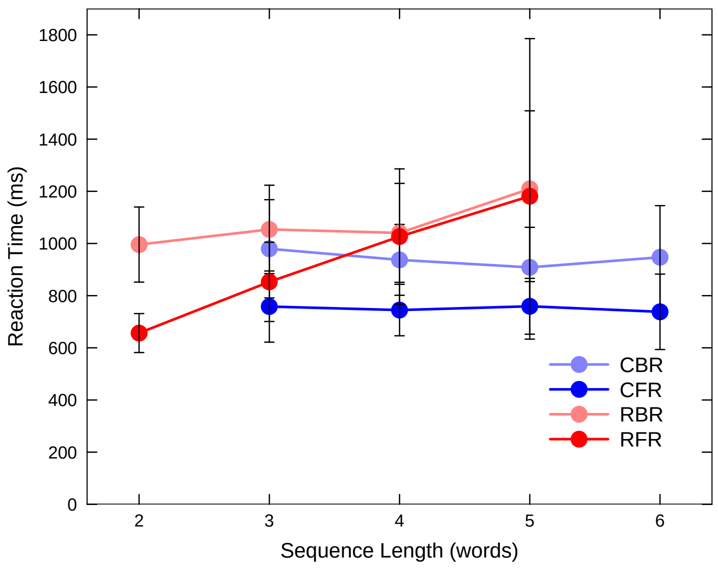


Figure 6.2 Experiment 3 subtrial: Observed mean reaction time as a function of task and sequence length.

The most critical comparisons were done for reaction times. Inspection of the RFR and CFR RTs in Figure 6.2 shows the same striking pattern as in Experiment 2: constant RT across sequence length for CFR, but a linear increase in RT across sequence length for RFR. This was confirmed by linear fits in which the slope for CFR was not reliably different from zero, *t*(11) = 1.11, *p* = .29, but the slope for RFR was, *t*(11) = 6.57, *p* <.001.

The same qualitative patterns as in Experiment 2 were found for the serial response functions (see Figure 6.24), response-type functions, and the word-set error functions for RFR and CFR. Overall, the pattern of observed data for CFR and RFR in Experiment 3 was highly consistent with the previous findings. This reconfirms that the systematic aspects of performance seen in Experiment 2 should be taken seriously. It also provides a clean base of comparison for the new working memory updating operations by showing that the participants who performed these new tasks in Experiment 3 performed the already studied operations in the same way as past participants did.

6.3.3 Comparison of Append and Prepend Operations

A second aim of this chapter was to explore the updating operation of prepending. By comparing append and prepend operations, different ideas about the processes and representations underlying updating could be compared. The first step was to confirm that startup performance was similar for CBR and CFR in Experiment 3. A comparison of recall accuracies found no reliable effect of task in a two-way within-subject ANOVA with task and sequence length as factors, $F(1, 11) = 0.117$, $p = .74$. Likewise, no reliable differences between tasks were found for the articulation time parameters of base duration, $t(11) = 1.91$, $p = .083$, or acceleration, $t(11) = 0.914$, $p = .38$. Finally, there was a small but reliable effect of task of about 36 ms found for reaction time in a two-way within-subject ANOVA, $F(1, 11) = 4.97$, $p = .048$. This difference indicates that participants were not necessarily in the identical mental state just before beginning updating for CFR and CBR. However, as argued before for RFR, the small size of this difference in the face of the overall similarity of startup performance makes it less worrisome. In particular, the fact that accuracy was not measurably impacted by this difference suggests that the underlying memory representations and processes were largely equivalent.

Subtrial performance, where working-memory updating occurred, was considered next. Again, CBR was compared to CFR in Experiment 3. In terms of accuracy, performance dropped off at a lower sequence length for CBR than for CFR (see Figure 6.1). This was confirmed by a comparison of memory spans (see Table 6.1) showing that CBR MSP was reliably less than CFR MSP, $t(11) = 7.58$, $p < .001$. In other words, under conditions that were otherwise identical, participants had a harder time prepending words to sequences than appending words to sequences. This could potentially suggest that prepending is fundamentally different from appending. However, by looking at the Experiment 4 subtrial accuracies for CBR (see Figure 6.3), it is clear that the articulatory loop was still playing a central role in determining performance. As found previously for CFR, there was a substantial and reliable effect of articulatory duration on memory span for CBR, with the one-syllable words of Word Set 1 producing a significantly greater span than the three-syllable words of Word Set 3, $t(11) = 6.74$, $p < .001$.

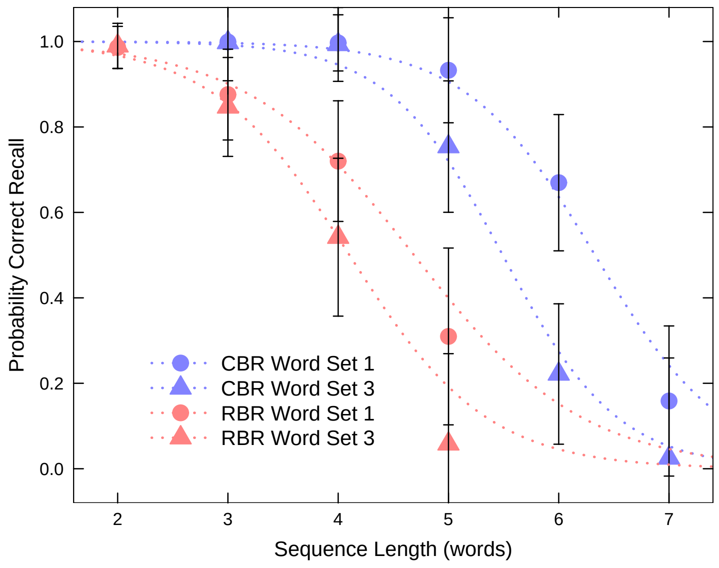


Figure 6.3 Experiment 4 subtrial: Observed mean probabilities of correct recall of the entire sequence and estimated mean logistic regression functions (dotted lines) as a function of task, word set, and sequence length.

Further insight is gained by looking at CBR and CFR reaction times in Experiment 3 (see Figure 6.2). The two most salient features are that both CFR and CBR RTs are approximately flat across sequence length, and that CBR is consistently slower than CFR. This was confirmed by a two-way within-subject ANOVA with task and sequence length as factors, which found that CBR RTs were reliably longer than CFR RTs, F(1,

11) = 25.2, $p < .001$, but there were no reliable differences with sequence length, $F(3, 33) = 0.994$, $p = .41$. This is consistent with the perceptual-motor account since the updating related processes that occur before articulation begins do not change with sequence length. Reasons for why RTs are consistently longer for CBR are addressed later during modeling.

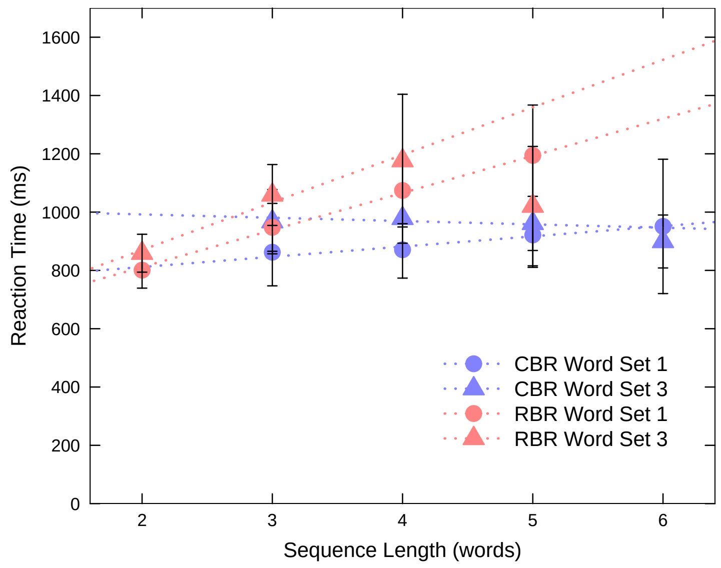


Figure 6.4 Experiment 4 subtrial: Observed mean reaction time as a function of task, word set, and sequence length with linear fits (dotted lines).

The Experiment 4 subtrial RTs for CBR provide another pattern supporting the claim that similar processes are at work in appending and deleting operations (see Figure 6.4). As found for CFR in previous experiments, the CBR RT slopes are not reliably different from zero for either Word Set 1, $t(11) = 1.44$, $p = .18$, or Word Set 3, $t(11) = .999$, $p = .34$, in CBR. But there is a reliable effect of articulatory duration on the intercepts, with the longer words leading to slower responses, $t(11) = 5.37$, $p <.001$.

The last primary measures of performance to consider are the subtrial articulation times. In Experiment 3, there were no reliable differences between CFR and CBR for base duration, d, $t(11) = 1.31$, $p = .22$, or acceleration, a, $t(11) = .344$, $p = .74$. This indicates that, unlike for RFR where recall was produced in a more conservative manner than for CFR, CBR recall was very similar to CFR recall. This supports the idea that performance differences between appending and prepending are not just side effects of differences in motor control, but are more closely linked to the representations and processes of working-memory updating. This will be pursued further when modeling CBR.

Finally, for CBR in Experiment 4, as for CFR in Experiment 2, articulatory duration had a reliable effect on subtrial articulation times that manifested in the base duration, $t(11) = 7.44$, $p < .001$, but not the acceleration, $t(11) = 1.65$, $p = .13$, of the curve fits.

In summary, the performance of prepending words to a sequence is qualitatively very similar to appending words to a sequence. This is the case in terms of the impact of both sequence length and articulatory duration on performance. This suggests that the perceptual-motor account of working-memory updating performance may account successfully for the observed behavior. However, there are specific quantitative differences between performance in CFR and CBR, both in terms of lower accuracies and slower reaction times, which require further interpretation through modeling of CBR.

6.3.4 Comparison of Deleting from the Beginning and End of Sequences

In this subsection, the operations of dropping a word from the beginning and end of a sequence are compared in the context of the RFR and RBR tasks, where appending and prepending are also occurring respectively. This is theoretically interesting because it provides a test of the concept of non-selective updating which was originally motivated by the RFR results in Experiment 2.

Performance during the startup period was compared between RFR and RBR in Experiment 3. In two-way within-subject ANOVAs with sequence length and task as factors, the differences in accuracy, $F(1, 11) = 1.78$, $p = .21$, reaction time, $F(1, 11) = .0005$, $p = .98$, and articulation time, $F(1, 11) = .0494$, $p = .83$, were not reliable. Performance during the startup period was very similar in the two tasks, so differences during subtrials can be taken seriously as indicators of differences in working-memory updating, not differences in initial state.

As discussed at the start of this chapter, a critical prediction for RBR performance, based on the idea of non-selective sequential updating operations, is that subtrial reaction times should increase linearly with sequence length, as opposed to being constant across sequence length. However, an initial inspection of observed mean RTs for RBR shows a non-linear pattern that starts out flat across sequence lengths two, three, and four and then rises at sequence length five (see Figure 6.2). While this pattern could stem from a unique new process, there are other possible explanations.

One concern is that, particularly for some of the poorer performing participants, fewer correct updating trials were available for RBR than for the other tasks. This was due to RBR being the hardest of the four tasks, as judged by memory span (see Table 6.1) and by informal report from the experimenters. As a measure of the relative stability of the reaction times across participants, I found the standard deviation of the individual participant slopes from the linear fits of subtrial reaction time for each of the four tasks. The values were 35.8 ms for CFR, 72.4 ms for RFR, 63.2 ms for CBR, and 155 ms for RBR. The RTs for RBR stood out as being considerably more variable than those for any of the other tasks. Upon inspection of individual participant data, I found that while reaction times were quite qualitatively consistent across participants for CFR, RFR, and CBR, they were highly variable for RBR. From this, I concluded that the non-linear pattern seen in the aggregate is not representative of individual participants, but is rather an artifact of averaging over qualitatively different subsets of data and performance strategies. This averaging across strategies violates Newell's (1973b) second injunction of psychological experimentation to "never average over methods."

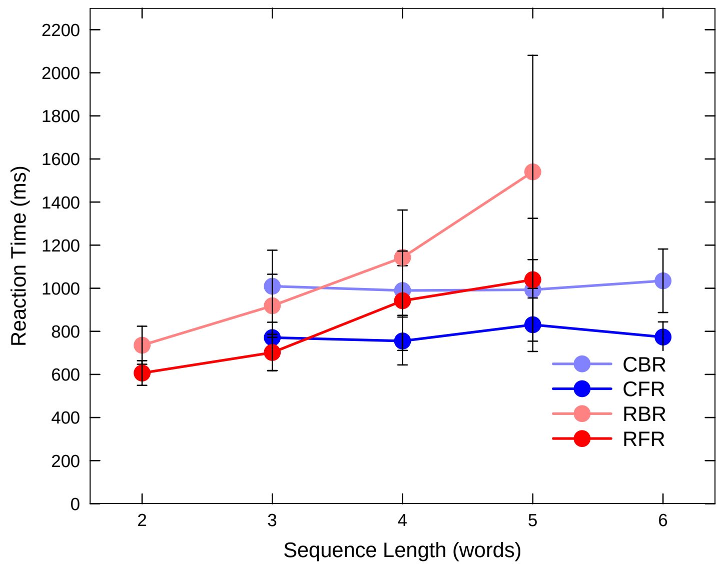


Figure 6.5 First subset of participants. Experiment 3 subtrial: Observed mean reaction time as a function of task and sequence length with linear fits (dotted lines).

Consequently, I split the participants into two subsets. The first subset consisted of five participants whose RT data exhibited a notable positive slope. The second subset consisted of the remaining seven participants, whose data are perhaps best described as highly variable and inconsistent. The subtrial reaction times for each subset of participants are shown in Figure 6.5 and Figure 6.6 respectively. The pattern of RBR RTs for the first subset is obviously linearly increasing, since the participants were selected on that criterion. The RBR slope is also reliably greater than the RFR slope for these participants, $t(4) = 2.91$, $p = .044$. The pattern for the second subset is perhaps inversely sloped, but given the variability within this subset, this slope is not necessarily interpretable. The RT pattern for the other three tasks is qualitatively very similar for the two groups. This is indicative of how much more stable the results for the other tasks were across participants.

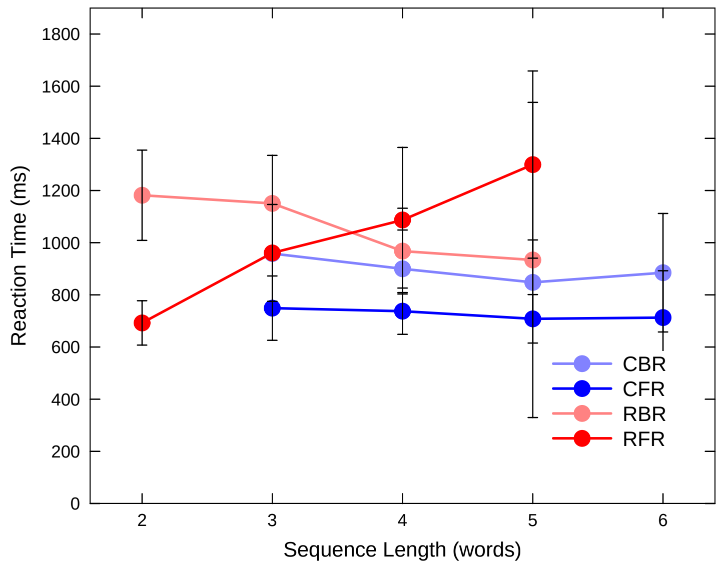


Figure 6.6 Second subset of participants. Experiment 3 subtrial: Observed mean reaction time as a function of task and sequence length.

In Experiment 3, it appears that a subset of the participants is exhibiting the predicted linear increase in RT with sequence length, but for the rest of the participants this is not the case. For further insight into RBR subtrial reaction times, Experiment 4 provided another set of participants. The observed RTs for this experiment were somewhat more consistent (see Figure 6.4). Out of 24 combinations of participant and word set, the slopes of the RBR subtrial RTs were positive for 21, and the mean slope was reliably greater than zero, $t(11) = 2.443$, $p = .033$. However, as in Experiment 3, the RBR slopes were still much more variable across participants than were the CBR slopes (with a standard deviation of 209.6 ms compared to 68.9 ms). And in the mean data, the monotonic increase in RBR RT for Word Set 3 was violated at sequence length five. As was the case for RFR in Experiment 2, neither the slopes, $t(11) = 1.17$, $p = .27$, nor the intercepts, $t(11) = .199$, $p = .85$, differed reliably across word set for RBR in Experiment 4.

Based on the fact that the results for the first subset of Experiment 3 participants were reasonably consistent with the results from Experiment 4, I decided that for subsequent analysis and modeling of Experiment 3 data, I would look only at these participants. This decision was also motivated by the idea that it would be better to model the data from a subset of participants who were using roughly the same strategy, than to develop a model based on a composite of varying strategies. Clearly, the shortcoming of this approach is that it does not account for the performance of a substantial number of other participants in Experiment 3 who are performing the task in a different manner. Future work will be needed to address the performance of these other participants.

The memory spans for each task for each subset of participants are listed in Table 6.1. Subtrial accuracies were compared between RBR and RFR in Experiment 3 for the subset of participants with linear increases in RT slope for RBR. Memory span was not reliably different between the tasks, $t(4) = 1.05$, $p = .35$. However, the data show a trend of separation in performance between tasks, and this is supported by a reliable task by sequence length interaction in accuracy, $F(3, 12) = 5.81$, $p = .011$. For RBR in Experiment 4, memory span for the shorter words was reliably greater than for the longer words, $t(11) = 4.15$, $p = .0016$. This effect of articulatory duration mirrored the one found previously for RFR.

Fits to the articulation times for the subset of participants with linear increases in RT slope did not differ reliably between RBR and RFR in Experiment 3 for either base duration, $t(4) = 1.93$, $p = .13$, or amplification factor, $t(4) = 1.46$, $p = .22$. This suggests that recall for RBR was performed in the same conservative manner as for RFR. In Experiment 4, as expected, the three syllable words had reliably longer base durations than the shorter words, $t(11) = 16.6$, $p < .001$, but the amplification factors were very similar, $t(11) = .0699$, $p = .95$. Again, this was consistent with previous findings for RFR.

Considering all the participants, the RBR data from Experiments 3 and 4 were equivocal about the degree to which dropping words from the end of a list is similar to dropping them from the beginning. However, for at least a subset of the participants, the data are quite consistent with the RBR task being performed as predicted by non-selective updating and the perceptual-motor account. For this subset of participants, the effects of sequence length and word set on performance were qualitatively very similar for RFR and RBR. The ability of the current theoretical approach to account for the RBR data will be pursued further in modeling work discussed later.

6.3.5 Removing Words from the End of Sequences

Another aim of this chapter was to provide a different context in which to evaluate further empirical evidence about the active or passive nature of deletion in verbal working memory. This was originally considered for RFR. Here it is considered for RBR.

The word-set error function for RBR in Experiment 4 is shown in Figure 6.7. The equivalent functions for both subsets of participants in Experiment 3 showed very similar patterns. Intrusions were rare overall, but when they occurred, they were most likely to originate from the most recently dropped word on the current trial. The likelihood of a dropped word intruding fell precipitously with the number of subtrials since it was dropped. Intrusions by words from the current word set that were not presented on the current trial were quite rare.

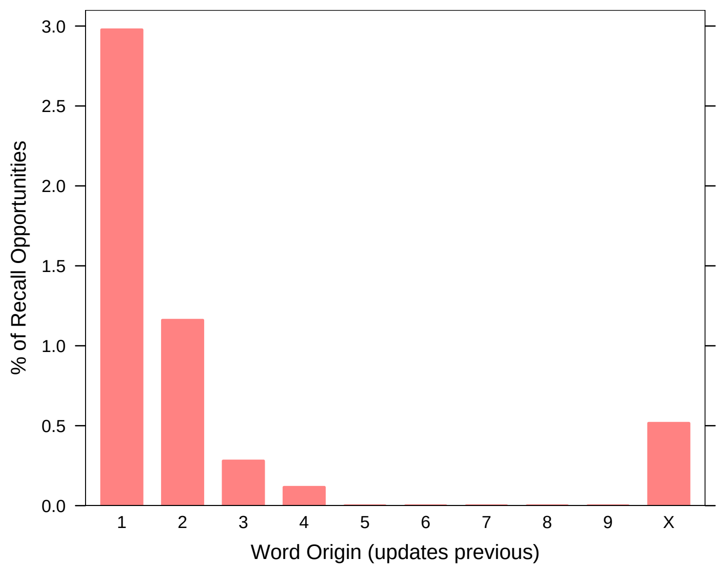


Figure 6.7 Experiment 4 RBR: Word set error function averaged over short and long word sets. The rightmost bar, labeled 'X,' indicates errors where the erroneous word was not in the recall sequence earlier in the trial.

This function is strikingly similar to the one from the equivalent analysis for RFR (see Figure 5.5). Since the status of the words is the same after they have been deleted in the two situations, it is likely that they were moved to this state by the same process. This similarity strongly supports the idea that the mechanism of deletion is the same for words dropped from the beginning and end of a sequence. The argument against inhibition applies here as it did for RFR. If words are deleted by active inhibition, then the most recently dropped word should not be the most likely to intrude, because it will have the strongest level of inhibition applied to it. On the other hand, these results are consistent with deletion by passive neglect, where the most recently dropped words are the least likely to have decayed away.

The most likely intrusions during RFR were words that had just been at the start of the sequence, while the most likely intrusions during RBR had just been at the end of the sequence. This symmetry is particularly interesting given that it follows from the number of subtrials since deletion, not the location of the deleted word within the sequence at the time of deletion. This is at odds with any model where strength of representation of the speech content is used for the representation of serial order (e.g., Page & Norris, 1998). In these models, there is a large asymmetry between the strength of representation of the first and last words in a sequence, and thus this symmetry of intrusions would not be predicted.

In addition to considering where intruding words have originated, we can consider where intruding words end up. This is indicated by the ‘Word Set Error’ response-type function shown in the center of Figure 6.8 for Experiment 4. While not shown, the equivalent functions for both subsets of participants in Experiment 3 were very similar. This function is distinctly different for RBR than it was for RFR (see Figure 5.6). For RFR, the middle of the sequence was the most likely position of intrusion, while for RBR the last serial position in the sequence was the most likely place.

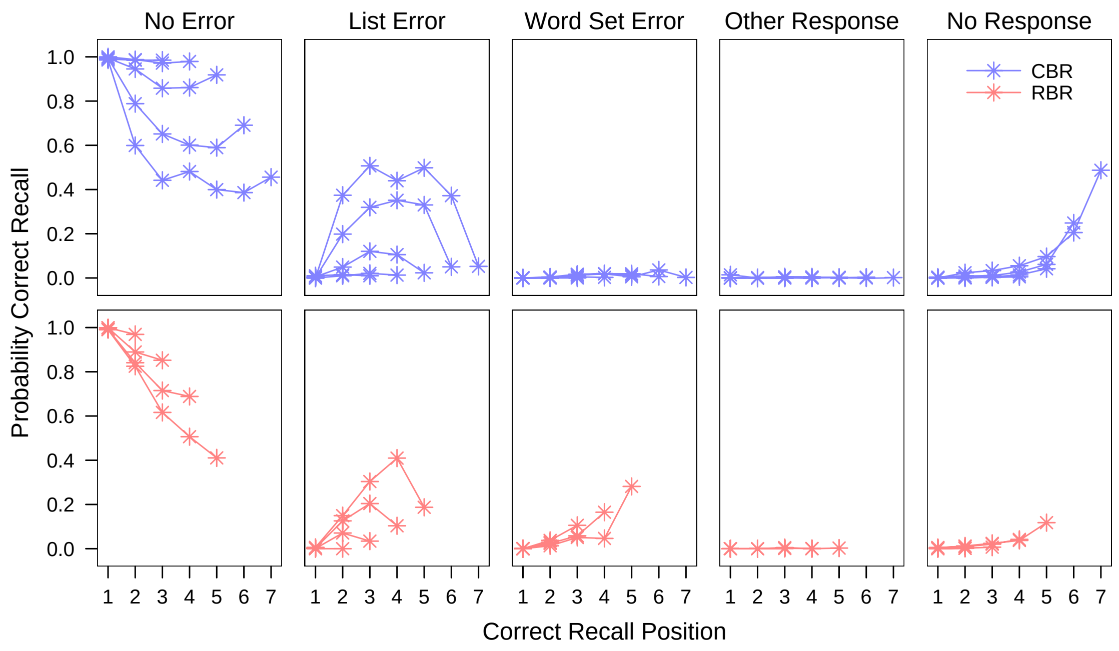


Figure 6.8 Experiment 4 subtrial: Response-type functions for each task.

On its own, this finding for RBR is consistent with either passive neglect or inhibition, but only passive neglect can handle the conjunction of the RBR and RFR findings. With a recovery strategy during recall, intrusions would be expected to appear in the positions where guessing most often takes place. In RBR, since the new word comes first in the sequence, and the previous last item is dropped, guessing will most often take place at the end of the sequence.

6.4 Modeling of CBR

An important aim of this chapter was to evaluate the generalizability of the perceptual-motor account of working-memory updating to new operations. Through the use of the CBR task, the operation of prepending words to a sequence was investigated. As reported earlier, the empirical results for CBR were consistent with the key claim that reaction times on subtrials should be constant across sequence length. However, modeling was necessary to evaluate the consistency of the CBR results with the current account across the full range of dependent variables and conditions. All modeling of Experiment 3 data was done for the subset of participants selected during the analysis of RBR.

As a first step in evaluating model fits to the Experiment 3 data, I ran the previous CFR Link-Decay Model and RFR Truncated-Articulation Model with the articulatory duration values found for participants in this experiment. This was initially done with decay parameters for speech objects and speech links set to the same values used for the best fits in Experiment 2 (speech objects decayed at a median of 6 seconds with a spread of 0.3, and speech links decayed at a median of 3 seconds with a spread of 0.6). However, I found that a longer speech link decay time of 4 seconds provided better fits for both the CFR and RFR models. While values for the other parameters were not modified, this new value was used for the median speech link decay time for modeling all of the tasks. Since relatively small numbers of participants were run in each of the experiments, I think it is reasonable that there could be between-subject differences in these parameters. Individual differences in memory performance may very well be partially due to differences in decay rates between individuals. Refer to Appendix A for a full listing of parameter values, Appendix B for goodness-of-fit-measures, and Appendix C for production rules.

Both the CFR and RFR models were generally successful in replicating the fits to observed data found in Experiment 2. This is not surprising given that the empirical results in Experiment 3 for CFR and RFR were found to be very similar to previous results. The fits for these models are shown in the following figures. For purposes of comparison and evaluation, the CFR fits are displayed with the CBR fits, while the RFR fits are displayed with the RBR fits.

6.4.1 Least-Change Model

I took the same approach to modeling CBR that I did for RFR. I started with the model already established as successful for CFR, the CFR Link-Decay Model, and modified it in the minimal way necessary to make it perform CBR, thus forming the CBR Least-Change Model. It was not necessary to start from the RFR model, because the differences between the RFR model and the CFR model turned on the covert rehearsal necessitated by having two updating operations per subtrial. Since CBR, like CFR, only involves adding a word to the sequence, the CFR model was the appropriate starting point. The modification was constrained only by the architectural features of EPIC.

The changes made to adapt the CFR model to CBR depended only on the different recall requirements during subtrials. The rules for all of the other processes in the model (e.g., performing during the startup period, implementing the recovery strategy, processing incoming overt or external speech, and reacting to the feedback after each subtrial) remained the same. In simple terms, the CBR Least-Change Model first says the newly presented item, and then overtly recalls the entire current recall chain.

For CFR, as soon as the new stimulus is perceived, the model begins recall by detecting the presence of the new speech object in working memory. It then finds the first speech object in the recall chain and retrieves its phonological content so it can be sent to the vocal motor processor for articulation. For CBR, this was changed so that the model detects the presence of the new speech object and immediately retrieves its content for recall. Only after this does the CBR Least-Change Model engage with the current recall chain by locating the first object in the chain.

Once the CBR Least-Change Model starts articulating the recall chain, it proceeds like the CFR model until the end of the chain is reached. At this point, whereas the CFR model located and articulated the new item, the CBR model simply stops recall and waits for feedback.

In order to evaluate the CBR Least-Change Model, it was run with the articulatory duration values found for participants in Experiment 3, and the decay parameters described earlier. The reaction times failed to match the observed data in an important way (see Figure 6.9). Whereas, in the observed data, the reaction times for CBR were significantly longer than those for CFR, the CBR Least-Change Model predicts that CBR reaction times will be slightly shorter than for CFR. This follows directly from the modifications to the beginning of recall. Since recall begins with the newly presented word, no time is spent retrieving the first word in the recall chain before articulation starts. An explanation is needed for this disparity between model and data.

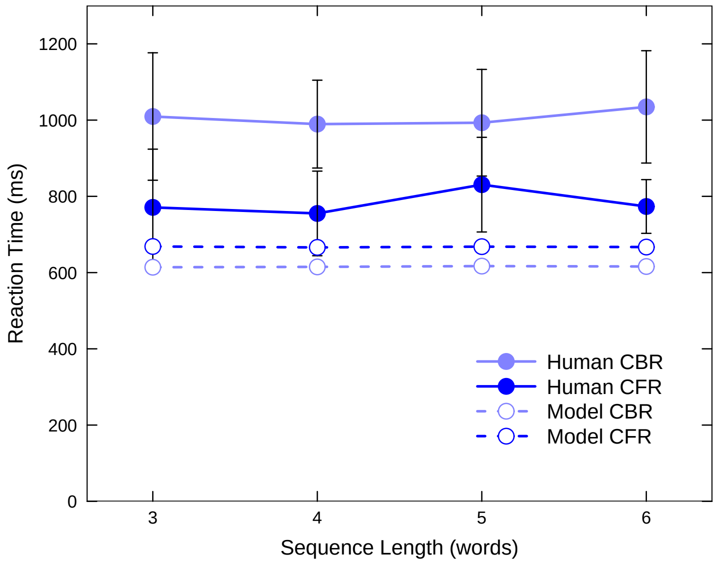


Figure 6.9 Comparison of subtrial RTs for CBR Least-Change Model and Experiment 3 CBR data, and CFR Link-Decay Model and Experiment 3 CFR data. Data are from the subset of participants being modeled.

A candidate reason for this disparity relates to a feature of the auditory perceptual processor that was assumed in the initial modeling of CFR: only fifty percent of an auditorily presented word needs to be processed for that word to enter working memory. This assumption was originally reached based on the reaction times found for CFR in Experiment 1. The RTs were sufficiently short that the only explanation for how production of the first word could begin so soon after the presentation of the new stimulus was that the stimulus was available to production rules before the entire word had been perceptually processed. However, in CFR the newly presented word was never immediately articulated. The CBR data suggest that when the word is going to be articulated immediately, the necessary information is not accessible, or not used, as early as claimed.

The CBR data suggest that the initial interpretation was probably wrong. One possibility is that speech objects are not available as early as claimed, but rather a signal indicating detection of the word is made available in working memory. On this account, recall in CFR is initiated upon detection of the new word, but before it is available in working memory, while in CBR, recall cannot be initiated until the new word is actually available, since it is recalled first. A second possibility is that speech objects are made available as previously claimed, but the full content associated with the speech objects is not available until the entire words have been processed. Another possibility is that the timing of when the object gets used changes strategically with the task. Without further experimentation, it is difficult to disentangle these possibilities. For now, it is sufficient to claim that in CBR, where the word being presented is immediately being produced, the entire word is perceived and processed first.

6.4.2 Postponement Model

Given these considerations, a second model of CBR, the CBR Postponement Model, was created. This model is identical to the CBR Least-Change Model except that, on subtrials, it waits until the presented stimulus is entirely processed before initiating recall. The reaction time predictions of the model for Experiment 3 are shown in Figure 6.10. The difference between the CFR and CBR models in RT is now similar to the difference found in the observed data. However, unexpectedly, the absolute values of the RTs are underpredicted for both tasks. The CFR RTs in Experiment 3 are shorter than those found in previous experiments for identical conditions. The reason for this disparity is not clear.

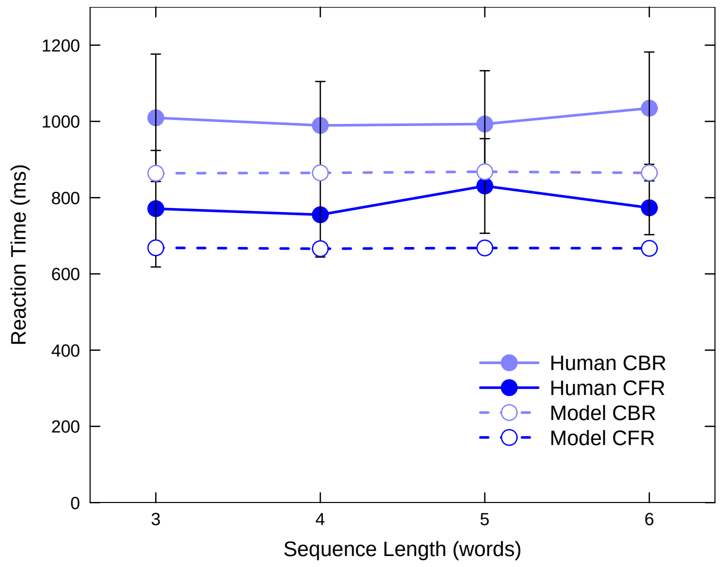


Figure 6.10 Comparison of subtrial RTs for CBR Postponement Model and Experiment 3 CBR data, and CFR Link-Decay Model and Experiment 3 CFR data. Data are from the subset of participants being modeled.

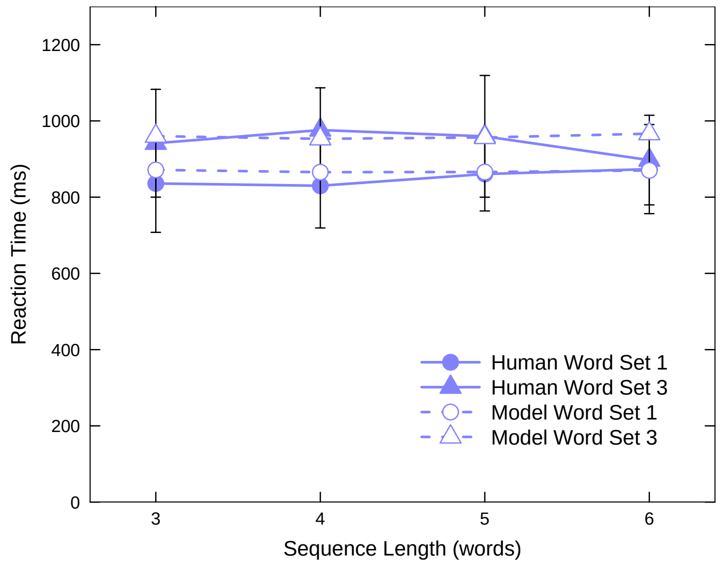


Figure 6.11 Comparison of subtrial RTs for CBR Postponement Model and Experiment 4 CBR data.

In order to check if the underprediction was an anomaly associated with this specific experiment, or a reproducible problem, the CBR Postponement Model was also run for Experiment 4, using both word sets and the associated articulatory durations from that experiment. As can be seen in Figure 6.11, both the absolute magnitude of the RTs and the differences between word sets are well predicted by the model. This suggests that the underpredictions for Experiment 3 may be an artifact of anomalously high observed RTs.

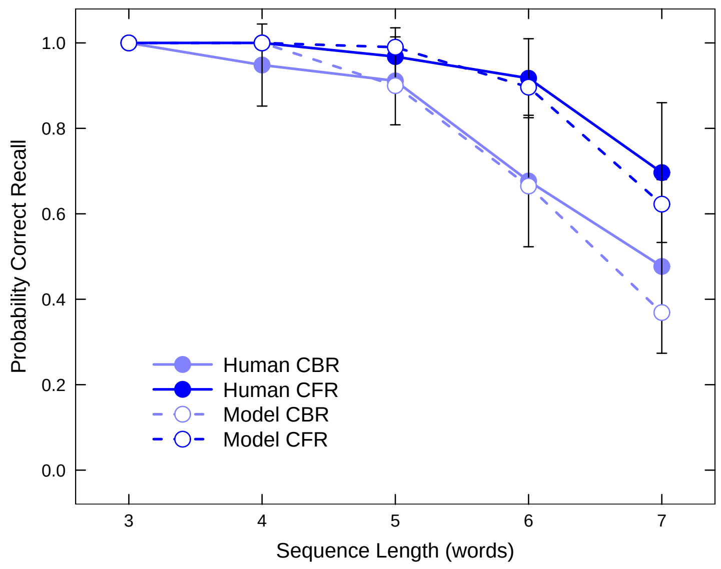


Figure 6.12 Comparison of subtrial accuracies for CBR Postponement Model and Experiment 3 CBR data, and CFR Link-Decay Model and Experiment 3 CFR data. Data are from the subset of participants being modeled.

Figure 6.12 shows the accuracy predictions from the CBR Postponement Model for Experiment 3. One success of these predictions is that the predicted values closely match the observed values for CBR. A second success is that the difference between performance on CFR and CBR is accurately reflected. This is notable because it is not immediately obvious that this would emerge from the differences between the CFR and CBR models. Upon inspection, it is primarily due to differences in the timing of articulation caused by saying the new word before the other words. This leads to an extra word intervening between cycles of articulation for words in the recall chain. For example, for CFR, in moving from a sequence length of three to four, three words intervene between cycles of articulation, while for CBR, four words intervene (see Figure 6.13).

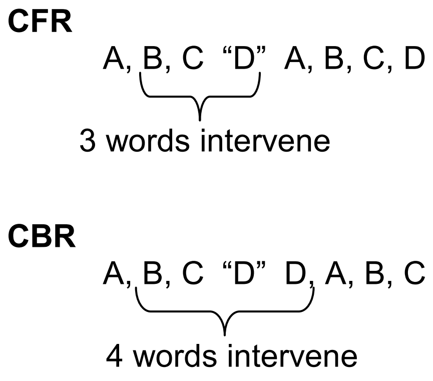


Figure 6.13 Illustration of differences in articulation during CFR and CBR tasks. Different numbers of other words intervene between repetitions. Letters represent words articulated by participant. Quoted letters represent words presented as stimuli. Time moves from left to right.

The CBR Postponement Model captures the effect of word set on accuracy found in Experiment 4 (see Figure 6.14). As was the case for the CFR model, it does this because of the interaction between decay and the time taken for articulation. Articulation times are also predicted accurately for Experiments 3 and 4 (see Figure 6.15 and Figure 6.16). This follows from the fact that articulation times for CFR and CBR were not different in Experiment 3, and the CFR and CBR models perform articulation in the same way once the recall chain is started. Finally, as exhibited in Figure 6.24, the CBR Postponement Model captures a number of features of the serial position functions and response-type functions. This includes the effect of sequence length and some aspects of the effect of serial position. However, the model fails to show the recency effect seen in the 'Position' SPF.

Interestingly, unlike the CFR and RFR models, the CBR Postponement Model captures notable features of the 'Item' SPFs, including the large drop between the first and second serial positions, and the slight recovery across the following serial positions. This is related to the fact that since the new item is said before the recall chain, the chances that the start-chain tag for the recall chain is lost are much higher for CBR than

CFR. This raises the likelihood that the second word to be recalled, which is the first item in the recall chain, will be lost. At the same time, the first item is the word which was just presented, so it is highly unlikely to be lost.

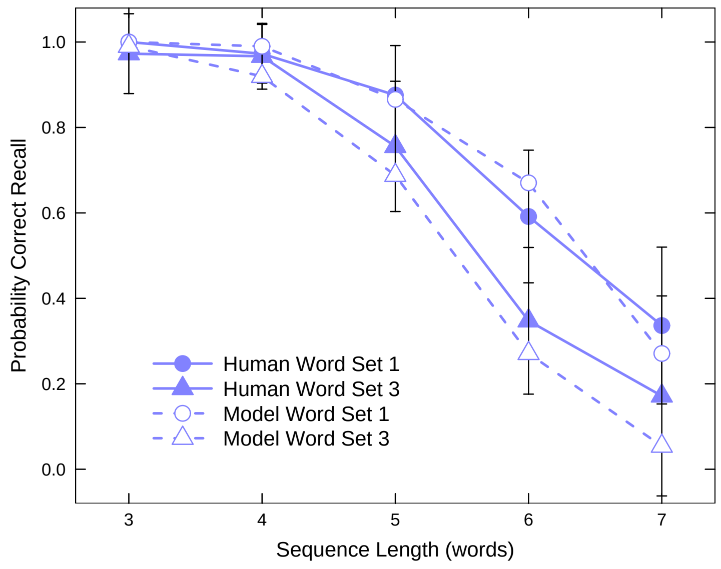


Figure 6.14 Comparison of subtrial accuracies for CBR Postponement Model and Experiment 4 CBR data.

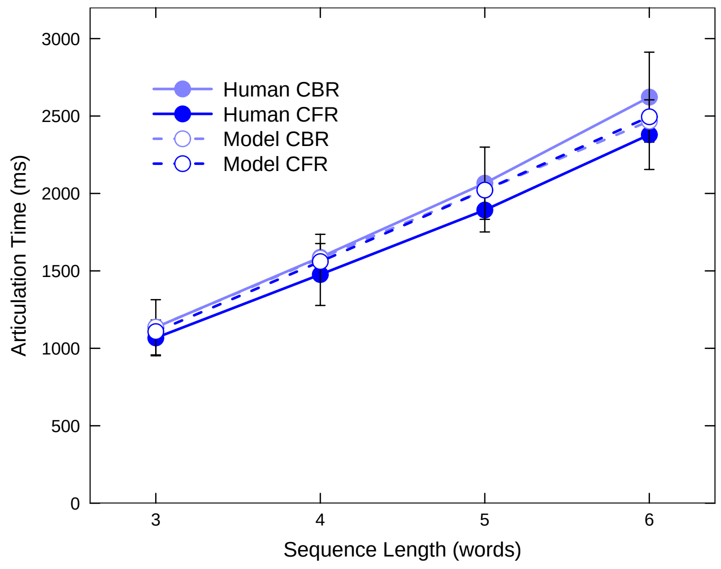


Figure 6.15 Comparison of subtrial ATs for CBR Postponement Model and Experiment 3 CBR data, and CFR Link-Decay Model and Experiment 3 CFR data. Data are from the subset of participants being modeled.

Overall, the CBR Postponement Model successfully predicts many features of the observed data across the full range of dependent variables. This model was generalized from the CFR Link-Decay Model with relatively minor modifications. This may be taken as the successful generalization of the perceptual-motor account to the working-memory updating operation of prepending words to a sequence.

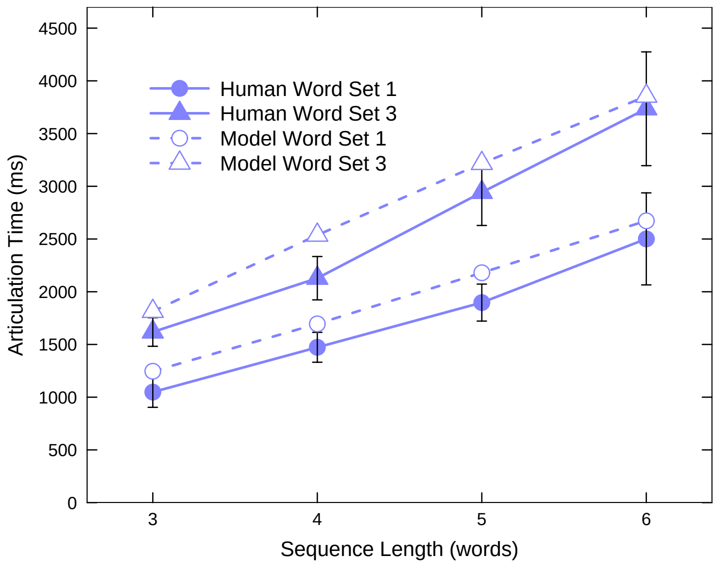


Figure 6.16 Comparison of subtrial ATs for CBR Postponement Model and Experiment 4 CBR data.

6.5 Modeling of RBR

Based on the analysis of RFR in Experiment 2, I proposed that one constraint on working-memory updating performance is that updates are non-selective. As a result, two cycles of rehearsal are required to perform the two operations of appending to a sequence and dropping a word from the beginning of a sequence. The use of the RBR task in Experiments 3 and 4 was intended to see if this hypothesis would generalize to the related situation where a word is prepended to a sequence and a word is dropped from the end of a sequence. The empirical results were mixed, but for at least a subset of participants, the data may be consistent with this hypothesis. In order to more fully evaluate this hypothesis, modeling was done.

6.5.1 Least-Change Model

Following an approach I have used repeatedly, I created an initial model of RBR performance, the RBR Least-Change Model, by making minimal modifications to the RFR Truncated-Rehearsal Model. The changes necessary to adapt the RFR model to RBR were all related to the truncated covert rehearsal and overt recall that take place during subtrials. The rules for implementing all of the other processes in the RBR Least-Change Model remained the same.

Covert rehearsal was changed for the RBR model, because in RFR the first word in the recall chain is skipped and then the rest of the words are articulated, while in RBR it is necessary to include the first word in the recall chain and omit the last word. The same logic was used to select this strategy as was used to select the RFR strategy: given the known constraints, this strategy requires the least number of words to be covertly articulated. In order to begin rehearsal with the first word in the recall chain, it was not skipped over as rehearsal was started. Rehearsal then proceeded in the same way as it did for RFR until the last item in the chain was reached. In RBR, it was necessary to detect the last item in the chain and end rehearsal without saying it. In RFR, rehearsal would end when no more unrehearsed words remained in the recall chain, but in RBR, rehearsal ended when a single item remained.[8]

After rehearsal, the next process was overt recall. In RFR, the model got a head start before the new stimulus was presented by sending the first word in the recall chain

[8] This was actually a more difficult production rule programming problem than most. This may have no theoretical import, but it is interesting to speculate whether this complexity had anything to do with participants' struggles with this task, and the seeming variability in their strategies.

to the vocal motor processor for preparation. This was not possible in RBR, since the first word to be recalled was the new stimulus. Instead the RBR model waits for the presentation of the new stimulus and then recalls it, followed by recalling the new recall chain that was built up from the covertly articulated sequence.

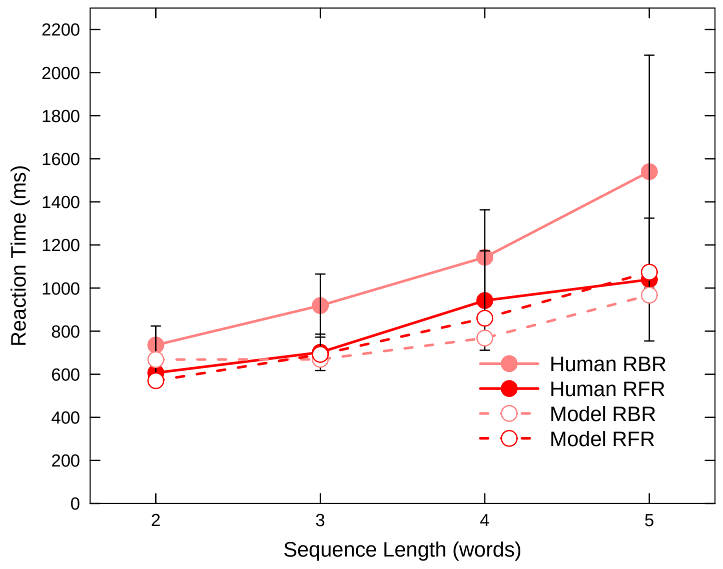


Figure 6.17 Comparison of subtrial RTs for RBR Least-Change Model and Experiment 3 RBR data, and RFR Truncated Rehearsal Model and Experiment 3 RFR data. Data are from the subset of participants being modeled.

The RBR Least-Change Model was evaluated against the RBR results for Experiment 3. Only participants in the subset who displayed an upward slope in RT were used in the comparison data. As discussed earlier, this was done to avoid averaging over strategies. Figure 6.17 shows the reaction time predictions for the RBR Least-Change Model along with the comparable predicted and observed data for RFR. The RFR predictions provide a good fit to the data, but the RBR predictions are far off. The RBR Least-Change Model underpredicts the steepness of the slope.

Since the observed Experiment 3 RTs for CFR and CBR were higher than in other experiments, I investigated this possibility for RBR by running the RBR Least-Change

Model against the Experiment 4 data as well. While the RT slope is in fact lower for Experiment 4 than for Experiment 3, it is still considerably higher than that predicted by the model (see Figure 6.18). As a result, I had to consider alternative explanations.

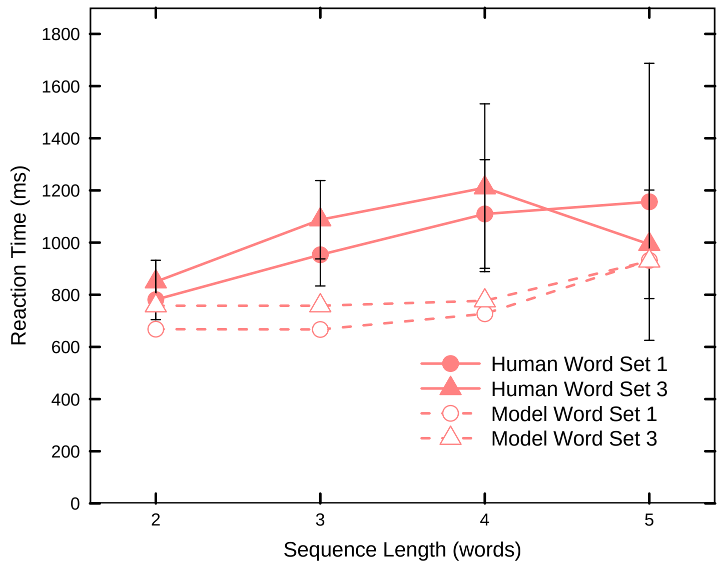


Figure 6.18 Comparison of subtrial RTs for RBR Least-Change Model and Experiment 4 RBR data.

6.5.2 Restrained-Rehearsal Model

A trend emerging across tasks through my modeling is that participants adopt more conservative strategies when dealing with tasks that are harder, less likely to be practiced, and more 'unusual.' For example, in RFR, participants seemed to be using a more conservative approach to overt articulation than they did in CFR. And in CBR, when the new stimulus was presented, participants seemed to be waiting longer before beginning articulation than they did in CFR.

Expanding on this theme, I surmised that participants may have been more conservative in their covert rehearsal during RBR than they were during RFR. One way to implement this in a model of RBR would be to not truncate covert articulation. However with this approach, the RBR RTs would exhibit an articulatory duration effect, but no such effect was found in Experiment 4. Another way to implement this in a model of RBR would be to reduce the pipelining of words to the vocal motor processor, in a way similar to what was done for overt speech in RFR. Given the lack of an articulatory duration effect, and the precedent of the pipelining approach for overt recall in RFR, I choose the latter for further modeling of RBR.

I developed a second model of RBR performance, the RBR Restrained-Rehearsal Model. This model was identical to the previous RBR Least-Change Model, except that the truncated covert rehearsal was not pipelined to the vocal motor processor. Pipelining was reduced in this model such that preparation of the next word did not begin until the articulation of the previous word was complete.

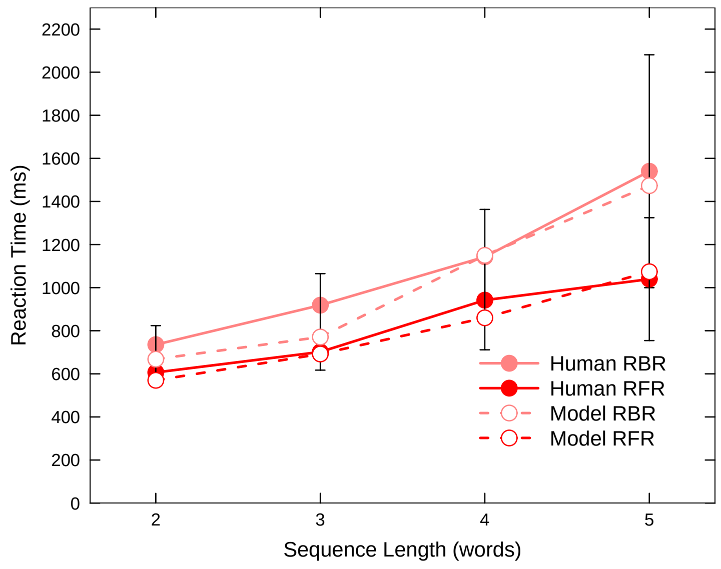


Figure 6.19 Comparison of subtrial RTs for RBR Restrained-Rehearsal Model and Experiment 3 RBR data, and RFR Truncated Rehearsal Model and Experiment 3 RFR data. Data are from the subset of participants being modeled.

The reaction time predictions of the RBR Restrained-Rehearsal Model are compared to the Experiment 3 data in Figure 6.19. The fit is considerably better than for the previous model. However, there is a non-linearity in the predicted RTs which is not representative of the data.[9] This non-linearity obscures the fact that the slope due to the un-pipelined truncated covert rehearsal is too steep. This is more evident when the RBR Restrained-Rehearsal Model is compared to the Experiment 4 data, where it underpredicts at lengths two and three, and overpredicts at lengths four and five (see Figure 6.20).

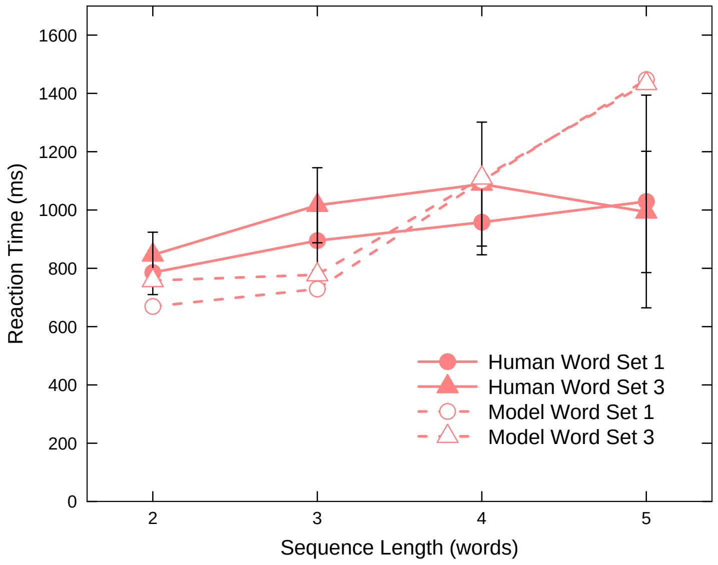


Figure 6.20 Comparison of subtrial RTs for RBR Restrained-Rehearsal Model and Experiment 4 RBR data.

The non-linearity is due to the fact that, at sequence length two, covert rehearsal is not on the critical path for reaction time. At length two, the time to initiate recall after perceiving the new stimulus is the limiting factor. This occurs because the new stimulus is needed for immediate recall, as in CBR, and the covert rehearsal finishes earlier than the initiation of recall. Only at length three and greater does covert rehearsal become the limiting process.

The RBR Restrained-Rehearsal Model predictions for subtrial accuracy from Experiment 3 are shown in Figure 6.21. The accuracies are overpredicted across sequence length for RBR. Furthermore, the model fails to show any separation from the RFR data.

[9] It looks like there might be a non-linearity in the observed data, but in an ANOVA on sequence length with polynomial contrasts only the linear contrast was reliable.

This was unexpected, because the situation is somewhat analogous to the one for CBR and CFR, where a difference in accuracy between tasks was appropriately predicted by the CFR and CBR models. It appears that the putative theoretical covert articulation process in RBR is negating an effect that might otherwise have led to a difference between RFR and RBR accuracy. The RBR Restrained-Rehearsal Model does predict an effect of articulatory duration on accuracy for Experiment 4, but again the accuracies are overpredicted (see Figure 6.22). This is a notable failure of the model.

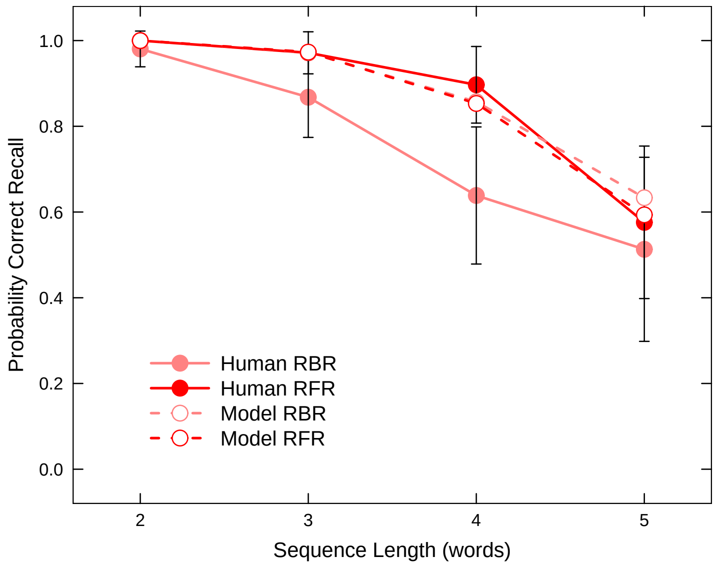


Figure 6.21 Comparison of subtrial accuracies for RBR Restrained-Rehearsal Model and Experiment 3 RBR data, and RFR Truncated Rehearsal Model and Experiment 3 RFR data. Data are from the subset of participants being modeled.

Articulation times were reasonably well predicted by the model, showing an appropriate effect of word set for Experiment 4 (see Figure 6.23). The predicted and observed serial position functions for RBR are very similar for Experiments 3 and 4. Figure 6.24 shows that, for Experiment 3, the ‘Position’ and ‘Relative Order’ serial position functions predicted by the RBR Restrained-Rehearsal Model appropriately show a lack of recency in the last serial position in contrast to the finding for RFR.

Refer to Appendix A for a full listing of parameter values, Appendix B for goodness-of-fit-measures, and Appendix C for production rules, for the RBR Restrained-Rehearsal Model.

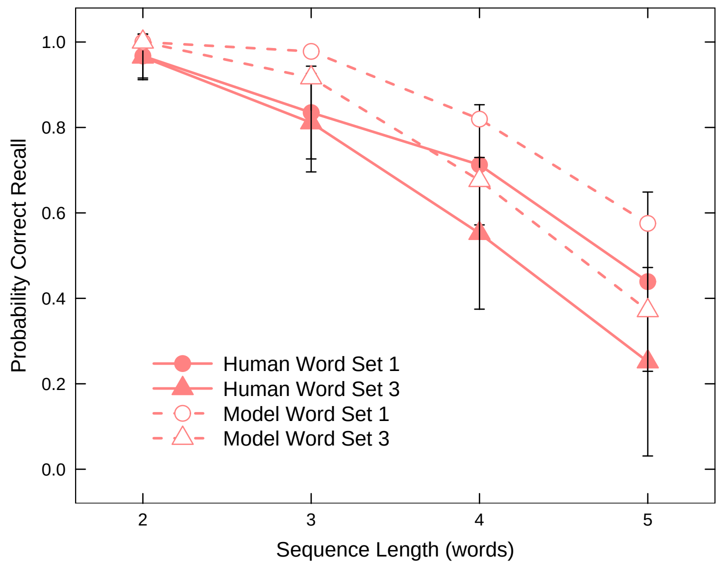


Figure 6.22 Comparison of subtrial accuracies for RBR Restrained-Rehearsal Model and Experiment 4 RBR data.

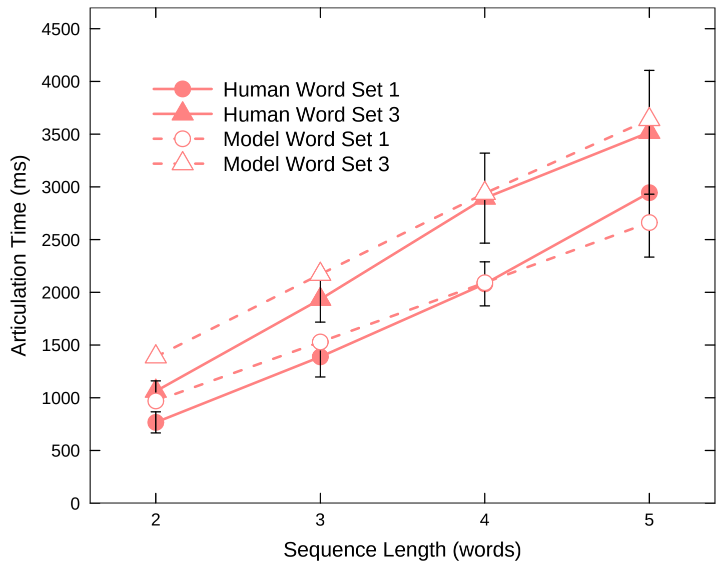


Figure 6.23 Comparison of subtrial ATs for RBR Restrained-Rehearsal Model and Experiment 4 RBR data.

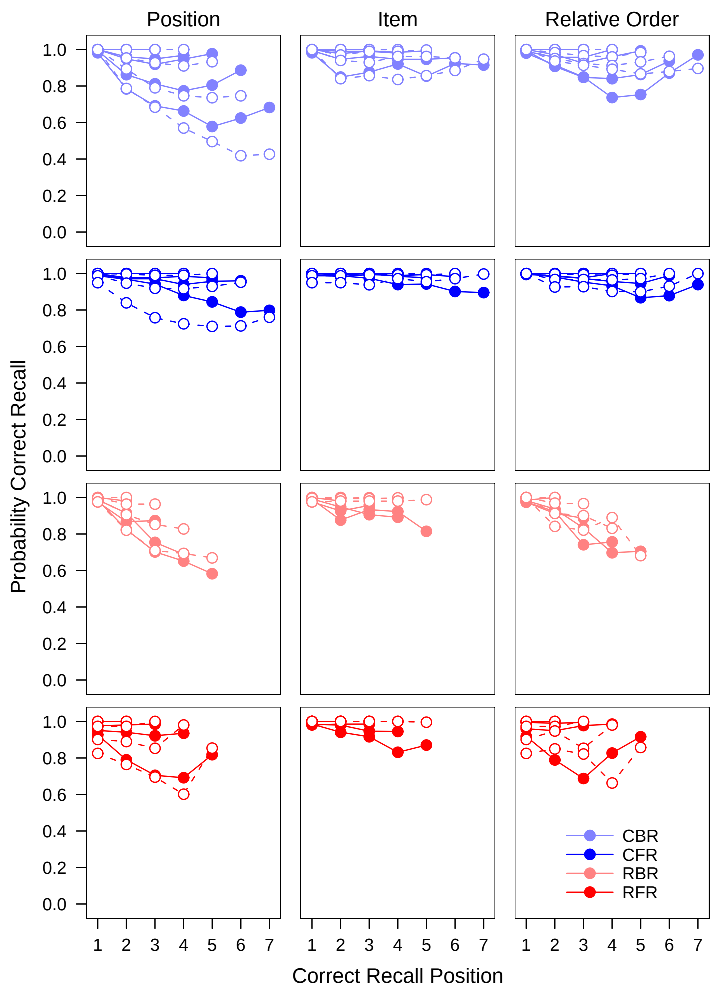


Figure 6.24 Subtrial SPFs for CBR Postponement Model, CFR Link-Decay Model, RBR Restrained-Rehearsal Model, and RFR Truncated-Articulation Model compared with Experiment 3 data. Data are from the subset of participants being modeled.

6.6 Discussion

In this chapter, I reported two experiments and related modeling. My overarching goal was to further evaluate and extend the perceptual-motor account of verbal working-memory updating.

One aim was to replicate findings from the previous chapter about the operations of dropping a word from the beginning of a sequence and appending a word to the end of a sequence. A surprising finding was that when both updating operations were required in RFR, the time to initiate recall increased linearly with the length of the sequence. This result was replicated in Experiment 3. Furthermore, the models developed for CFR and RFR successfully accounted for performance again in Experiment 3. In these models, articulatory and auditory perceptual processes were deployed to implement updating operations on verbal working-memory representations, and updating was non-selective.

A second aim was to compare prepending a word to a sequence with appending a word, and to evaluate whether prepending could also be understood with the current account. I found that performance in CBR with prepending was similar in many ways to performance in CFR with appending. One important example of this was that reaction times for CBR did not depend on the length of the sequence being manipulated. However, there were specific ways in which performance was worse for CBR, including lower accuracies. When the perceptual-motor account was applied to prepending, I found that it accounted for these major features of performance in both Experiments 3 and 4. A key to this success was that serial order is represented by relative links between speech objects. Other models of working memory, in which serial order is represented through slots or ordinal tags, would not be able to explain a prepend operation that yields constant RTs for longer sequences.

A third aim in this chapter was to further evaluate the hypothesis of deletion by passive neglect. I did this in the RBR task by looking at the operation of dropping a word from the end of a sequence while also prepending a word. I found that when words are dropped from the end of a sequence, they are occasionally and incorrectly reinserted into the updated sequence. The most recently dropped words are the most likely to intrude, in a pattern that is consistent with the hypothesis that the words were not actively deleted, but simply left to decay in memory without being renewed through articulation.

A fourth aim was to further evaluate the hypothesis of non-selective updating. According to this hypothesis, each updating operation requires the updated sequence to be rearticulated. In RBR, two updates, a prepend and a deletion, are needed on each subtrial. As a result, the reaction times should increase linearly with sequence length, since a covert cycle of articulation will be required before the overt recall. In Experiment 4, the aggregate fits showed a reliable slope. However, I found this for only a subset of the participants in Experiment 3. This suggested that other participants performed RBR in a currently unspecified manner.

To better understand what was occurring in RBR, the model of RFR was modified to perform RBR. Surprisingly, this RBR model incorrectly predicted a non-linear component to the increase in reaction time with sequence length in addition to the linear component. This could indicate that non-selective updating does not in fact imply linear increases in reaction time in this type of task. However, it could also indicate that this model does not appropriately implement non-selective updating.

The model was not constructed from scratch to perform RBR according to a particular theory. Rather, an existing model of RFR was minimally modified to perform RBR. While this has the benefit of leading to a closely related set of models, in this case it may have led me astray. In the RBR model, covert rehearsal and overt recall are not directly linked. Covert rehearsal is initiated upon feedback about the previous subtrial, while overt recall is initiated by the perception of the new word. The timing of these events is such that at a sequence length of two there is a gap between the end of rehearsal and the start of recall. The correct strategy may do a better job of coordinating events.

In this chapter, the results and modeling for three tasks (CFR, RFR, and CBR) involving three distinct updating operations were consistent with the perceptual-motor account of verbal working-memory updating. Both the results and modeling for a fourth task, RBR, were in some ways more equivocal, and suggest that there is further work to be done.

CHAPTER 7

General Discussion

In this dissertation, I employed a series of experiments and computational models to investigate updating operations in serial verbal working memory. Across four experiments, I analyzed behavior on four new overt rehearsal tasks designed to illuminate the performance of four distinct updating operations. Performance was measured in terms of its temporal characteristics, its accuracy, and the particular types of errors that were made. To understand the mental processes and representations involved, I developed a series of computational models that made predictions about each of these aspects of performance. I started with a limited set of cognitive architectural mechanisms and strategies and extended these from one task and model to the next. This had the benefit of pushing a small set of ideas as far as they could go, but was limited in that the concepts and underlying assumptions fell short at times.

7.1 Verbal Working-Memory Updating Performance

Participants were asked to update ordered sequences of words held in memory in four ways: adding to the end of a sequence (appending), adding to the beginning of a sequence (prepending), dropping from the beginning while also appending (rolling forward), and dropping from the end while also prepending (rolling backward). Performance was best both in terms of accuracy and speed when appending to a sequence. When prepending, participants were less accurate and slower, but qualitatively they performed similarly to when they were appending. In contrast, when participants were rolling forward, they were not only less accurate, but the timing of the initiation of their responses was qualitatively different. Having more words in the sequence, which had not impacted reaction time for appending and prepending, now slowed it. Finally, accuracy was worst when rolling backward. Some participants' timing for rolling backward was qualitatively similar to rolling forward, while for other participants, their timing was more difficult to characterize succinctly.

7.2 Models of Updating

In the following subsections, I discuss the major concepts embodied by the models of updating performance developed here.

7.2.1 Perceptual-Motor Approach

The models developed here show that updating of verbal working memory relies heavily on peripheral processes of auditory perception and vocal motor action. This account does not claim that there is no executive control involved in updating; rather it claims that the control processes which coordinate updating are the same processes that manage storage, maintenance, and recall. Updating is not a distinct elementary process.

This contrasts with an alternative theory that states that "memory updating is not performed by the articulatory loop" (Morris & Jones, 1990, p. 117). According to this view, a subtrial on one of the rehearsal tasks would be performed in two independent steps. First, an updating process would modify the internal representation of the sequence, and then the updated representation would be articulated.

I have shown that these separate steps are not necessary to explain performance. An updated representation is constructed *through* the process of articulation and subsequent perception. Adding a word to a sequence is done by articulating that word in the appropriate place in the sequence, thus generating a new representation with the word included. Deleting a word from a sequence is done by omitting that word during articulation of the sequence, thus generating a new representation without the word included.

This theory explains why auditory perceptual and vocal motor brain regions are active during the n-back task, an updating task, even when the stimuli are presented visually and no overt recall is required (Schumacher et al., 1996). Also explained is why no additional prefrontal brain regions were involved in updating compared to maintenance in a running memory task (Postle et al., 2001).

Apparently, primitive processes of rehearsal are combined to implement more complex operations of updating. Drawing an analogy to computers, this theory is like a reduced instruction set computing (RISC) architecture, where a small number of simple primitive operations are used in combination to perform complex tasks, as compared to a complex instruction set computing (CISC) architecture where there are a large number of more complex primitive operations available.

7.2.2 Episodic Speech Objects and Serial-Order Links

In the models developed here, speech objects bind bits of serial order and phonological content into representations of specific instances of words. These speech objects are episodic because a new object is created each time a word is perceived. When the model hears "Cheese, cheese, cheese," the resulting representation contains three linked speech objects, not one. Episodic working memory crucially interacts with perceptual-motor rehearsal in the implementation of working-memory updating. When a reconfigured sequence of words is articulated, the resulting memory representation has been updated *because* this sequence is perceived and stored as a new episode. The success of the models presented here provides support for the assumption of episodic working memory.

In the models, serial order is represented with links between the transient speech objects. Consistent with Mueller (2002), I concluded that links between speech objects must be able to decay independently of the speech objects. This accounts for the relative abundance of errors where the correct words are recalled in the wrong order. I have expanded the support for this model of serial order by showing that it can account for the way ordered representations are updated as well as maintained.

Using the terminology developed in Section 2.2.2, the models here are fixed-link models. Might other types of models also account for the data? Without launching into a full consideration of serial order representation in working memory (worthy of a thesis of its own), I will point out a few relevant issues. Malleable-slot models and malleable-tag models would both have a hard time accounting for the fact that the time to append and prepend is unchanging with the number of words in the sequence. In one case or the other, every word in the sequence would need to be updated either by moving it to an adjacent slot, or assigning it a different ordinal tag. Fixed-slot models would have a hard time explaining the evidence that order information can be lost independently of phonological information, since the order is implicit in where the information is stored. Malleable-link models would require separate updating operations to act on representations, but as discussed in the previous subsection, the evidence suggests that such additional operations are not needed to account for performance. This leaves fixed-tag models as a viable alternative approach. Further experimentation and modeling will be necessary to distinguish these approaches.

7.2.3 Non-selectivity of Operations

According to the perceptual-motor approach, updating a sequence involves articulation of the sequence. I have additionally found that when task demands require multiple modifications of a sequence, each modification is done in a separate cycle of rehearsal. For example, when participants are asked to roll forward, they first articulate the sequence covertly to drop the initial item before adding the new item during overt recall.

I have conceptualized this as indicating that updating operations are non-selective, in that they each operate on the entire sequence. Kessler and Meiran (2006) found evidence for non-selectivity when participants were changing values in a sequence of numbers in a variant of the Garavan memory updating task. As far as I know, the experiments reported here are the first to provide evidence of non-selectivity when words are being added and deleted from a sequence.

At this point it is not known whether the non-selectivity of updating is an unyielding architectural constraint, or a strategic choice that could be modified through training and practice. An analogy can be drawn to the debate about the ostensible "response selection bottleneck" in multiple task performance and whether it is architectural or strategic (e.g., Meyer & Kieras, 1997). The fact that some participants in Experiment 3 did not show the expected pattern of performance may indicate that non-selectivity is not architectural, but this is speculative, since it has not yet been determined how they performed the task.

7.2.4 Deletion by Passive Neglect

Past research has disagreed about whether deleting a word from a sequence involves an active process of inhibition or just passive neglect (e.g., De Beni and Palladino

(2004) and Hartman et al. (2001) versus Palladino et al. (2001) and Ruiz et al. (2005)). A corollary to the perceptual-motor approach is that deletion of a word from a sequence is performed by leaving it out of a cycle of articulation. The word does not occur in the new representation, and the old representation eventually decays. This is a version of passive neglect. As expected from this account, the likelihood that a deleted word intrudes into recall is greatest immediately after deletion, and declines rapidly over time. Passive neglect is sufficient to account for deletion from serial verbal working memory in the tasks studied here.

7.2.5 Importance of Strategy

An emphasis on the importance of strategy in performance is surely not a new or unique idea (e.g., Meyer & Kieras, 1997; Mueller, 2002; Newell, 1973b). However, it is still worth saying a few words about this in relation to the current work. Strategy will reach further down into performance when the primitives are finer grained. According to the perceptual-motor approach, a few primitive operations of articulation and perception are combined to implement different updating operations. This makes strategy more important since it comes into play in even the simplest circumstances. As a result, it is challenging to successfully predict performance in a new task in a truly a priori fashion because there are so many strategic possibilities available. This applies even in basic tasks like those studied here.

One way to mitigate this effect is to develop heuristics that accompany the more tightly specified constraints of theory. In the present work, one promising heuristic is that more difficult tasks are performed in a more conservative fashion. Overt recall was performed in a less pipelined manner in the two tasks that required deletion as well as addition. Response processes started later after presentation of the new word in the tasks that required prepending instead of appending. And covert articulation may have been less pipelined when rolling backward rather than forward. Interestingly, given the important role of time in these tasks because of decay processes, this conservatism may have actually contributed to lower performance in tasks perceived by the participant as more difficult.

By studying such trends, it may be possible to anticipate them, or at least understand them as systematic and not arbitrary.

7.2.6 Avoiding Overfitting of Models

Overfitting is a legitimate concern with any model-fitting enterprise. Unreliable or meaningless aspects of the data may be taken into account in a model. Also, there may be so many mechanisms and parameters in a model that it fits every nuance of a particular data set, and yet fails to generalize in a meaningful way. I have worked to avoid these pitfalls by attending to both the modeling approach and the observed data.

I have used statistical tests to determine which features of the observed data are reliable. This has included testing the reliability of differences in accuracy, reaction time, and articulation time across tasks, word sets, and sequence lengths. In addition, I have sought to replicate major findings by using each rehearsal task in multiple experiments run with different groups of participants.

In my modeling of working-memory updating, I limited the architecture, parameters, and strategy. I used a circumscribed architecture based on EPIC. Working-memory updating was accomplished through perceptual-motor rehearsal processes. Information about words and serial order was lost only through decay. I also limited the amount of parameter fitting. Only a few decay parameters were varied. For a given experiment, even these parameters were generally kept constant across tasks. The table of parameter values (Table A.3 in Appendix A) shows how little parameters were manipulated in model fitting.

Perhaps the greatest danger of overfitting is in the selection of strategy as realized in the production rules for a model. I sought to minimize this danger by taking a disciplined approach to model construction. I started with the simplest feasible model for a single task, CFR, and modified it in the minimal way to address major reliable aspects of performance. In moving from task to task, instead of starting from scratch each time with a new ad-hoc model, I started with an extant model from another task, and minimally modified it to perform the new task. In this way, I constrained the variation in strategy across tasks, and the result was a set of models that share common production rules for common task components. By limiting the architecture, strategy, and parameters, I was able to determine the ability of a core set of principles to account for the reliable features of the observed data.

7.2.7 Limitations of Research

I made certain simplifying assumptions to keep the space of model possibilities tractable, but this came at a cost. The only processes of information loss included in the models were decay over time for speech objects and speech links. Other work suggests that decay alone cannot account for errors where particular words were omitted from the sequence being recalled (Mueller, 2002). I also assumed that external, covert and overt speech all decay at the same rate, but some evidence suggests this not to be the case (Kieras et al., 1998).

The recovery strategy used in the models only considered words in the current recall chain as candidates for recall. However, the existence of intrusion errors from dropped words in RFR and RBR shows that this cannot be the full story. A more complete model of recovery should probably consider all words still in memory.

The model of immediate serial recall used for comparison in Experiment 1 was similar to the rehearsal models, but did not use identical production rules. A better parameter match between models may have resulted from a fuller integration.

For RBR in Experiment 3, only a subset of the participants with qualitatively consistent data was modeled. To more fully understand rolling backward updating, more work needs to be done to analyze and model the other participants' performance.

While performance was analyzed in a variety of ways (accuracy, reaction time, articulation time, serial position functions, response-type functions, and word set error functions) there are other possible analyses. There is something of a tradition in the verbal working-memory literature of trumping models by finding shortcomings through increasingly inventive analyses. For example, the models never repeated words during recall because they kept track of the words that had been recalled with tags that did not decay. It is possible that an analysis of repetitions in the observed data would show this to be an incorrect assumption.

7.3 Future Directions

The work in this dissertation can be followed up and expanded at no less than four levels of abstraction. At the most proximal level, the issues just discussed as limitations can be addressed. For example, intrusions of deleted words can be modeled; external, overt, and covert speech can be allowed to decay at different rates; and analyses can be performed to check for word repetitions in recall.

At a second level, specific empirical claims and theoretical implications that have arisen from this work can be evaluated further. For example, the non-selectivity of updating operations could be further tested empirically. Does the co-occurrence of two updating operations truly lead to positive reaction time slopes, or is the critical requirement actually the need to perform deletion?

Another example would be a further consideration of architectural assumptions as they relate to working-memory updating. In the present models, speech objects and links are episodic and lost only through decay, but internal control state is directly created and destroyed by production rules. This control state includes goals, steps, and tags that store the current progress of the system across cognitive cycles. Direct updating of control state is fundamental to the way in which the production rules are written, and is done in almost every rule. This dissertation has shown that verbal working-memory updating can be explained without appealing to processes that directly modify or delete speech objects or links. However, the rehearsal processes that perform this updating extensively use the ability to directly update control state. It would be interesting to explore how the concept of unchangeable episodic traces that are only lost through decay could be extended to all items in working memory, including control state. If this were possible, it would, in a sense, reunify working memory, by showing that all working memories are transient episodic traces that cannot be modified but only ignored and superseded by new memories. Models built on such architectures may very well need a substantially different approach to executive control.

At a third level, the theory specified here could be tested on a wider range of task situations and updating demands. What happens when two words are to be added to the beginning of a sequence at once? Two at the end? One on each end? Does non-specificity apply to these cases as well? What happens when overt recall is no longer required, or when additional task components such as matching are added?

Finally, at the highest level of abstraction, the ideas developed here about memory for ordered sequences of words could be generalized to other modalities. Is spatial working memory updated in an analogous manner? For example, does updating of a sequence of spatial locations depend on rehearsal of a sequence of eye movements?

7.4 Conclusions

This dissertation provides an account of the mental processes and representations involved in the updating of verbal working memory. Through the use of careful experimentation and detailed computational modeling, I have taken an abstract executive process and decomposed it into an explicit theory of performance. I see this as a step towards a deeper understanding of the executive control processes that make humans uniquely capable animals.

APPENDICES

APPENDIX A

Parameters of Models

A.1 Invariant Parameters

The following parameters were maintained at the same values throughout modeling.

Table A.1 Unvarying EPIC parameters used in modeling.

Processor and parameter	Value
Cognitive processor	
CYCLE-TIME	50 ms
Auditory perceptual processor	
SOUND-DETECTION-TIME	50 ms
SOUND-RECODING-TIME	285 ms
SOUND-DECAY-TIME	100 ms
PHRASE-RECODING-TIME	100 ms
COVERT-RECODING-TIME	0 ms
OVERT-RECODING-TIME	0 ms
Vocal motor processor	
FEATURE-PROCESSING-TIME	50 ms
INITIATION-TIME	50 ms
PRODUCTION-ONSET-DELAY	10 ms
SIGNAL-DELAY	0 ms
SIGNAL-DELETION-DELAY	75 ms

A.2 Model Specific Parameters

A small number of auditory perceptual-processor parameters varied across models. However, they were constrained in certain ways. For example, external, overt, and covert speech were always assigned the same decay values, and all links and speech tags were assigned the same decay values. As such the following table lists the actual EPIC parameters that were mapped to each 'virtual' parameter that was varied.

Table A.2 Actual EPIC parameters mapped to each virtual parameter.

Virtual parameter	Actual parameters
Speech object decay mean	EXTERNAL-SPEECH-DECAY-P1 OVERT-SPEECH-DECAY-P1 COVERT-SPEECH-DECAY-P1
Speech object decay spread	EXTERNAL-SPEECH-DECAY-P2 OVERT-SPEECH-DECAY-P2 COVERT-SPEECH-DECAY-P2
Speech link decay mean	EXTERNAL-SPEECH-LINK-DECAY-P1 OVERT-SPEECH-LINK-DECAY-P1 COVERT-SPEECH-LINK-DECAY-P1 EXTERNAL-SPEECH-END-DECAY-P1 SPEECH-TAG-DECAY-P1
Speech link decay spread	EXTERNAL-SPEECH-LINK-DECAY-P2 OVERT-SPEECH-LINK-DECAY-P2 COVERT-SPEECH-LINK-DECAY-P2 EXTERNAL-SPEECH-END-DECAY-P2 SPEECH-TAG-DECAY-P2

The following table lists the values that the virtual parameters were assigned for each model discussed in the text.

Table A.3 Parameters values for best fits for each model.

Experiment and task		Model	Speech object decay		Speech link decay	
			Mean (ms)	Spread	Mean (ms)	Spread
Experiment 1						
	CFR	Basic	4000	0.3	NA	NA
		Guessing	4000	0.3	NA	NA
		Link-decay	6000	0.3	3000	0.6
	ISR	ISR	12000	0.3	3000	0.2
Experiment 2						
	CFR	Link-decay	6000	0.3	3000	0.6
	RFR	Least-change	6000	0.3	3000	0.6
		Covert	6000	0.3	3000	0.6
		Search	6000	0.3	3000	0.6
		Truncated	6000	0.3	3000	0.6
Experiment 3						
	CFR	Link-decay	6000	0.3	4000	0.6
	RFR	Truncated	6000	0.3	4000	0.6
	CBR	Least-change	6000	0.3	4000	0.6
		Postponement	6000	0.3	4500	0.6
	RBR	Least-change	6000	0.3	4000	0.6
		Restrained	6000	0.3	4000	0.6
Experiment 4						
	CBR	Postponement	6000	0.3	4500	0.6
	RBR	Restrained	6000	0.3	4000	0.6

APPENDIX B

Goodness-of-fit Measures for Models

Tables reporting goodness-of-fit measures are included to provide an overall assessment of how well the models fit the observed data. This is provided as a more quantitative evaluation of the models than looking at the graphs. In addition, not every graph can be shown for every model, so this provides a convenient, if highly abbreviated proxy. That being said, goodness-of-fit measures are notoriously suspect ways to evaluate a model. Each measure has its strengths and weaknesses in getting at the intuitive idea of a 'good fit.' Thus, I have included three measures which have different strengths and weaknesses.

The first measure, R^2, provides a measure of the proportion of the variance from the grand mean of the human data accounted for by the model. A weakness of this measure is that it focuses on variance to the detriment of focusing on the absolute magnitude of the values. In cases where there is little or no variance, but theoretically interesting absolute values (e.g., reaction times that do not vary across conditions) R^2 becomes meaningless and I have excluded it.

The second measure, root mean square error (*RMSE*), provides a measure of the deviation of the model predictions from the actual values in the appropriate units of measure. This measure focuses on absolute error, so it provides a nice complement to R^2. A disadvantage is that it is hard to compare *RMSE* values across dependent variables because the units are different.

The third measure, normalized root mean square error (*nRMSE*), provides a measure of the deviation of the model predictions from the actual values on a normalized scale. This overcomes the comparison problem of *RMSE*, but it introduces a new problem of oversensitivity to deviations from small values. For a variable where some values are very close to zero and other values are relatively far from zero, the prediction error around the values near to zero is disproportionately weighted.

All of these measures must be interpreted carefully, but in conjunction they can provide some useful information about goodness-of-fit. Caveat emptor.

B.1 Experiment 1

B.1.1 CFR Link-Decay Model

Table B.1 Goodness-of-fit measures for CFR Link-Decay Model compared to Experiment 1 subtrial data.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.913	.103	.133
Reaction time (ms)	.894	21.2	.030
Articulation time (ms)	.940	174	.111
Serial position functions	.734	.072	.079
Response-type functions	.980	.049	.246

B.1.2 ISR Model

Table B.2 Goodness-of-fit measures for ISR Model compared to Experiment 1 data.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.892	.127	.229
Serial position functions	.519	.133	.156
Response-type functions	.878	.114	.572

B.2 Experiment 2

B.2.1 CFR Link-Decay Model

Table B.3 Goodness-of-fit measures for CFR Link-Decay Model compared to Experiment 2 subtrial data.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.921	.092	.118
Reaction time (ms)	.765	73.2	.114
Articulation time (ms)	.892	270	.161
Serial position functions	.561	.086	.094
Response-type functions	.975	.056	.282

B.2.2 RFR Truncated-Articulation Model

Table B.4 Goodness-of-fit measures for RFR Truncated-Articulation Model compared to Experiment 2 subtrial data.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.940	.054	.067
Reaction time (ms)	.864	79.2	.108
Articulation time (ms)	.868	311	.219
Serial position functions	.371	.088	.097
Response-type functions	.948	.077	.389

B.3 Experiment 3

B.3.1 CFR Link-Decay Model

Table B.5 Goodness-of-fit measures for CFR Link-Decay Model compared to Experiment 3 subtrial data for the subset of participants modeled.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.918	.035	.039
Reaction time (ms)	NA	119	.152
Articulation time (ms)	.995	98.1	.057
Serial position functions	.105	.043	.045
Response-type functions	.984	.048	.242

B.3.2 RFR Truncated-Articulation Model

Table B.6 Goodness-of-fit measures for RFR Truncated Rehearsal Model compared to Experiment 3 subtrial data for the subset of participants modeled.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.982	.023	.027
Reaction time (ms)	.942	48.3	.058
Articulation time (ms)	.952	312	.215
Serial position functions	.448	.064	.069
Response-type functions	.976	.055	.275

B.3.3 CBR Postponement Model

Table B.7 Goodness-of-fit measures for CBR Postponement Model compared to Experiment 3 subtrial data for the subset of participants modeled.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.931	.054	.067
Reaction time (ms)	NA	142	.141
Articulation time (ms)	.987	80.4	.043
Serial position functions	.668	.057	.062
Response-type functions	.978	.051	.257

B.3.4 RBR Restrained-Rehearsal Model

Table B.8 Goodness-of-fit measures for RBR Restrained-Rehearsal Model compared to Experiment 3 subtrial data for the subset of participants modeled.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.886	.127	.170
Reaction time (ms)	.965	115	.106
Articulation time (ms)	.938	204	.116
Serial position functions	.735	.072	.082
Response-type functions	.980	.047	.237

B.4 Experiment 4

B.4.1 CBR Postponement Model

Table B.9 Goodness-of-fit measures for CBR Postponement Model compared to Experiment 4 subtrial data.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.968	.055	.080
Reaction time (ms)	NA	34.6	.037
Articulation time (ms)	.993	201	.090
Serial position functions	.783	.078	.090
Response-type functions	.956	.069	.345

B.4.2 RBR Restrained-Rehearsal Model

Table B.10 Goodness-of-fit measures for RBR Restrained-Rehearsal Model compared to Experiment 4 subtrial data.

	Measure		
Variable	R^2	*RMSE*	*nRMSE*
Probability correct recall	.966	.117	.171
Reaction time (ms)	.187	239	.235
Articulation time (ms)	.946	236	.113
Serial position functions	.807	.077	.091
Response-type functions	.963	.062	.313

APPENDIX C

Production Rules for Models

It would be unwieldy to include the entire sets of production rules for all models discussed in this dissertation. However, since the production rules are the most fully specified description of the strategies embodied in the models, it is appropriate to include a reasonable sampling of them. As such, I have included the full rule set for the CFR Link-Decay Model, since it is the foundation on which the modeling for the other tasks was based. In addition, I have included the rules for the subtrial recall and rehearsal methods of the concluding model for each task. The other parts of these models are largely identical in function to the CFR Link-Decay Model. Taken together, the rules included here describe the most successful models for each of the four rehearsal tasks.

C.1 CFR Link-Decay Model

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rules for the Cumulative Forward Rehearsal (CFR) task.
;; These rules implement the Link Decay strategy for CFR.
;;
;; Note that the rules are written as separate methods or threads.
;; Each collection of rules to implement a separate major process
;; is demarcated by a descriptive header. The rules are written
;; to be reusable across tasks. The only method specific to CFR
;; is the subtrial recall process, since this is where the updating
;; specific to CFR occurs.
;;
;; The processes are:
;; Block Initiation
;; Trial Initiation
;; Startup Recall
;; CFR Update Recall
;; Guessing strategy
;; Hear Articulated Sequence
;; Hear Subtrial or Startup Feedback
;; Cleanup After Trial
;; Housekeeping
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Block Initiation (REH-TASK)
;;
;; Task general rules to start a block of trials.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; The first rule that fires for any rehearsal task block
(R-BLOCK--BEGIN
 IF (
  (GOAL DO REH-TASK)
  (NOT (STATUS R-TASK UNDERWAY))
 ) THEN (
  (ADDDB (STATUS R-TASK UNDERWAY))
  (ADDDB (STEP R-TASK WAIT-FOR BLOCK-TYPE))

  (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
))

; Waits for the beginning of block message which will indicate which
; rehearsal task is to be performed and moves the eyes to it for
; identification.
(R-BLOCK--SEE-TASK
 IF (
  (GOAL DO REH-TASK)
  (STEP R-TASK WAIT-FOR BLOCK-TYPE)

  (VISUAL ?OBJECT DETECTION ONSET)

  (MOTOR OCULAR PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)

  (DELDB (STEP R-TASK WAIT-FOR BLOCK-TYPE))
  (ADDDB (STEP R-TASK IDENTIFY BLOCK-TYPE))
))

; Identifies start of a block of CFR and sets the goal to perform it
(R-BLOCK--IDENTIFY-TASK-AS-CFR
 IF (
  (GOAL DO REH-TASK)
  (STEP R-TASK IDENTIFY BLOCK-TYPE)

  (VISUAL ?OBJECT LABEL CFR)
 ) THEN (
  (DELDB (GOAL DO REH-TASK))
  (ADDDB (GOAL DO R-TASK CFR))

  (DELDB (STEP R-TASK IDENTIFY BLOCK-TYPE))
  (ADDDB (STEP R-TASK BEGIN TRIAL))
))

; Starts performing task by beginning the first trial
(R-BLOCK--BEGIN-TRIAL
 IF (
  (GOAL DO R-TASK ?TASK)
  (STEP R-TASK BEGIN TRIAL)
 ) THEN (
  (DELDB (STEP R-TASK BEGIN TRIAL))

  (ADDDB (STATUS R-TRIAL UNDERWAY))
  (ADDDB (GOAL DO R-TRIAL ?TASK))
  (ADDDB (STEP R-TRIAL WAIT-FOR TRIAL-START))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Trial Initiation (R-TRIAL ?TASK)
;;
```

```
;; Task general rules to start a trial of a rehearsal task.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Look at the trial start message when it arrives
(R-TRIAL--SEE-START
 IF (
  (GOAL DO R-TRIAL ???)
  (STEP R-TRIAL WAIT-FOR TRIAL-START)

  (VISUAL ?OBJECT DETECTION ONSET)

  (MOTOR OCULAR PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)

  (DELDB (STEP R-TRIAL WAIT-FOR TRIAL-START))
  (ADDDB (STEP R-TRIAL IDENTIFY TRIAL-START))
))

; Confirm that we are starting a new trial...
(R-TRIAL--IDENTIFY-START
 IF (
  (GOAL DO R-TRIAL ???)
  (STEP R-TRIAL IDENTIFY TRIAL-START)

  (VISUAL ?OBJECT LABEL TRIAL)
 ) THEN (
  (DELDB (STEP R-TRIAL IDENTIFY TRIAL-START))

  ; Thread to hear the sequence
  (ADDDB (GOAL DO R-HEAR-SEQUENCE EXTERNAL))

  ; Thread to hear the recall signal
  (ADDDB (GOAL DO R-STARTUP-RECALL))
  (ADDDB (STEP R-STARTUP-RECALL WAIT-FOR RECALL-SIGNAL))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Startup Recall (R-STARTUP-RECALL)
;;
;; Task general rules to recall the initial startup sequence on a trial
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Hears the cue that the startup has ended...
(R-STARTUP-RECALL--HEAR-RECALL-SIGNAL
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL WAIT-FOR RECALL-SIGNAL)

  (AUDITORY DETECTION EVENT ONSET)
 ) THEN (
  (DELDB (STEP R-STARTUP-RECALL WAIT-FOR RECALL-SIGNAL))

  ; Begin recall
  (ADDDB (STEP R-STARTUP-RECALL BEGIN RECALL))
  (ADDDB (STATUS R-STARTUP-RECALL STARTING))

  ; Launch thread to hear own speech
  (ADDDB (GOAL DO R-HEAR-SEQUENCE OVERT))

  ; Launch thread to listen for feedback
  (ADDDB (GOAL DO R-HEAR-FEEDBACK))
))

; Begin to recall the startup sequence
(R-STARTUP-RECALL--BEGIN
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL BEGIN RECALL)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ?MARKER TYPE ???)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS R-REHEARSED-CHAIN-START)
  (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  ; Get the content
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG R-STARTUP-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT))

  ; Jump into the main chain recall sequence
  (DELDB (STEP R-STARTUP-RECALL BEGIN RECALL))
  (ADDDB (STEP R-STARTUP-RECALL PRODUCE RECALL))
))

; Error handling: speech object is missing
(R-STARTUP-RECALL--BEGIN--NO-OBJECT
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL BEGIN RECALL)

  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ??? TYPE ???))
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS R-REHEARSED-CHAIN-START)
  (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  ; Jump into the main chain recall sequence
  (DELDB (STEP R-STARTUP-RECALL BEGIN RECALL))
  (ADDDB (STEP R-STARTUP-RECALL CONTINUE RECALL))
))


; Error handling: REHEARSED-CHAIN-START tag missing
(R-STARTUP-RECALL--BEGIN--NO-CHAIN-START
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL BEGIN RECALL)

  (NOT (AUDITORY SPEECH-TAG ??? ??? IS R-REHEARSED-CHAIN-START))
 ) THEN (
  ; Update tags
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  ; Jump into the main chain recall sequence
  (DELDB (STEP R-STARTUP-RECALL BEGIN RECALL))
  (ADDDB (STEP R-STARTUP-RECALL CONTINUE RECALL))
))

; Continue rehearsing the current sequence by retrieving content for
; next to recall
(R-STARTUP-RECALL--CONTINUE
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ?MARKER TYPE ???)
 ) THEN (
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG R-STARTUP-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT))

  (DELDB (STEP R-STARTUP-RECALL CONTINUE RECALL))
  (ADDDB (STEP R-STARTUP-RECALL PRODUCE RECALL))
))

; Error handling: speech object is missing
(R-STARTUP-RECALL--CONTINUE--NO-OBJECT
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ??? TYPE ???))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS EXTERNAL))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: The next object is not in the recall chain
(R-STARTUP-RECALL--CONTINUE--NO-CURRENT
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (DIFFERENT ?ITEM-ID GONE)
  (NOT (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS EXTERNAL))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: Next tag is set to GONE, i.e. link decay
(R-STARTUP-RECALL--CONTINUE--NEXT-TO-RECALL-IS-GONE
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (EQUAL ?ITEM-ID GONE)
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS EXTERNAL))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: No objects left to recall
(R-STARTUP-RECALL--CONTINUE--NO-NEXT-TO-RECALL
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (NOT (TAG R-TRIAL ??? IS NEXT-TO-RECALL))
 ) THEN (
  ; We don't have anymore objects to recall
  (DELDB (STEP R-STARTUP-RECALL CONTINUE RECALL))
  (ADDDB (STEP R-STARTUP-RECALL END RECALL))
))

; Say the first word that needs to be recalled
(R-STARTUP-RECALL--PRODUCE--START
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL PRODUCE RECALL)
  (STATUS R-STARTUP-RECALL STARTING)

  ; Assumes that content is available
  (TAG R-STARTUP-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT)
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE START ?CONTENT)

  (DELDB (TAG R-STARTUP-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT))

  (DELDB (STEP R-STARTUP-RECALL PRODUCE RECALL))
  (ADDDB (STEP R-STARTUP-RECALL FIND NEXT))

  (DELDB (STATUS R-STARTUP-RECALL STARTING))
))

; Say the word that needs to be recalled
(R-STARTUP-RECALL--PRODUCE--CONTINUE
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL PRODUCE RECALL)
  (NOT (STATUS R-STARTUP-RECALL STARTING))

  ; Assumes that content is available
  (TAG R-STARTUP-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT)
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE CONTINUE ?CONTENT)

  (DELDB (TAG R-STARTUP-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT))

  (DELDB (STEP R-STARTUP-RECALL PRODUCE RECALL))
  (ADDDB (STEP R-STARTUP-RECALL FIND NEXT))
))

; Find the next object in the chain
(R-STARTUP-RECALL--FIND-NEXT
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL FIND NEXT)

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE EXTERNAL MARKER ??? TYPE ???)
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG R-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

  (DELDB (STEP R-STARTUP-RECALL FIND NEXT))
  (ADDDB (STEP R-STARTUP-RECALL CONTINUE RECALL))
))

; Error handling: current object is missing
(R-STARTUP-RECALL--FIND-NEXT--OBJECT-MISSING
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL FIND NEXT)
  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ??? TYPE ???))
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (STEP R-STARTUP-RECALL FIND NEXT))
  (ADDDB (STEP R-STARTUP-RECALL CONTINUE RECALL))
))

; Cleanup after recall: Remove the STIMULUS-END tag
(R-STARTUP-RECALL--CLEANUP--REMOVE-CURRENT-STIMULUS-END
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL END RECALL)

  (TAG R-RECALL ?ITEM-ID IS EXTERNAL CURRENT-STIMULUS-END)
 ) THEN (
  (DELDB (TAG R-RECALL ?ITEM-ID IS EXTERNAL CURRENT-STIMULUS-END))
))

; Wrap up the startup recall - we are already waiting for feedback
(R-STARTUP-RECALL--END
 IF (
  (GOAL DO R-STARTUP-RECALL)
  (STEP R-STARTUP-RECALL END RECALL)
 ) THEN (
  (DELDB (STEP R-STARTUP-RECALL END RECALL))
  (DELDB (GOAL DO R-STARTUP-RECALL))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CFR Update Recall (CFR-UPDATE-RECALL)
;;
;; CFR rules for recalling the updated sequence on a subtrial
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Preparation for an update subtrial
(CFR-UPDATE-RECALL--PREPARE
 IF (
  (GOAL DO R-TRIAL CFR)
  (STATUS R-HEAR-FEEDBACK CORRECT)
  (NOT (GOAL DO CFR-UPDATE-RECALL))
 ) THEN (
  (DELDB (STATUS R-HEAR-FEEDBACK CORRECT))

  (ADDDB (GOAL DO CFR-UPDATE-RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL WAIT-FOR STIMULUS))
))

; Hear the new word presented on each subtrial
(CFR-UPDATE-RECALL--HEAR-NEW-STIMULUS
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL WAIT-FOR STIMULUS)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ??? TYPE ???)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))
 ) THEN (
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
  (CREATE-AUDITORY-SPEECH-TAG R-TRIAL ?ITEM-ID R-ADD-CHAIN-START)

  ; Begin recall
  (DELDB (STEP CFR-UPDATE-RECALL WAIT-FOR STIMULUS))
  (ADDDB (STATUS CFR-UPDATE-RECALL STARTING))
  (ADDDB (STEP CFR-UPDATE-RECALL BEGIN RECALL))

  ; Launch thread to hear own speech
  (ADDDB (GOAL DO R-HEAR-SEQUENCE OVERT))

  ; Launch thread to listen for feedback
  (ADDDB (GOAL DO R-HEAR-FEEDBACK))
))

; Start recalling the updated sequence with the recall chain
(CFR-UPDATE-RECALL--BEGIN
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL BEGIN RECALL)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ?MARKER TYPE ???)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS R-REHEARSED-CHAIN-START)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  ; Get the content
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT))

  ; Send the first item to be produced
  (DELDB (STEP CFR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL PRODUCE RECALL))
))

; Error handling: The first speech object is missing
(CFR-UPDATE-RECALL--BEGIN--NO-OBJECT
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL BEGIN RECALL)

  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ??? TYPE ???))
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS R-REHEARSED-CHAIN-START)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  ; Jump into the main chain recall sequence
  (DELDB (STEP CFR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: REHEARSED-CHAIN-START tag is missing
(CFR-UPDATE-RECALL--BEGIN--NO-CHAIN-START
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL BEGIN RECALL)

  (NOT (AUDITORY SPEECH-TAG ??? ??? IS R-REHEARSED-CHAIN-START))
 ) THEN (
  ; Update tags
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  ; Jump into the main chain recall sequence
  (DELDB (STEP CFR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
))

; Continue rehearsing the current sequence by retrieving content for
; next to recall
(CFR-UPDATE-RECALL--CONTINUE
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ?MARKER TYPE ???)
 ) THEN (
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT))

  (DELDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL PRODUCE RECALL))
))

; Error handling: Speech object is missing
(CFR-UPDATE-RECALL--CONTINUE--NO-OBJECT
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS OVERT))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: The next object is not in the recall chain
(CFR-UPDATE-RECALL--CONTINUE--NO-CURRENT
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (DIFFERENT ?ITEM-ID GONE)
  (NOT (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS OVERT))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: We lost the link to the next object
(CFR-UPDATE-RECALL--CONTINUE--NEXT-TO-RECALL-IS-GONE
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (EQUAL ?ITEM-ID GONE)
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS OVERT))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
```

```
))

; Error handling: No more speech objects to recall
(CFR-UPDATE-RECALL--CONTINUE--NO-NEXT-TO-RECALL
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (NOT (TAG R-TRIAL ??? IS NEXT-TO-RECALL))
 ) THEN (
  ; We don't have anymore objects to recall
  (DELDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL END RECALL))
))

; Say the first word that needs to be recalled
(CFR-UPDATE-RECALL--PRODUCE--START
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL PRODUCE RECALL)
  (STATUS CFR-UPDATE-RECALL STARTING)

  ; Assumes that content is available
  (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT)
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE START ?CONTENT)

  (DELDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT))

  (DELDB (STEP CFR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL FIND NEXT))

  (DELDB (STATUS CFR-UPDATE-RECALL STARTING))
))

; Say the next word that needs to be recalled
(CFR-UPDATE-RECALL--PRODUCE--CONTINUE
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL PRODUCE RECALL)
  (NOT (STATUS CFR-UPDATE-RECALL STARTING))

  ; Assumes that content is available
  (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT)
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE CONTINUE ?CONTENT)

  (DELDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT))

  (DELDB (STEP CFR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP CFR-UPDATE-RECALL FIND NEXT))
))

; Find the next object in the recall chain
(CFR-UPDATE-RECALL--FIND-NEXT
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL FIND NEXT)

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
  (NOT (TAG R-RECALL ?NEXT-ID IS OVERT CURRENT-STIMULUS-END))
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
```

```
   (ADDDB (TAG R-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

   (DELDB (STEP CFR-UPDATE-RECALL FIND NEXT))
   (ADDDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Speech object is missing
(CFR-UPDATE-RECALL--FIND-NEXT--OBJECT-MISSING
 IF (
   (GOAL DO CFR-UPDATE-RECALL)
   (STEP CFR-UPDATE-RECALL FIND NEXT)

   (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
   (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
 ) THEN (
   (DELDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
   (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT RECALLED))

   (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
   (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

   (DELDB (STEP CFR-UPDATE-RECALL FIND NEXT))
   (ADDDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
))

; We are at the end of the recall chain, so its time for the add object
(CFR-UPDATE-RECALL--FIND-NEXT--ADD-NEW-OBJECT
 IF (
   (GOAL DO CFR-UPDATE-RECALL)
   (STEP CFR-UPDATE-RECALL FIND NEXT)

   (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
   (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
   (TAG R-RECALL ?NEXT-ID IS OVERT CURRENT-STIMULUS-END)

   (AUDITORY SPEECH-TAG ?UNUM ?ADD-ID IS R-ADD-CHAIN-START)
 ) THEN (
   (DELDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
   (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT RECALLED))

   (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
   (ADDDB (TAG R-TRIAL ?ADD-ID IS NEXT-TO-RECALL))

   (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)

   (DELDB (STEP CFR-UPDATE-RECALL FIND NEXT))
   (ADDDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Time for the add object, but the tag is missing
(CFR-UPDATE-RECALL--FIND-NEXT--NO-ADD-NEW-OBJECT
 IF (
   (GOAL DO CFR-UPDATE-RECALL)
   (STEP CFR-UPDATE-RECALL FIND NEXT)

   (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
   (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
   (TAG R-RECALL ?NEXT-ID IS OVERT CURRENT-STIMULUS-END)

   (NOT (AUDITORY SPEECH-TAG ??? ??? IS R-ADD-CHAIN-START))
 ) THEN (
   (DELDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
   (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT RECALLED))

   (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
   (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

   (DELDB (STEP CFR-UPDATE-RECALL FIND NEXT))
   (ADDDB (STEP CFR-UPDATE-RECALL CONTINUE RECALL))
))
```

```
; Cleanup after recall: Remove the CURRENT-STIMULUS-END tag
(CFR-UPDATE-RECALL--CLEANUP--REMOVE-CURRENT-STIMULUS-END
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL END RECALL)

  (TAG R-RECALL ?ITEM-ID IS OVERT CURRENT-STIMULUS-END)
 ) THEN (
  (DELDB (TAG R-RECALL ?ITEM-ID IS OVERT CURRENT-STIMULUS-END))
))

; Wrap up recall - we are already waiting for feedback
(CFR-UPDATE-RECALL--END
 IF (
  (GOAL DO CFR-UPDATE-RECALL)
  (STEP CFR-UPDATE-RECALL END RECALL)
 ) THEN (
  (DELDB (STEP CFR-UPDATE-RECALL END RECALL))
  (DELDB (GOAL DO CFR-UPDATE-RECALL))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Guessing strategy (R-REBUILD-CHAINS)
;;
;; Task general rules to implement a guessing strategy
;;
;; Mark all speech objects left to recall.
;; Find all intact chains of objects, marking the chain terminating
;; with the last object as the end chain.
;; Pick a chain to recall.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Mark any remaining objects as NOT-FOUND, except the
; CURRENT-STIMULUS-END and the ADD-CHAIN
(R-REBUILD-CHAINS--MARK-OBJECTS
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS MARK ITEMS)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
  (NOT (TAG R-RECALL ?NEXT-ID IS ?SOURCE CURRENT-STIMULUS-END))
  (NOT (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS R-ADD-CHAIN-START))
  (TAG R-TRIAL ?ITEM-ID IS ?SOURCE CURRENT)
 ) THEN (
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS NOT-FOUND))
))

; Find the last object if possible and mark it as the
; END-CHAIN-HEAD, and as FOUND
(R-REBUILD-CHAINS--MARK-OBJECTS--END-CHAIN-HEAD
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS MARK ITEMS)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
  (TAG R-TRIAL ?ITEM-ID IS ?SOURCE CURRENT)
  (TAG R-RECALL ?NEXT-ID IS ?SOURCE CURRENT-STIMULUS-END)
 ) THEN (
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS END-CHAIN-HEAD))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS FOUND))
))

; Find the update object and mark it as ADD-OBJECT and as FOUND
(R-REBUILD-CHAINS--MARK-OBJECTS--ADD-OBJECT
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS MARK ITEMS)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???) ; Note that
we don't care about the real source
  (TAG R-TRIAL ?ITEM-ID IS ?SOURCE CURRENT)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS R-ADD-CHAIN-START)
 ) THEN (
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS ADD-OBJECT))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS FOUND))
))

; Advance the flow of control after marking all current objects
(R-REBUILD-CHAINS--MARK-OBJECTS--END
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STATUS R-REBUILD-CHAINS REBUILDING CHAINS)
  (STEP R-REBUILD-CHAINS MARK ITEMS)
 ) THEN (
  (DELDB (STEP R-REBUILD-CHAINS MARK ITEMS))
  (ADDDB (STEP R-REBUILD-CHAINS SELECT ITEM))
))

; Select one of the NOT-FOUND objects as CHAIN-HEAD
(R-REBUILD-CHAINS--SELECT-OBJECT
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS SELECT ITEM)

  (TAG R-REBUILD-CHAINS ?ITEM-ID IS NOT-FOUND)

  (RANDOMLY-CHOOSE-ONE ?ITEM-ID)
 ) THEN (
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CHAIN-HEAD))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT))

  (DELDB (STEP R-REBUILD-CHAINS SELECT ITEM))
  (ADDDB (STEP R-REBUILD-CHAINS FIND SUCCESSOR))
))

; If no NOT-FOUND objects are left, then move to chain selection
(R-REBUILD-CHAINS--SELECT-OBJECT--NO-OBJECTS-LEFT
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS SELECT ITEM)

  ; No item exists
  (NOT (TAG R-REBUILD-CHAINS ??? IS NOT-FOUND))
 ) THEN (
  (DELDB (STEP R-REBUILD-CHAINS SELECT ITEM))
  (ADDDB (STEP R-REBUILD-CHAINS PICK HEAD))
))

; Currently selected object can't be found
(R-REBUILD-CHAINS--FIND-SUCCESSOR--NO-OBJECT
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS FIND SUCCESSOR)

  (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
 ) THEN (
  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT))
  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS NOT-FOUND))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS FOUND))

  (DELDB (STEP R-REBUILD-CHAINS FIND SUCCESSOR))
  (ADDDB (STEP R-REBUILD-CHAINS SELECT ITEM))
))

; No successor to the currently selected object, or the link is GONE
(R-REBUILD-CHAINS--FIND-SUCCESSOR--NO-SUCCESSOR
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS FIND SUCCESSOR)

  (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT SOURCE ??? MARKER ??? TYPE ???)
  (NOT (TAG R-REBUILD-CHAINS ?NEXT IS NOT-FOUND))
  (NOT (TAG R-REBUILD-CHAINS ?NEXT IS FOUND))
 ) THEN (
  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT))
  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS NOT-FOUND))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS FOUND))

  (DELDB (STEP R-REBUILD-CHAINS FIND SUCCESSOR))
  (ADDDB (STEP R-REBUILD-CHAINS SELECT ITEM))
))

; Found the successor to the currently selected object, and
; it is marked NOT-FOUND
(R-REBUILD-CHAINS--FIND-SUCCESSOR--SUCCESSOR--NOT-FOUND
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS FIND SUCCESSOR)

  (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT SOURCE ??? MARKER ??? TYPE ???)
  (TAG R-REBUILD-CHAINS ?NEXT IS NOT-FOUND)
 ) THEN (
  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT))
  (ADDDB (TAG R-REBUILD-CHAINS ?NEXT IS CURRENT))
))

; Found the successor to the currently selected object,
; and it is marked FOUND
(R-REBUILD-CHAINS--FIND-SUCCESSOR--SUCCESSOR--FOUND
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS FIND SUCCESSOR)

  (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT SOURCE ??? MARKER ??? TYPE ???)
  (TAG R-REBUILD-CHAINS ?NEXT IS FOUND)
  (NOT (TAG R-REBUILD-CHAINS ?NEXT IS END-CHAIN-HEAD))
 ) THEN (
  (DELDB (TAG R-REBUILD-CHAINS ?NEXT IS CHAIN-HEAD))

  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT))
  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS NOT-FOUND))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS FOUND))

  (DELDB (STEP R-REBUILD-CHAINS FIND SUCCESSOR))
  (ADDDB (STEP R-REBUILD-CHAINS SELECT ITEM))
))

; Found the successor to the currently selected object,
; and it is the END-CHAIN-HEAD
(R-REBUILD-CHAINS--FIND-SUCCESSOR--SUCCESSOR--FOUND--END-CHAIN-HEAD
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS FIND SUCCESSOR)

  (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT SOURCE ??? MARKER ??? TYPE ???)
  (TAG R-REBUILD-CHAINS ?NEXT IS FOUND)
  (TAG R-REBUILD-CHAINS ?NEXT IS END-CHAIN-HEAD)
 ) THEN (
  (DELDB (TAG R-REBUILD-CHAINS ?NEXT IS END-CHAIN-HEAD))

  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CHAIN-HEAD))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS END-CHAIN-HEAD))

  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS CURRENT))
  (DELDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS NOT-FOUND))
  (ADDDB (TAG R-REBUILD-CHAINS ?ITEM-ID IS FOUND))

  (DELDB (STEP R-REBUILD-CHAINS FIND SUCCESSOR))
  (ADDDB (STEP R-REBUILD-CHAINS SELECT ITEM))
))

; Pick a chain to recall
(R-REBUILD-CHAINS--PICK-CHAIN-TO-RECALL
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS PICK HEAD)

  (TAG R-REBUILD-CHAINS ?ITEM-ID IS CHAIN-HEAD)

  (RANDOMLY-CHOOSE-ONE ?ITEM-ID)
 ) THEN (
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (DELDB (STEP R-REBUILD-CHAINS PICK HEAD))
  (ADDDB (STEP R-REBUILD-CHAINS CLEANUP TAGS))

  ; We've still got to cleanup, but we signal the fact we are
; done to the calling process
  (DELDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
))

; Pick the end chain to recall
(R-REBUILD-CHAINS--PICK-CHAIN-TO-RECALL--END-CHAIN-HEAD
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS PICK HEAD)

  (NOT (TAG R-REBUILD-CHAINS ??? IS CHAIN-HEAD))
  (TAG R-REBUILD-CHAINS ?ITEM-ID IS END-CHAIN-HEAD)
 ) THEN (
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (DELDB (STEP R-REBUILD-CHAINS PICK HEAD))
  (ADDDB (STEP R-REBUILD-CHAINS CLEANUP TAGS))

  (DELDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
))

; We found no chains to recall, but we did find the ADD-OBJECT
(R-REBUILD-CHAINS--PICK-CHAIN-TO-RECALL--ADD-OBJECT
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS PICK HEAD)

  (NOT (TAG R-REBUILD-CHAINS ??? IS CHAIN-HEAD))
  (NOT (TAG R-REBUILD-CHAINS ??? IS END-CHAIN-HEAD))
  (TAG R-REBUILD-CHAINS ?ITEM-ID IS ADD-OBJECT)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS R-ADD-CHAIN-START)
 ) THEN (
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)

  (DELDB (STEP R-REBUILD-CHAINS PICK HEAD))
  (ADDDB (STEP R-REBUILD-CHAINS CLEANUP TAGS))

  ; We've still got to cleanup, but we signal the fact we are
  ; done to the calling process
  (DELDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
))

; Nothing left to recall so we give up on guessing
(R-REBUILD-CHAINS--PICK-CHAIN-TO-RECALL--NO-CHAIN-HEAD
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS PICK HEAD)
  (NOT (TAG R-REBUILD-CHAINS ??? IS CHAIN-HEAD))
  (NOT (TAG R-REBUILD-CHAINS ??? IS END-CHAIN-HEAD))
  (NOT (TAG R-REBUILD-CHAINS ??? IS ADD-OBJECT))
 ) THEN (
  (DELDB (STEP R-REBUILD-CHAINS PICK HEAD))
  (ADDDB (STEP R-REBUILD-CHAINS CLEANUP TAGS))

  ; We've still got to cleanup, but we signal the fact we are
  ; done to the calling process
  (DELDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
))

; Clean up after guessing: Remove tags
(R-REBUILD-CHAINS--CLEANUP-REBUILD-TAGS
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS CLEANUP TAGS)

  (TAG R-REBUILD-CHAINS ?OBJECT IS ?SOMETHING)
 ) THEN (
  (DELDB (TAG R-REBUILD-CHAINS ?OBJECT IS ?SOMETHING))
))

; Finished cleaning up after guessing
(R-REBUILD-CHAINS--CLEANUP-REBUILD-TAGS--END
 IF (
  (GOAL DO R-REBUILD-CHAINS ?SOURCE)
  (STEP R-REBUILD-CHAINS CLEANUP TAGS)
 ) THEN (
  (DELDB (STEP R-REBUILD-CHAINS CLEANUP TAGS))
  (DELDB (GOAL DO R-REBUILD-CHAINS ?SOURCE))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Hear Articulated Sequence (R-HEAR-SEQUENCE ?SOURCE)
;;
;; Method for hearing a sequence of words and tagging it for subsequent
;; recall. This can handle an overt, covert or external sequence.
;; It will tag it appropriately, including START and END tags, and
;; per word tags. It can handle sequences of one or more objects.
;;
;; Note: This is designed to be "multi-threaded" in that there can
;; be methods active for each source simultaneously, and the rules
;; won't clash with each other. In order for this to work, the
;; source must be included in the GOAL and STEPs.
;;
;; Note: This also uses an aggressive style of indepedent sub-rules
;; that reduces repetition of actions, but leads to more rules firing
;; simultaneously to handle the different portions of a task.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; If the goal to perform this method is placed in memory, then this
; rule will launch the method. Note that the source must be specified
; in the goal
(R-HEAR-SEQUENCE--BEGIN
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)

  (NOT (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI))
 ) THEN (
  (ADDDB (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI))
  (ADDDB (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR START))
))

; Mark each new object from the source as NEW and TO-BE-RECALLED
(R-HEAR-SEQUENCE--ADD-NEW-TAG
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR ???)

  ; Find a new item from the correct source
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ?SOURCE MARKER ?MARKER TYPE ???)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))
 ) THEN (
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS ?SOURCE NEW))
))

; Mark the first object from the source as the chain start
(R-HEAR-SEQUENCE--ADD-START-TAG
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR START)

  ; Find a new item from the correct source
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ?SOURCE MARKER ?MARKER TYPE ???)
  (DIFFERENT ?MARKER END)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))
 ) THEN (
  ; Mark it as the start.
  (CREATE-AUDITORY-SPEECH-TAG R-TRIAL ?ITEM-ID R-REHEARSED-CHAIN-START)

  ; We are now waiting for the sequence to complete
  (DELDB (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR START))
  (ADDDB (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR END))
))

; Mark the first object from the source as the chain start, when
; it is also the last
(R-HEAR-SEQUENCE--ADD-START-TAG-AT-END
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR START)

  ; Find a new item from the correct source
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ?SOURCE MARKER END TYPE ???)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))
 ) THEN (
  ; Mark it as the start
  (CREATE-AUDITORY-SPEECH-TAG R-TRIAL ?ITEM-ID R-REHEARSED-CHAIN-START)

  ; We are now waiting for the sequence to complete
  (DELDB (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR START))
))


; Add a new END tag, if we have the NEXT link for this object
(R-HEAR-SEQUENCE--ADD-END-TAG
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR ???)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ?SOURCE MARKER ?MARKER TYPE ???)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))
  (DIFFERENT ?NEXT-ID GONE)
 ) THEN (
  (ADDDB (TAG R-RECALL ?NEXT-ID IS ?SOURCE NEW-STIMULUS-END))
))

; Remove the old END tag when we get a new object
(R-HEAR-SEQUENCE--REMOVE-END-TAG
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR ???)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ?SOURCE MARKER ?MARKER TYPE ???)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))

  ; Find the current END tag, which we will now update
  (TAG R-RECALL ?OLD-ID IS ?SOURCE NEW-STIMULUS-END)
 ) THEN (
  (DELDB (TAG R-RECALL ?OLD-ID IS ?SOURCE NEW-STIMULUS-END))
))

; If we hear an item with an END marker, then we know it is the end,
; so we should stop the current method. Note that the above rules will
; handle the item, this just needs to tear down the method.
(R-HEAR-SEQUENCE--FOUND-END-MARKER
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR ?MARKER)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ?SOURCE MARKER END TYPE ???)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))
 ) THEN (
  (DELDB (STEP R-HEAR-SEQUENCE ?SOURCE WAIT-FOR ?MARKER))
  (ADDDB (STEP R-HEAR-SEQUENCE ?SOURCE MAKE CURRENT))
))

; The sequence we just heard is now the current recall chain
(R-HEAR-SEQUENCE--MAKE-CURRENT
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE MAKE CURRENT)

  (TAG R-TRIAL ?ITEM-ID IS ?SOURCE NEW)
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS ?SOURCE NEW))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS ?SOURCE CURRENT))
))

; The sequence we just heard is now the current recall chain
(R-HEAR-SEQUENCE--MAKE-CURRENT-STIMULUS-END
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE MAKE CURRENT)

  (TAG R-RECALL ?ITEM-ID IS ?SOURCE NEW-STIMULUS-END)
 ) THEN (
  (DELDB (TAG R-RECALL ?ITEM-ID IS ?SOURCE NEW-STIMULUS-END))
  (ADDDB (TAG R-RECALL ?ITEM-ID IS ?SOURCE CURRENT-STIMULUS-END))
))

; Done hearing the sequence
(R-HEAR-SEQUENCE--END
 IF (
  (GOAL DO R-HEAR-SEQUENCE ?SOURCE)
  (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE ?SOURCE MAKE CURRENT)
 ) THEN (
  (DELDB (GOAL DO R-HEAR-SEQUENCE ?SOURCE))
  (DELDB (STATUS R-HEAR-SEQUENCE ?SOURCE WAITING-FOR STIMULI))
  (DELDB (STEP R-HEAR-SEQUENCE ?SOURCE MAKE CURRENT))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Hear Subtrial or Startup Feedback (R-HEAR-FEEDBACK)
;;
;; Rules for hearing subtrial feedback and either beginning to
;; clean up, or preparing for next the next subtrial.
;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; When the goal to R-HEAR-FEEDBACK is placed in memory, this will
; launch the method.
(R-HEAR-FEEDBACK--BEGIN
 IF (
  (GOAL DO R-HEAR-FEEDBACK)

  (NOT (STATUS R-HEAR-FEEDBACK WAITING-FOR FEEDBACK))
 ) THEN (
  (ADDDB (STEP R-HEAR-FEEDBACK HEAR FEEDBACK))
  (ADDDB (STATUS R-HEAR-FEEDBACK WAITING-FOR FEEDBACK))
))

; We got it wrong - so cleanup for next trial
(R-HEAR-FEEDBACK--ERROR
 IF (
  (GOAL DO R-HEAR-FEEDBACK)
  (STEP R-HEAR-FEEDBACK HEAR FEEDBACK)
  (STATUS R-HEAR-FEEDBACK WAITING-FOR FEEDBACK)

  (AUDITORY RECOGNITION TYPE TONE FREQ 110 STATE ON) ; Error sound
 ) THEN (
  (ADDDB (GOAL DO R-CLEANUP-TRIAL))
))

; Completed the trial correctly, so time to end and cleanup
(R-HEAR-FEEDBACK--COMPLETE
 IF (
  (GOAL DO R-HEAR-FEEDBACK)
  (STEP R-HEAR-FEEDBACK HEAR FEEDBACK)
  (STATUS R-HEAR-FEEDBACK WAITING-FOR FEEDBACK)

  (AUDITORY RECOGNITION TYPE TONE FREQ 880 STATE ON) ; Complete sound
 ) THEN (
  (ADDDB (GOAL DO R-CLEANUP-TRIAL))
))

; We got it right, so get ready for next sub-trial
(R-HEAR-FEEDBACK--CORRECT
 IF (
  (GOAL DO R-HEAR-FEEDBACK)
  (STEP R-HEAR-FEEDBACK HEAR FEEDBACK)
  (STATUS R-HEAR-FEEDBACK WAITING-FOR FEEDBACK)

  (AUDITORY RECOGNITION TYPE TONE FREQ 440 STATE ON) ; Correct sound
 ) THEN (
  (ADDDB (STATUS R-HEAR-FEEDBACK CORRECT))
))

; When we hear feedback, stop listening for overt objects if
; we still are.
(R-HEAR-FEEDBACK--STOP-HEARING-OVERT
 IF (
  (GOAL DO R-HEAR-FEEDBACK)
  (STEP R-HEAR-FEEDBACK HEAR FEEDBACK)

  (AUDITORY RECOGNITION TYPE TONE FREQ ?FREQ STATE ON)
  (DIFFERENT ?FREQ 55) ; This is making sure we don't get confused by
                       ; the startup recall signal

  (GOAL DO R-HEAR-SEQUENCE OVERT)
  (STATUS R-HEAR-SEQUENCE OVERT WAITING-FOR STIMULI)
  (STEP R-HEAR-SEQUENCE OVERT WAIT-FOR ?MARKER)
 ) THEN (
  (DELDB (STEP R-HEAR-SEQUENCE OVERT WAIT-FOR ?MARKER))
  (ADDDB (STEP R-HEAR-SEQUENCE OVERT MAKE CURRENT))
))

; When we hear feedback stop waiting to hear feedback.
(R-HEAR-FEEDBACK--END
 IF (
  (GOAL DO R-HEAR-FEEDBACK)
  (STEP R-HEAR-FEEDBACK HEAR FEEDBACK)

  (AUDITORY RECOGNITION TYPE TONE FREQ ?FREQ STATE ON)
  (DIFFERENT ?FREQ 55) ; This is making sure we don't get confused by
                       ; the startup recall signal
 ) THEN (
  ; Remove the method to hear feedback
  (DELDB (GOAL DO R-HEAR-FEEDBACK))
  (DELDB (STATUS R-HEAR-FEEDBACK WAITING-FOR FEEDBACK))
  (DELDB (STEP R-HEAR-FEEDBACK HEAR FEEDBACK))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Cleanup After Trial (R-CLEANUP-TRIAL)
;;
;; Rules for cleaning up after the trial.
;; Ideally the methods above would cleanup for themselves
;; but they don't always do it.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Launch thread to cleanup the trial
(R-CLEANUP-TRIAL--BEGIN
 IF (
  (GOAL DO R-CLEANUP-TRIAL)

  (NOT (STATUS R-CLEANUP-TRIAL CLEANUP TRIAL))
 ) THEN (
  (ADDDB (STATUS R-CLEANUP-TRIAL CLEANUP TRIAL))
))

; The recall chain is now old items
(R-CLEANUP-TRIAL--MOVE-TRIAL-TAGS
 IF (
  (GOAL DO R-CLEANUP-TRIAL)
  (STATUS R-CLEANUP-TRIAL CLEANUP TRIAL)

  (TAG R-TRIAL ?ITEM-ID IS ?SOURCE CURRENT)
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS ?SOURCE CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS ?SOURCE RECALLED))
))

; Clean up any trial related tags
(R-CLEANUP-TRIAL--REMOVE-TRIAL-TAGS
 IF (
  (GOAL DO R-CLEANUP-TRIAL)
  (STATUS R-CLEANUP-TRIAL CLEANUP TRIAL)

  (TAG R-TRIAL ?ITEM-ID IS ?SOMETHING)
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS ?SOMETHING))
))

; Clean up any speech tags
(R-CLEANUP-TRIAL--REMOVE-SPEECH-TAGS
 IF (
  (GOAL DO R-CLEANUP-TRIAL)
  (STATUS R-CLEANUP-TRIAL CLEANUP TRIAL)

  (AUDITORY SPEECH-TAG ?UNUM ??? IS ???)
 ) THEN (
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
))

; We are done with cleanup and ready to wait for the next trial
; to begin
```

```
(R-CLEANUP-TRIAL--END
 IF (
  (GOAL DO R-CLEANUP-TRIAL)
  (STATUS R-CLEANUP-TRIAL CLEANUP TRIAL)
 ) THEN (
  (DELDB (GOAL DO R-CLEANUP-TRIAL))
  (DELDB (STATUS R-CLEANUP-TRIAL CLEANUP TRIAL))
  (ADDDB (STEP R-TRIAL WAIT-FOR TRIAL-START))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Housekeeping
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; If an old object disappears, then get rid of its associated tag
(R-TRIAL--REMOVE-RECALLED-OBJECT-TAGS
 IF (
  (TAG R-TRIAL ?ITEM-ID IS ?SOURCE RECALLED)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS ?SOURCE RECALLED))
))
```

C.2 RFR Truncated-Articulation Model

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rules for the Rolling Forward Rehearsal (RFR) task.
;; These rules implement the Truncated Covert Articulation strategy
;; for RFR.
;;
;; Only the RFR Update Recall and Update Rehearsal rules are listed,
;; as the rest of the rules are substantially the same as those for
;; the CFR Guessing strategy.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Identifies start of a block of CBR and sets the goal to perform it
(R-BLOCK--IDENTIFY-TASK-AS-RFR
 IF (
  (GOAL DO REH-TASK)
  (STEP R-TASK IDENTIFY BLOCK-TYPE)

  (VISUAL ?OBJECT LABEL RFR)
 ) THEN (
  (DELDB (GOAL DO REH-TASK))
  (ADDDB (GOAL DO R-TASK RFR))

  (DELDB (STEP R-TASK IDENTIFY BLOCK-TYPE))
  (ADDDB (STEP R-TASK BEGIN TRIAL))
))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; RFR Update Rehearsal (RFR-UPDATE-REHEARSAL)
;;
;; RFR rules for covert truncated rehearsal on a subtrial. The
;; first word is skipped, and the new word is not yet added.
;;
;; Rehearsal uses a simple form of recovery in case of lost links
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Step over the first item in the sequence
(RFR-UPDATE-REHEARSAL--BEGIN
 IF (
  ; Time to do covert rehearsal
```

```
   (GOAL DO RFR-UPDATE-REHEARSAL)
   (STEP RFR-UPDATE-REHEARSAL BEGIN REHEARSAL)

   ; Find the first item in the chain to be rehearsed
   (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ?MARKER TYPE ???)
   (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW)
   (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS RFR-REHEARSED-CHAIN-START)
  ) THEN (
   ; Get rid of the chain start tag, and mark the next word to rehearse
   (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
   (DELDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW))
   (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT RECALLED))
   (ADDDB (TAG RFR-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

   ; We aren't going to say the first word, so just continue on
   (DELDB (STEP RFR-UPDATE-REHEARSAL BEGIN REHEARSAL))
   (ADDDB (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL))

   ; Flag that rehearsal is ongoing
   (ADDDB (STATUS RFR-UPDATE-REHEARSAL REHEARSING))

   ; Get a thread going to hear self rehearse
   (ADDDB (GOAL DO RFR-HEAR-SEQUENCE COVERT))

   ; Start waiting to hear the new item, and be ready to prepare recall
   (ADDDB (GOAL DO RFR-UPDATE-RECALL))
   (ADDDB (STEP RFR-UPDATE-RECALL WAIT-FOR STIMULUS))
   (ADDDB (STEP RFR-UPDATE-RECALL PREPARE RECALL))

   ; Set a process up to listen for feedback
   (ADDDB (GOAL DO RFR-HEAR-FEEDBACK))
 ))

 ; Error handling: Object is missing
 (RFR-UPDATE-REHEARSAL--BEGIN--NO-OBJECT
  IF (
   ; Time to do covert rehearsal
   (GOAL DO RFR-UPDATE-REHEARSAL)
   (STEP RFR-UPDATE-REHEARSAL BEGIN REHEARSAL)

   ; We have the tag pointing to the first item, but no first item
   (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ??? TYPE ???))
   (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW)
   (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS RFR-REHEARSED-CHAIN-START)
  ) THEN (
   ; Get rid of the chain start tag
   (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
   (DELDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW))
   (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT RECALLED))
   (ADDDB (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL))

   ; We aren't going to say the first word, so just continue on
   (DELDB (STEP RFR-UPDATE-REHEARSAL BEGIN REHEARSAL))
   (ADDDB (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL))

   ; Flag that rehearsal is ongoing
   (ADDDB (STATUS RFR-UPDATE-REHEARSAL REHEARSING))

   ; Get a thread going to hear self rehearse
   (ADDDB (GOAL DO RFR-HEAR-SEQUENCE COVERT))

   ; Start waiting to hear the new item, and be ready to prepare recall
   (ADDDB (GOAL DO RFR-UPDATE-RECALL))
   (ADDDB (STEP RFR-UPDATE-RECALL WAIT-FOR STIMULUS))
   (ADDDB (STEP RFR-UPDATE-RECALL PREPARE RECALL))

   ; Set a process up to listen for feedback
   (ADDDB (GOAL DO RFR-HEAR-FEEDBACK))
 ))

 ; Error handling: Restart rehearsal when a link is missing
```

```
(RFR-UPDATE-REHEARSAL--RESTART
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RFR-UPDATE-REHEARSAL)
  (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We don't have the next speech object
  (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL)

  ; We do have NEW tags remaining
  (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW)
  (RANDOMLY-CHOOSE-ONE ?ITEM-ID)
 ) THEN (
  ;Move the pointer to the next item
  (DELDB (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
))

; Continue rehearsing the current sequence
(RFR-UPDATE-REHEARSAL--CONTINUE
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RFR-UPDATE-REHEARSAL)
  (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We have the next speech object
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ?MARKER TYPE ???)
  (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW)
  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
 ) THEN (
  ; Get the phonological item to rehearse, and set up the
  ; subvocalization of it
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS SUBVOCALIZE CONTINUE ^TO-BE-
RECALLED-CONTENT OVERT))
  (ADDDB (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
  (ADDDB (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ;Move the pointer to the next item
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW))
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT RECALLED))
))

; Error handling: Object is missing
(RFR-UPDATE-REHEARSAL--CONTINUE--NO-OBJECT
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RFR-UPDATE-REHEARSAL)
  (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We don't have the next speech object
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ??? TYPE ???))
  (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW)
  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
 ) THEN (
  ; Get rid of the chain start tag
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW))
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS OVERT RECALLED))
))
```

```
; Error handling: Next object not in recall chain
(RFR-UPDATE-REHEARSAL--CONTINUE--NO-NEW
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RFR-UPDATE-REHEARSAL)
  (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We don't have the next speech object
  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (DIFFERENT ?ITEM-ID GONE)
  (NOT (TAG RFR-TRIAL ?ITEM-ID IS OVERT NEW))
 ) THEN (
  ; Change the next marker to GONE
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL))
))

; Subvocalize the word that needs to be rehearsed
(RFR-UPDATE-REHEARSAL--PRODUCE
 IF (
  ; Time to rehearse an item
  (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL)

  ; We actually have an item to articulate
  (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT ?SOURCE)
  (DIFFERENT ?CONTENT NIL)

  ; There are still items to rehearse after this...
  (TAG RFR-TRIAL ??? IS OVERT NEW)

  ; Ready to speak
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  ; Changed to use truncated articulation!!!!
  (SEND-TO-MOTOR-F 'VOCAL ?STYLE ?MARKER (TRUNC ?CONTENT))

  ; Clean up the step and tags
  (DELDB (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
  (DELDB (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
  (DELDB (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT
?SOURCE))
))

; Error handling: Missing the word that needs to be recalled,
; just skip it and move on
(RFR-UPDATE-REHEARSAL--PRODUCE--NO-ITEM
 IF (
  ; Time to rehearse an item
  (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL)

  ; The item to articulate is missing
  (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER NIL ?SOURCE)

  ; There are still items to rehearse after this...
  (TAG RFR-TRIAL ??? IS OVERT NEW)

  ; Ready to speak
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  ; Clean up the step and tags
  (DELDB (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
  (DELDB (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
  (DELDB (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER NIL ?SOURCE))
))

; Subvocalize the last word that needs to be rehearsed
(RFR-UPDATE-REHEARSAL--PRODUCE--LAST-NEW
 IF (
  ; Time to rehearse an item
  (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL)

  ; We actually have an item to articulate
  (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT ?SOURCE)
  (DIFFERENT ?CONTENT NIL)

  ; This is the last item to rehearse
  (NOT (TAG RFR-TRIAL ??? IS OVERT NEW))
  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)

  ; Ready to speak
  (MOTOR VOCAL PROCESSOR FREE)
  ) THEN (
  ; Changed to use truncated articulation!!!!
  (SEND-TO-MOTOR-F 'VOCAL ?STYLE 'END (TRUNC ?CONTENT))

  ; Clean up the step and tags
  (DELDB (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
  (DELDB (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
  (DELDB (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT
?SOURCE))

  ; Stop the rehearsal method
  (DELDB (GOAL DO RFR-UPDATE-REHEARSAL))
  (DELDB (STATUS RFR-UPDATE-REHEARSAL REHEARSING))
  (DELDB (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL))
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
))

; Error handling: Missing the word that needs to be recalled,
; just skip it and move on
(RFR-UPDATE-REHEARSAL--PRODUCE--LAST-NEW-NO-ITEM
 IF (
  ; Time to rehearse an item
  (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL)

  ; We actually have an item to articulate
  (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER NIL ?SOURCE)

  ; This is the last item to rehearse
  (NOT (TAG RFR-TRIAL ??? IS OVERT NEW))
  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)

  ; Ready to speak
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  ; Clean up the step and tags
  (DELDB (STEP RFR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
  (DELDB (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
  (DELDB (TAG RFR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT
?SOURCE))

  ; Stop the rehearsal method
  (DELDB (GOAL DO RFR-UPDATE-REHEARSAL))
  (DELDB (STATUS RFR-UPDATE-REHEARSAL REHEARSING))
  (DELDB (STEP RFR-UPDATE-REHEARSAL CONTINUE REHEARSAL))
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; RFR Update Recall (RFR-UPDATE-RECALL)
;;
;; RFR rules for recalling the updated sequence on a subtrial
;; Takes the covertly rehearsed sequence as the recall chain
;; Prepares the first word to be said in advance
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
; Prepare the first item to recall as soon as the covert rehearsal
; has been completed and heard.
(RFR-UPDATE-RECALL--PREPARE
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL PREPARE RECALL)

  ; Wait until we've heard all of the covert rehearsal before doing
  ; this, because we may need the item to prep it...
  (NOT (STATUS RFR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))

  ; Only do this if we are still waiting for the stimulus, otherwise,
  ; just say it, don't prep it first
  (STEP RFR-UPDATE-RECALL WAIT-FOR STIMULUS)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE COVERT MARKER ?MARKER TYPE ???)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS RFR-REHEARSED-CHAIN-START)
  (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW)
 ) THEN (
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG RFR-UPDATE-RECALL TO-BE-PREPPED-CONTENT IS PREPARE START ^TO-BE-RECALLED-
CONTENT))

  ; Do the prep
  (ADDDB (STEP RFR-UPDATE-RECALL DO-PREPARE RECALL))
  (DELDB (STEP RFR-UPDATE-RECALL PREPARE RECALL))
))

; Skip the preparation if it is already time to begin the recall
(RFR-UPDATE-RECALL--PREPARE--CANCEL
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL PREPARE RECALL)

  ; Wait until we've heard all of the covert rehearsal before doing
  ; this, because we may need the item to prep it...
  (NOT (STATUS RFR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))

  ; It's too late, so don't do the prep
  (NOT (STEP RFR-UPDATE-RECALL WAIT-FOR STIMULUS))
 ) THEN (
  (DELDB (STEP RFR-UPDATE-RECALL PREPARE RECALL))
))

; Send the preparation command to the Vocal Motor processor
; if we have the content
(RFR-UPDATE-RECALL--PREPARE--PREPARE
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL DO-PREPARE RECALL)

  (TAG RFR-UPDATE-RECALL TO-BE-PREPPED-CONTENT IS PREPARE START ?CONTENT)
  (DIFFERENT ?CONTENT NIL)

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR-F 'VOCAL 'PREPARE 'START (UNTRUNC ?CONTENT))

  ; Clean up
  (DELDB (STEP RFR-UPDATE-RECALL DO-PREPARE RECALL))
  (DELDB (TAG RFR-UPDATE-RECALL TO-BE-PREPPED-CONTENT IS PREPARE START ?CONTENT))
))

; Silently fail and cleanup if the content isn't available for prep
(RFR-UPDATE-RECALL--PREPARE--NO-ITEM
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL DO-PREPARE RECALL)

  (TAG RFR-UPDATE-RECALL TO-BE-PREPPED-CONTENT IS PREPARE START NIL)
 ) THEN (
  ; Clean up
  (DELDB (STEP RFR-UPDATE-RECALL DO-PREPARE RECALL))
  (DELDB (TAG RFR-UPDATE-RECALL TO-BE-PREPPED-CONTENT IS PREPARE START NIL))
))

; Need to mark new stimulus as final item and begin recall...
(RFR-UPDATE-RECALL--HEAR-NEW-STIMULUS
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL WAIT-FOR STIMULUS)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ??? TYPE ???)
  (NOT (TAG RFR-TRIAL ?ITEM-ID IS ??? ???))
 ) THEN (
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW))

  ; Begin recall
  (DELDB (STEP RFR-UPDATE-RECALL WAIT-FOR STIMULUS))
  (ADDDB (STATUS RFR-UPDATE-RECALL STARTING))
  (ADDDB (STEP RFR-UPDATE-RECALL BEGIN RECALL))

  ; Launch thread to hear own speech
  (ADDDB (GOAL DO RFR-HEAR-SEQUENCE OVERT))

  ; Launch thread to listen for feedback
  (ADDDB (GOAL DO RFR-HEAR-FEEDBACK))
))

; Marks the start as the NEXT-TO-RECALL and sends it to PRODUCE
(RFR-UPDATE-RECALL--BEGIN
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL BEGIN RECALL)
  ; Wait for rehearsal to end before starting...
  (NOT (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
  (NOT (STATUS RFR-UPDATE-REHEARSAL REHEARSING))

  ; Should we wait until we hear the covert sequence end???
  (NOT (STATUS RFR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE COVERT MARKER ?MARKER TYPE ???)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS RFR-REHEARSED-CHAIN-START)
  (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  ; Get the content
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT COVERT))

  ; Send the first item to be produced
  (DELDB (STEP RFR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL PRODUCE RECALL))
))

; Error handling: Object missing
(RFR-UPDATE-RECALL--BEGIN--NO-OBJECT
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL BEGIN RECALL)
  ; Wait for rehearsal to end before starting...
  (NOT (STATUS RFR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
  (NOT (STATUS RFR-UPDATE-REHEARSAL REHEARSING))

  ; Should we wait until we hear the covert sequence end???
  (NOT (STATUS RFR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))

  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE COVERT MARKER ??? TYPE ???))
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS RFR-REHEARSED-CHAIN-START)
  (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL))

  ; Jump into the main chain recall sequence
  (DELDB (STEP RFR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL CONTINUE RECALL))
))

; Continue rehearsing the current sequence by retrieving content for
; next to recall
(RFR-UPDATE-RECALL--CONTINUE
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ?SOURCE MARKER ?MARKER TYPE ???)
  )
 THEN
 (
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT ?SOURCE))

  (DELDB (STEP RFR-UPDATE-RECALL CONTINUE RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL PRODUCE RECALL))
))

; Error handling: Object missing
(RFR-UPDATE-RECALL--CONTINUE--NO-OBJECT
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO RFR-REBUILD-CHAINS COVERT))
  (ADDDB (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP RFR-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: Next item is not in the recall chain
(RFR-UPDATE-RECALL--CONTINUE--NO-NEW
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (DIFFERENT ?ITEM-ID GONE)
  (NOT (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO RFR-REBUILD-CHAINS COVERT))
  (ADDDB (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP RFR-REBUILD-CHAINS MARK ITEMS))
))
```

```
; Error handling: Missing the link to next object
(RFR-UPDATE-RECALL--CONTINUE--NEXT-TO-RECALL-IS-GONE
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (EQUAL ?ITEM-ID GONE)
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO RFR-REBUILD-CHAINS COVERT))
  (ADDDB (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP RFR-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: No objects left to recall
(RFR-UPDATE-RECALL--CONTINUE--NO-NEXT-TO-RECALL
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS RFR-REBUILD-CHAINS REBUILDING CHAINS))

  (NOT (TAG RFR-TRIAL ??? IS NEXT-TO-RECALL))
 ) THEN (
  ; We don't have anymore objects to recall
  (DELDB (STEP RFR-UPDATE-RECALL CONTINUE RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL END RECALL))
))

; Say the first word that needs to be recalled, if it is a covert item
(RFR-UPDATE-RECALL--PRODUCE--START
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL PRODUCE RECALL)
  (STATUS RFR-UPDATE-RECALL STARTING)

  ; Assumes that content is available
  (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COVERT)
  (MOTOR VOCAL MODALITY FREE)
 ) THEN (
  (SEND-TO-MOTOR-F 'VOCAL ?STYLE 'START (UNTRUNC ?CONTENT))

  (DELDB (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COV-
ERT))

  (DELDB (STEP RFR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL FIND NEXT))

  (DELDB (STATUS RFR-UPDATE-RECALL STARTING))
))

; Say the first word that needs to be recalled, if it is the add item
(RFR-UPDATE-RECALL--PRODUCE--START-ADD-OBJECT
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL PRODUCE RECALL)
  (STATUS RFR-UPDATE-RECALL STARTING)

  ; Assumes that content is available
  (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTERNAL)
  (MOTOR VOCAL MODALITY FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE START ?CONTENT)

  (DELDB (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTER-
NAL))
```

```
  (DELDB (STEP RFR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL FIND NEXT))

  (DELDB (STATUS RFR-UPDATE-RECALL STARTING))
))

; Say the next word that needs to be recalled, if it is a covert item
(RFR-UPDATE-RECALL--PRODUCE--CONTINUE
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL PRODUCE RECALL)
  (NOT (STATUS RFR-UPDATE-RECALL STARTING))

  ; Assumes that content is available
  (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COVERT)
  (MOTOR VOCAL MODALITY FREE)
 ) THEN (
  (SEND-TO-MOTOR-F 'VOCAL ?STYLE 'CONTINUE (UNTRUNC ?CONTENT))

  (DELDB (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COV-
ERT))

  (DELDB (STEP RFR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL FIND NEXT))
))

; Say the next word that needs to be recalled, if it is the add item
(RFR-UPDATE-RECALL--PRODUCE--CONTINUE-ADD-OBJECT
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL PRODUCE RECALL)
  (NOT (STATUS RFR-UPDATE-RECALL STARTING))

  ; Assumes that content is available
  (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTERNAL)
  (MOTOR VOCAL MODALITY FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE CONTINUE ?CONTENT)

  (DELDB (TAG RFR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTER-
NAL))

  (DELDB (STEP RFR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP RFR-UPDATE-RECALL FIND NEXT))
))

; Find the next object in the recall chain
(RFR-UPDATE-RECALL--FIND-NEXT
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL FIND NEXT)

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
  (NOT (TAG RFR-RECALL ?NEXT-ID IS COVERT STIMULUS-END))

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW))
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT RECALLED))

  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

  (DELDB (STEP RFR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RFR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Current object missing
(RFR-UPDATE-RECALL--FIND-NEXT--OBJECT-MISSING
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL FIND NEXT)

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW))
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT RECALLED))

  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (STEP RFR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RFR-UPDATE-RECALL CONTINUE RECALL))
))

; We've reached the end of the recall chain, so its time for the
; add object
(RFR-UPDATE-RECALL--FIND-NEXT--ADD-NEW-OBJECT
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL FIND NEXT)

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
  (TAG RFR-RECALL ?NEXT-ID IS COVERT STIMULUS-END)

  (AUDITORY SPEECH-TAG ?UNUM ?ADD-ID IS ADD-CHAIN-START)

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW))
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT RECALLED))

  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL ?ADD-ID IS NEXT-TO-RECALL))

  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)

  (DELDB (STEP RFR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RFR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Tag for the add object is missing
(RFR-UPDATE-RECALL--FIND-NEXT--NO-ADD-NEW-OBJECT
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL FIND NEXT)

  (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE ??? MARKER ??? TYPE ???)
  (TAG RFR-RECALL ?NEXT-ID IS COVERT STIMULUS-END)

  (NOT (AUDITORY SPEECH-TAG ??? ??? IS ADD-CHAIN-START))

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT NEW))
  (ADDDB (TAG RFR-TRIAL ?ITEM-ID IS COVERT RECALLED))

  (DELDB (TAG RFR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RFR-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (STEP RFR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RFR-UPDATE-RECALL CONTINUE RECALL))
))

; Cleanup after recall: Remove the STIMULUS-END tag
(RFR-UPDATE-RECALL--CLEANUP--REMOVE-STIMULUS-END
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL END RECALL)

  (TAG RFR-RECALL ?ITEM-ID IS COVERT STIMULUS-END)
 ) THEN (
  (DELDB (TAG RFR-RECALL ?ITEM-ID IS COVERT STIMULUS-END))
))

; Cleanup after recall: Remove the ADD-CHAIN-START tag
(RFR-UPDATE-RECALL--CLEANUP--REMOVE-ADD-CHAIN-START
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL END RECALL)

  (AUDITORY SPEECH-TAG ?UNUM ??? IS ADD-CHAIN-START)
 ) THEN (
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
))

; Wrap up the update recall
(RFR-UPDATE-RECALL--END
 IF (
  (GOAL DO RFR-UPDATE-RECALL)
  (STEP RFR-UPDATE-RECALL END RECALL)
 ) THEN (
  (DELDB (STEP RFR-UPDATE-RECALL END RECALL))
  (DELDB (GOAL DO RFR-UPDATE-RECALL))
))
```

C.3 CBR Postponement Model

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rules for the Cumulative Backward Rehearsal (CBR) task.
;; These rules implement the Postponement strategy for CBR.
;;
;; Only the CBR Update Recall rules are listed, as the rest of the
;; rules are substantially the same as those for the CFR Guessing
;; strategy.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Identifies start of a block of CBR and sets the goal to perform it
(R-BLOCK--IDENTIFY-TASK-AS-CBR
 IF (
  (GOAL DO REH-TASK)
  (STEP R-TASK IDENTIFY BLOCK-TYPE)

  (VISUAL ?OBJECT LABEL CBR)
 ) THEN (
  (DELDB (GOAL DO REH-TASK))
  (ADDDB (GOAL DO R-TASK CBR))

  (DELDB (STEP R-TASK IDENTIFY BLOCK-TYPE))
  (ADDDB (STEP R-TASK BEGIN TRIAL))
))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; CBR Update Recall (CBR-UPDATE-RECALL)
;;
;; CBR rules for recalling the updated sequence on a subtrial
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Preparation for an update subtrial
(CBR-UPDATE-RECALL--PREPARE
 IF (
  (GOAL DO R-TRIAL CBR)
  (STATUS R-HEAR-FEEDBACK CORRECT)
  (NOT (GOAL DO CBR-UPDATE-RECALL))
 ) THEN (
  (DELDB (STATUS R-HEAR-FEEDBACK CORRECT))

  (ADDDB (GOAL DO CBR-UPDATE-RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL WAIT-FOR STIMULUS))
))


; Hear the new word presented on each subtrial
; We want to hear the entire word, since we are going to be saying
; it as the next action.
(CBR-UPDATE-RECALL--HEAR-NEW-STIMULUS
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL WAIT-FOR STIMULUS)

  ; ** Wait until the entire stimulus has been perceived
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ??? TYPE ???)
  (NOT (TAG R-TRIAL ?ITEM-ID IS ??? ???))
 ) THEN (
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT))

  ; Update tags
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  ; Get the content
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY START ^TO-BE-RECALLED-
CONTENT))

  ; Begin recall
  (DELDB (STEP CBR-UPDATE-RECALL WAIT-FOR STIMULUS))
  (ADDDB (STATUS CBR-UPDATE-RECALL STARTING))
  (ADDDB (STEP CBR-UPDATE-RECALL PRODUCE RECALL))

  ; Launch thread to hear own speech
  (ADDDB (GOAL DO R-HEAR-SEQUENCE OVERT))

  ; Launch thread to listen for feedback
  (ADDDB (GOAL DO R-HEAR-FEEDBACK))
))

; Marks the start of recall chain as the NEXT-TO-RECALL and
; begins regular algorithm
(CBR-UPDATE-RECALL--BEGIN
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL BEGIN RECALL)

  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ?MARKER TYPE ???)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS REHEARSED-CHAIN-START)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  ; Get the content
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT))

  ; Send the first item to be produced
  (DELDB (STEP CBR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL PRODUCE RECALL))
))

; Error handling: The first speech object is missing
```

```
(CBR-UPDATE-RECALL--BEGIN--NO-OBJECT
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL BEGIN RECALL)

  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ??? TYPE ???))
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS REHEARSED-CHAIN-START)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
 ) THEN (
  ; Update tags
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  ; Jump into the main chain recall sequence
  (DELDB (STEP CBR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: REHEARSED-CHAIN-START tag is missing
(CBR-UPDATE-RECALL--BEGIN--NO-CHAIN-START
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL BEGIN RECALL)

  (NOT (AUDITORY SPEECH-TAG ??? ??? IS R-REHEARSED-CHAIN-START))
 ) THEN (
  ; Update tags
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  ; Jump into the main chain recall sequence
  (DELDB (STEP CBR-UPDATE-RECALL BEGIN RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
))

; Continue rehearsing the current sequence by retrieving content
; for next to recall
(CBR-UPDATE-RECALL--CONTINUE
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ?MARKER TYPE ???)
 ) THEN (
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
CONTENT))

  (DELDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL PRODUCE RECALL))
))

; Error handling: Speech object is missing
(CBR-UPDATE-RECALL--CONTINUE--NO-OBJECT
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS OVERT))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
```

```
))

; Error handling: The next object is not in the recall chain
(CBR-UPDATE-RECALL--CONTINUE--NO-CURRENT
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (DIFFERENT ?ITEM-ID GONE)
  (NOT (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS OVERT))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: We lost the link to the next object
(CBR-UPDATE-RECALL--CONTINUE--NEXT-TO-RECALL-IS-GONE
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (EQUAL ?ITEM-ID GONE)
 ) THEN (
  ; We don't have a next object, so we drop into chain building
  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (ADDDB (GOAL DO R-REBUILD-CHAINS OVERT))
  (ADDDB (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))
  (ADDDB (STEP R-REBUILD-CHAINS MARK ITEMS))
))

; Error handling: No more speech objects to recall
(CBR-UPDATE-RECALL--CONTINUE--NO-NEXT-TO-RECALL
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL CONTINUE RECALL)
  (NOT (STATUS R-REBUILD-CHAINS REBUILDING CHAINS))

  (NOT (TAG R-TRIAL ??? IS NEXT-TO-RECALL))
 ) THEN (
  ; We don't have anymore objects to recall
  (DELDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL END RECALL))
))

; Say the first word that needs to be recalled
(CBR-UPDATE-RECALL--PRODUCE--START
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL PRODUCE RECALL)
  (STATUS CBR-UPDATE-RECALL STARTING)

  ; Assumes that content is available
  (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT)
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE START ?CONTENT)

  (DELDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT))

  (DELDB (STEP CBR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL FIND NEXT))
  (DELDB (STATUS CBR-UPDATE-RECALL STARTING))
))

; Say the next word that needs to be recalled
(CBR-UPDATE-RECALL--PRODUCE--CONTINUE
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL PRODUCE RECALL)
  (NOT (STATUS CBR-UPDATE-RECALL STARTING))

  ; Assumes that content is available
  (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT)
  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE CONTINUE ?CONTENT)

  (DELDB (TAG R-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT))

  (DELDB (STEP CBR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP CBR-UPDATE-RECALL FIND NEXT))
))

; Find the next object in the recall chain
(CBR-UPDATE-RECALL--FIND-NEXT
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL FIND NEXT)

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ??? TYPE ???)
  (NOT (TAG R-RECALL ?NEXT-ID IS OVERT CURRENT-STIMULUS-END))
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG R-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

  (DELDB (STEP CBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Speech object is missing
(CBR-UPDATE-RECALL--FIND-NEXT--OBJECT-MISSING
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL FIND NEXT)

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (STEP CBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
))

; After the new object, we go to the recall chain
(CBR-UPDATE-RECALL--FIND-NEXT--REHEARSED-CHAIN-START
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL FIND NEXT)

  ; We just said the add object...
  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT)

  (AUDITORY SPEECH-TAG ?UNUM ?NEXT-ID IS REHEARSED-CHAIN-START)

 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG R-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)

  (DELDB (STEP CBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Time for the recall chain, but the tag is missing
(CBR-UPDATE-RECALL--FIND-NEXT--REHEARSED-CHAIN-START--MISSING
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL FIND NEXT)

  ; We just said the add object...
  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT)

  (NOT (AUDITORY SPEECH-TAG ??? ??? IS REHEARSED-CHAIN-START))
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS EXTERNAL RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG R-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (STEP CBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP CBR-UPDATE-RECALL CONTINUE RECALL))
))

; We've reached the end of the recall chain
(CBR-UPDATE-RECALL--FIND-NEXT--CURRENT-STIMULUS-END
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL FIND NEXT)

  (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ??? TYPE ???)
  (TAG R-RECALL ?NEXT-ID IS OVERT CURRENT-STIMULUS-END)
 ) THEN (
  (DELDB (TAG R-TRIAL ?ITEM-ID IS OVERT CURRENT))
  (ADDDB (TAG R-TRIAL ?ITEM-ID IS OVERT RECALLED))

  (DELDB (TAG R-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (DELDB (STEP CBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP CBR-UPDATE-RECALL END RECALL))
))


; Cleanup after recall: Remove the CURRENT-STIMULUS-END tag
(CBR-UPDATE-RECALL--CLEANUP--REMOVE-CURRENT-STIMULUS-END
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL END RECALL)

  (TAG R-RECALL ?ITEM-ID IS OVERT CURRENT-STIMULUS-END)
 ) THEN (
  (DELDB (TAG R-RECALL ?ITEM-ID IS OVERT CURRENT-STIMULUS-END))
))
```

```
; Wrap up recall - we are already waiting for feedback
(CBR-UPDATE-RECALL--END
 IF (
  (GOAL DO CBR-UPDATE-RECALL)
  (STEP CBR-UPDATE-RECALL END RECALL)
 ) THEN (
  (DELDB (STEP CBR-UPDATE-RECALL END RECALL))
  (DELDB (GOAL DO CBR-UPDATE-RECALL))
))
```

C.4 RBR Restrained-Rehearsal Model

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Rules for the Rolling Backward Rehearsal (RBR) task.
;; These rules implement the Restrained Rehearsal strategy for RBR.
;;
;; Only the RBR Update Recall rules are listed, as the rest of the
;; rules are substantially the same as those for the RFR Link-Decay
;; strategy.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Identifies the start of a block of RBR and sets the goal to
; perform it
(R-BLOCK--IDENTIFY-TASK-AS-RBR
 IF (
  (GOAL DO REH-TASK)
  (STEP R-TASK IDENTIFY BLOCK-TYPE)

  (VISUAL ?OBJECT LABEL RBR)
 ) THEN (
  (DELDB (STEP REH-TASK IDENTIFY BLOCK-TYPE))
  (ADDDB (GOAL DO RBR-TASK))
))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; RBR Update Rehearsal (RBR-UPDATE-REHEARSAL)
;;
;; RBR rules for covert truncated rehearsal on a subtrial. The
;; last word is skipped, and the new word is not yet added. The
;; rehearsal here is less aggressive than it was for RFR.
;;
;; Rehearsal uses a simple form of recovery in case of lost links
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Step over the first item in the sequence
(RBR-UPDATE-REHEARSAL--BEGIN
 IF (
  ; Time to do covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL BEGIN REHEARSAL)

  ; Find the first item in the chain to be rehearsed
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ?MARKER TYPE ???)
  (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS REHEARSED-CHAIN-START)
 ) THEN (
  ; Get rid of the chain start tag, and mark the next word to rehearse
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT RECALLED))
  (ADDDB (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-REHEARSE))

  ; Get the phonological item to rehearse, and set up the
  ; subvocalization of it
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS SUBVOCALIZE START ^TO-BE-
RECALLED-CONTENT OVERT))
  (DELDB (STEP RBR-UPDATE-REHEARSAL BEGIN REHEARSAL))
  (ADDDB (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL))
  (ADDDB (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
  (ADDDB (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; Flag that rehearsal is ongoing
  (ADDDB (STATUS RBR-UPDATE-REHEARSAL REHEARSING))

  ; Get a thread going to hear self rehearse
  (ADDDB (GOAL DO RBR-HEAR-SEQUENCE COVERT))

  ; Start waiting to hear the new item, and be ready to prepare recall
  (ADDDB (GOAL DO RBR-UPDATE-RECALL))
  (ADDDB (STEP RBR-UPDATE-RECALL WAIT-FOR STIMULUS))

  ; Set a process up to listen for feedback
  (ADDDB (GOAL DO RBR-HEAR-FEEDBACK))
))

; Error handling: Object is missing
(RBR-UPDATE-REHEARSAL--BEGIN--NO-OBJECT
 IF (
  ; Time to do covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL BEGIN REHEARSAL)

  ; We have the tag pointing to the first item, but no first item
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ??? TYPE ???))
  (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW)
  (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS REHEARSED-CHAIN-START)
 ) THEN (
  ; Get rid of the chain start tag
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT RECALLED))
  (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-REHEARSE))

  ; We can't say the first word, so just continue on
  (DELDB (STEP RBR-UPDATE-REHEARSAL BEGIN REHEARSAL))
  (ADDDB (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL))

  ; Flag that rehearsal is ongoing
  (ADDDB (STATUS RBR-UPDATE-REHEARSAL REHEARSING))

  ; Get a thread going to hear self rehearse
  (ADDDB (GOAL DO RBR-HEAR-SEQUENCE COVERT))

  ; Start waiting to hear the new item, and be ready to prepare recall
  (ADDDB (GOAL DO RBR-UPDATE-RECALL))
  (ADDDB (STEP RBR-UPDATE-RECALL WAIT-FOR STIMULUS))

  ; Set a process up to listen for feedback
  (ADDDB (GOAL DO RBR-HEAR-FEEDBACK))
))

; Error handling: Missing REHEARSED-CHAIN-START tag
(RBR-UPDATE-REHEARSAL--BEGIN--NO-CHAIN-START
 IF (
  ; Time to do covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL BEGIN REHEARSAL)

  (NOT (AUDITORY SPEECH-TAG ??? ??? IS REHEARSED-CHAIN-START))
 ) THEN (
  (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-REHEARSE))

  ; We can't say the first word, so just continue on
  (DELDB (STEP RBR-UPDATE-REHEARSAL BEGIN REHEARSAL))
  (ADDDB (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL))
  ; Flag that rehearsal is ongoing
  (ADDDB (STATUS RBR-UPDATE-REHEARSAL REHEARSING))

  ; Get a thread going to hear self rehearse
  (ADDDB (GOAL DO RBR-HEAR-SEQUENCE COVERT))

  ; Start waiting to hear the new item, and be ready to prepare recall
  (ADDDB (GOAL DO RBR-UPDATE-RECALL))
  (ADDDB (STEP RBR-UPDATE-RECALL WAIT-FOR STIMULUS))

  ; Set a process up to listen for feedback
  (ADDDB (GOAL DO RBR-HEAR-FEEDBACK))
))

; Error handling: Restart rehearsal when a link is missing
(RBR-UPDATE-REHEARSAL--RESTART
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We don't have the next speech object
  (TAG RBR-TRIAL GONE IS NEXT-TO-REHEARSE)

  ; Need to avoid picking the last item!!!
  ; We do have NEW tags remaining
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ?MARKER TYPE ???)
  (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW)
  (NOT (TAG RBR-RECALL ?NEXT-ID IS OVERT STIMULUS-END))
  (RANDOMLY-CHOOSE-ONE ?ITEM-ID)
 ) THEN (
  ;Move the pointer to the next item
  (DELDB (TAG RBR-TRIAL GONE IS NEXT-TO-REHEARSE))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE))
))


; Error handling: Restart rehearsal when we are at the end
(RBR-UPDATE-REHEARSAL--RESTART--FROM-STIMULUS-END
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We have the next speech object, but it is the last object
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ??? TYPE ???)
  (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW)
  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE)
  (TAG RBR-RECALL ?NEXT-ID IS OVERT STIMULUS-END)

  (AUDITORY SPEECH ITEM-ID ?NEW-ID NEXT ?NEW-NEXT-ID SOURCE OVERT MARKER ??? TYPE ???)
  (TAG RBR-TRIAL ?NEW-ID IS OVERT NEW)
  (NOT (TAG RBR-RECALL ?NEW-NEXT-ID IS OVERT STIMULUS-END))
  (RANDOMLY-CHOOSE-ONE ?NEW-ID)
 ) THEN (
  ;Move the pointer to the next item
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE))
  (ADDDB (TAG RBR-TRIAL ?NEW-ID IS NEXT-TO-REHEARSE))
))

; Continue rehearsing the current sequence
(RBR-UPDATE-REHEARSAL--CONTINUE
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We have the next speech object
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE OVERT MARKER ?MARKER TYPE ???)
  (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW)
  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE)
  (NOT (TAG RBR-RECALL ?NEXT-ID IS OVERT STIMULUS-END))
 ) THEN (
  ; Get the phonological item to rehearse, and set up the subvocalization of it
  (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
  (ADDDB (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS SUBVOCALIZE CONTINUE ^TO-BE-
RECALLED-CONTENT OVERT))
  (ADDDB (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
  (ADDDB (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ;Move the pointer to the next item
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE))
  (ADDDB (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-REHEARSE))

  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT RECALLED))
))

; Error handling: Object is missing
(RBR-UPDATE-REHEARSAL--CONTINUE--NO-OBJECT
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We don't have the next speech object
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE OVERT MARKER ??? TYPE ???))
  (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW)
  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE)
 ) THEN (
  ; Get rid of the chain start tag
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE))
  (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-REHEARSE))

  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS OVERT RECALLED))
))

; Error handling: Next object not in recall chain
(RBR-UPDATE-REHEARSAL--CONTINUE--NO-NEW
 IF (
  ; We are doing covert rehearsal
  (GOAL DO RBR-UPDATE-REHEARSAL)
  (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL)
  ; We aren't articulating the previous item
  (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))

  ; We don't have the next speech object
  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE)
  (DIFFERENT ?ITEM-ID GONE)
  (NOT (TAG RBR-TRIAL ?ITEM-ID IS OVERT NEW))
 ) THEN (
  ; Change the next marker to GONE
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-REHEARSE))
  (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-REHEARSE))
))

; Subvocalize the word that needs to be rehearsed
(RBR-UPDATE-REHEARSAL--PRODUCE
 IF (
  ; Time to rehearse an item
  (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL)
```

```
   ; We actually have an item to articulate
   (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT ?SOURCE)
   (DIFFERENT ?CONTENT NIL)

   ; There are at least two items left, so this isn't the
   ; last to be said!
   (TAG RBR-TRIAL ?ITEM-ID-1 IS OVERT NEW)
   (TAG RBR-TRIAL ?ITEM-ID-2 IS OVERT NEW)
   (DIFFERENT ?ITEM-ID-1 ?ITEM-ID-2)
   (USE-ONLY-ONE ?ITEM-ID-1) ; Insures that multiple instantiations
                             ; don't occur

   ; Ready to speak
   (MOTOR VOCAL MODALITY FREE)
  ) THEN (
   ; Changed to use truncated articulation!!!!
   (SEND-TO-MOTOR-F 'VOCAL ?STYLE ?MARKER (TRUNC ?CONTENT))

   ; Clean up the step and tags
   (DELDB (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
   (DELDB (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
   (DELDB (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT
?SOURCE))
))

; Error handling: Missing the word that needs to be recalled,
; just skip it and move on
(RBR-UPDATE-REHEARSAL--PRODUCE--NO-ITEM
 IF (
   ; Time to rehearse an item
   (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL)

   ; The item to articulate is missing
   (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER NIL ?SOURCE)

   ; There are at least two items left, so this isn't the last
   ; to be said!
   (TAG RBR-TRIAL ?ITEM-ID-1 IS OVERT NEW)
   (TAG RBR-TRIAL ?ITEM-ID-2 IS OVERT NEW)
   (DIFFERENT ?ITEM-ID-1 ?ITEM-ID-2)

   ; Ready to speak
   (MOTOR VOCAL MODALITY FREE)
  ) THEN (
   ; Clean up the step and tags
   (DELDB (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
   (DELDB (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
   (DELDB (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER NIL ?SOURCE))
))

; Subvocalize the last word that needs to be rehearsed
(RBR-UPDATE-REHEARSAL--PRODUCE--LAST-NEW
 IF (
   ; Time to rehearse an item
   (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL)

   ; We actually have an item to articulate
   (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT ?SOURCE)
   (DIFFERENT ?CONTENT NIL)

   ; This is the last item to rehearse, because there is exactly
   ; one other NEW item around
   (TAG RBR-TRIAL ?LAST-ID IS OVERT NEW)
   (IF-ONLY-ONE ?LAST-ID)

   ; Something is next?
   (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-REHEARSE)

   ; Ready to speak
   (MOTOR VOCAL MODALITY FREE)
  ) THEN (
   ; Changed to use truncated articulation!!!!
   (SEND-TO-MOTOR-F 'VOCAL ?STYLE 'END (TRUNC ?CONTENT))

   ; Clean up the step and tags
   (DELDB (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
   (DELDB (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
   (DELDB (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT
 ?SOURCE))

   ; Stop the rehearsal method
   (DELDB (GOAL DO RBR-UPDATE-REHEARSAL))
   (DELDB (STATUS RBR-UPDATE-REHEARSAL REHEARSING))
   (DELDB (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL))
   (DELDB (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-REHEARSE))
   (DELDB (TAG RBR-TRIAL ?LAST-ID IS OVERT NEW))
   (ADDDB (TAG RBR-TRIAL ?LAST-ID IS OVERT RECALLED))
 ))

 ; Error handling: Missing the word that needs to be recalled,
 ; just skip it and move on
 (RBR-UPDATE-REHEARSAL--PRODUCE--LAST-NEW-NO-ITEM
  IF (
   ; Time to rehearse an item
   (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL)

   ; We actually have an item to articulate
   (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER NIL ?SOURCE)

   ; This is the last item to rehearse, because there is exactly
   ; one other NEW item around
   (TAG RBR-TRIAL ?LAST-ID IS OVERT NEW)
   (IF-ONLY-ONE ?LAST-ID)

   ; Something is next
   (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-REHEARSE)

   ; Ready to speak
   (MOTOR VOCAL MODALITY FREE)
  ) THEN (
   ; Clean up the step and tags
   (DELDB (STEP RBR-UPDATE-REHEARSAL PRODUCE REHEARSAL))
   (DELDB (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
   (DELDB (TAG RBR-UPDATE-REHEARSAL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT
 ?SOURCE))

   ; Stop the rehearsal method
   (DELDB (GOAL DO RBR-UPDATE-REHEARSAL))
   (DELDB (STATUS RBR-UPDATE-REHEARSAL REHEARSING))
   (DELDB (STEP RBR-UPDATE-REHEARSAL CONTINUE REHEARSAL))
   (DELDB (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-REHEARSE))
   (DELDB (TAG RBR-TRIAL ?LAST-ID IS OVERT NEW))
   (ADDDB (TAG RBR-TRIAL ?LAST-ID IS OVERT RECALLED))
 ))


 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 ;;
 ;; RBR Update Recall (RBR-UPDATE-RECALL)
 ;;
 ;; RBR rules for recalling the updated sequence on a subtrial
 ;; Takes the covertly rehearsed sequence as the recall chain
 ;; Prepares the first word to be said in advance
 ;;
 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 ; In RBR, the new item is going to be said first.
 (RBR-UPDATE-RECALL--HEAR-NEW-STIMULUS
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL WAIT-FOR STIMULUS)
   (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE EXTERNAL MARKER ??? TYPE ???)
   (NOT (TAG RBR-TRIAL ?ITEM-ID IS ??? ???))

   ; Wait until we hear the covert sequence end
   (NOT (STATUS RBR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))
  ) THEN (
   (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS EXTERNAL NEW))
   (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

   (DELDB (STEP RBR-UPDATE-RECALL WAIT-FOR STIMULUS))

   ; Get the content
   (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
   (ADDDB (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY START ^TO-BE-RECALLED-
 CONTENT EXTERNAL))

   ; Begin recall
   (ADDDB (STATUS RBR-UPDATE-RECALL STARTING))
   (ADDDB (STEP RBR-UPDATE-RECALL PRODUCE RECALL))

   ; Launch thread to hear own speech
   (ADDDB (GOAL DO RBR-HEAR-SEQUENCE OVERT))
 ))

 ; Marks the start as the NEXT-TO-RECALL and sends it to PRODUCE
 (RBR-UPDATE-RECALL--BEGIN
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL BEGIN RECALL)
   ; Wait for rehearsal to end before starting...
   (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
   (NOT (STATUS RBR-UPDATE-REHEARSAL REHEARSING))

   ; Should we wait until we hear the covert sequence end???
   (NOT (STATUS RBR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))

   (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE COVERT MARKER ?MARKER TYPE ???)
   (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS REHEARSED-CHAIN-START)
   (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW)
  ) THEN (
   ; Update tags
   (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
   (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

   ; Get the content
   (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
   (ADDDB (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
 CONTENT COVERT))

   ; Send the first item to be produced
   (DELDB (STEP RBR-UPDATE-RECALL BEGIN RECALL))
   (ADDDB (STEP RBR-UPDATE-RECALL PRODUCE RECALL))
 ))

 ; Error handling: Object missing
 (RBR-UPDATE-RECALL--BEGIN--NO-OBJECT
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL BEGIN RECALL)
   ; Wait for rehearsal to end before starting...
   (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
   (NOT (STATUS RBR-UPDATE-REHEARSAL REHEARSING))

   ; Should we wait until we hear the covert sequence end???
   (NOT (STATUS RBR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))

   (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE COVERT MARKER ??? TYPE ???))
   (AUDITORY SPEECH-TAG ?UNUM ?ITEM-ID IS REHEARSED-CHAIN-START)
   (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW)
  ) THEN (
   ; Update tags
   (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
   (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-RECALL))

   ; Jump into the main chain recall sequence
   (DELDB (STEP RBR-UPDATE-RECALL BEGIN RECALL))
   (ADDDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
 ))

 ; Error handling: Missing start of chain
 (RBR-UPDATE-RECALL--BEGIN--NO-CHAIN-START
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL BEGIN RECALL)
   ; Wait for rehearsal to end before starting...
   (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
   (NOT (STATUS RBR-UPDATE-REHEARSAL REHEARSING))

   ; Should we wait until we hear the covert sequence end???
   (NOT (STATUS RBR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))

   (NOT (AUDITORY SPEECH-TAG ??? ??? IS REHEARSED-CHAIN-START))
  ) THEN (
   ; Update tags
   (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-RECALL))

   ; Jump into the main chain recall sequence
   (DELDB (STEP RBR-UPDATE-RECALL BEGIN RECALL))
   (ADDDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
 ))

 ; Continue rehearsing the current sequence by retrieving content
 ; for next to recall
 (RBR-UPDATE-RECALL--CONTINUE
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL CONTINUE RECALL)
   (NOT (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))

   (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
   (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW)
   (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ?SOURCE MARKER ?MARKER TYPE ???)
  ) THEN (
   (RETRIEVE-PHONOLOGICAL-INFORMATION ^TO-BE-RECALLED-CONTENT ?ITEM-ID)
   (ADDDB (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS SAY ?MARKER ^TO-BE-RECALLED-
 CONTENT ?SOURCE))

   (DELDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
   (ADDDB (STEP RBR-UPDATE-RECALL PRODUCE RECALL))
 ))

 ; Error handling: Object missing
 (RBR-UPDATE-RECALL--CONTINUE--NO-OBJECT
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL CONTINUE RECALL)
   (NOT (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))

   (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
   (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW)
   (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))
  ) THEN (
   ; We don't have a next object, so we drop into chain building
   (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

   (ADDDB (GOAL DO RBR-REBUILD-CHAINS COVERT))
   (ADDDB (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))
   (ADDDB (STEP RBR-REBUILD-CHAINS MARK ITEMS))
 ))

 ; Error handling: Next item is not in the recall chain
 (RBR-UPDATE-RECALL--CONTINUE--NO-NEW
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL CONTINUE RECALL)
   (NOT (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))

   (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
   (DIFFERENT ?ITEM-ID GONE)
   (NOT (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW))
  ) THEN (
   ; We don't have a next object, so we drop into chain building
   (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

   (ADDDB (GOAL DO RBR-REBUILD-CHAINS COVERT))
   (ADDDB (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))
   (ADDDB (STEP RBR-REBUILD-CHAINS MARK ITEMS))
 ))

 ; Error handling: Missing the link to next object
 (RBR-UPDATE-RECALL--CONTINUE--NEXT-TO-RECALL-IS-GONE
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL CONTINUE RECALL)
   (NOT (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))

   (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
   (EQUAL ?ITEM-ID GONE)
  ) THEN (
   ; We don't have a next object, so we drop into chain building
   (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

   (ADDDB (GOAL DO RBR-REBUILD-CHAINS COVERT))
   (ADDDB (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))
   (ADDDB (STEP RBR-REBUILD-CHAINS MARK ITEMS))
 ))

 ; Error handling: No objects left to recall
 (RBR-UPDATE-RECALL--CONTINUE--NO-NEXT-TO-RECALL
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL CONTINUE RECALL)
   (NOT (STATUS RBR-REBUILD-CHAINS REBUILDING CHAINS))

   (NOT (TAG RBR-TRIAL ??? IS NEXT-TO-RECALL))
  ) THEN (
   ; We don't have anymore objects to recall
   (DELDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
   (ADDDB (STEP RBR-UPDATE-RECALL END RECALL))
 ))

 ; Say the first word that needs to be recalled, if it is a covert item
 (RBR-UPDATE-RECALL--PRODUCE--START
  IF (
   (GOAL DO RBR-UPDATE-RECALL)
   (STEP RBR-UPDATE-RECALL PRODUCE RECALL)
   (STATUS RBR-UPDATE-RECALL STARTING)

   ; Assumes that content is available
   (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COVERT)
   (MOTOR VOCAL MODALITY FREE)
  ) THEN (
   (SEND-TO-MOTOR-F 'VOCAL ?STYLE 'START (UNTRUNC ?CONTENT))

   (DELDB (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COV-
 ERT))

   (DELDB (STEP RBR-UPDATE-RECALL PRODUCE RECALL))
   (ADDDB (STEP RBR-UPDATE-RECALL FIND NEXT))

   (DELDB (STATUS RBR-UPDATE-RECALL STARTING))
 ))
```

```
; Say the first word that needs to be recalled, if it is the add item
(RBR-UPDATE-RECALL--PRODUCE--START-ADD-OBJECT
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL PRODUCE RECALL)
  (STATUS RBR-UPDATE-RECALL STARTING)

  ; Assumes that content is available
  (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTERNAL)
  (MOTOR VOCAL MODALITY FREE)

  ; Wait for rehearsal to end before starting...
  (NOT (STATUS RBR-UPDATE-REHEARSAL WAITING-FOR REHEARSAL))
  (NOT (STATUS RBR-UPDATE-REHEARSAL REHEARSING))
  ; Should we wait until we hear the covert sequence end???
  (NOT (STATUS RBR-HEAR-SEQUENCE COVERT WAITING-FOR STIMULI))
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE START ?CONTENT)

  (DELDB (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTER-
NAL))

  (DELDB (STEP RBR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP RBR-UPDATE-RECALL FIND NEXT))

  (DELDB (STATUS RBR-UPDATE-RECALL STARTING))
))

; Say the next word that needs to be recalled, if it is a covert item
(RBR-UPDATE-RECALL--PRODUCE--CONTINUE
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL PRODUCE RECALL)
  (NOT (STATUS RBR-UPDATE-RECALL STARTING))

  ; Assumes that content is available
  (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COVERT)
  (MOTOR VOCAL MODALITY FREE)
 ) THEN (
  (SEND-TO-MOTOR-F 'VOCAL ?STYLE 'CONTINUE (UNTRUNC ?CONTENT))

  (DELDB (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT COV-
ERT))

  (DELDB (STEP RBR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP RBR-UPDATE-RECALL FIND NEXT))
))

; Say the next word that needs to be recalled, if it is the add item
(RBR-UPDATE-RECALL--PRODUCE--CONTINUE-ADD-OBJECT
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL PRODUCE RECALL)
  (NOT (STATUS RBR-UPDATE-RECALL STARTING))

  ; Assumes that content is available
  (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTERNAL)
  (MOTOR VOCAL MODALITY FREE)
 ) THEN (
  (SEND-TO-MOTOR VOCAL ?STYLE CONTINUE ?CONTENT)

  (DELDB (TAG RBR-UPDATE-RECALL TO-BE-RECALLED-CONTENT IS ?STYLE ?MARKER ?CONTENT EXTER-
NAL))

  (DELDB (STEP RBR-UPDATE-RECALL PRODUCE RECALL))
  (ADDDB (STEP RBR-UPDATE-RECALL FIND NEXT))
))

; Find the next object in the recall chain
(RBR-UPDATE-RECALL--FIND-NEXT
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL FIND NEXT)

  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE COVERT MARKER ??? TYPE ???)
  (NOT (TAG RBR-RECALL ?NEXT-ID IS COVERT STIMULUS-END))

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS COVERT RECALLED))

  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

  (DELDB (STEP RBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Current object missing
(RBR-UPDATE-RECALL--FIND-NEXT--OBJECT-MISSING
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL FIND NEXT)

  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW)
  (NOT (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ??? SOURCE ??? MARKER ??? TYPE ???))

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS COVERT RECALLED))

  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (STEP RBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
))

; We just said the add object, and now are looking for the start
; of the sequence
(RBR-UPDATE-RECALL--FIND-NEXT--REHEARSED-CHAIN-START
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL FIND NEXT)

  ; We just said the add object...
  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG RBR-TRIAL ?ITEM-ID IS EXTERNAL NEW)

  (AUDITORY SPEECH-TAG ?UNUM ?NEXT-ID IS REHEARSED-CHAIN-START)

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS EXTERNAL NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS EXTERNAL RECALLED))

  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RBR-TRIAL ?NEXT-ID IS NEXT-TO-RECALL))

  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)

  (DELDB (STEP RBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
))

; Error handling: Start of chain missing
(RBR-UPDATE-RECALL--FIND-NEXT--REHEARSED-CHAIN-START--MISSING
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL FIND NEXT)

  ; We just said the add object...
  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG RBR-TRIAL ?ITEM-ID IS EXTERNAL NEW)

  (NOT (AUDITORY SPEECH-TAG ??? ??? IS REHEARSED-CHAIN-START))

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS EXTERNAL NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS EXTERNAL RECALLED))

  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))
  (ADDDB (TAG RBR-TRIAL GONE IS NEXT-TO-RECALL))

  (DELDB (STEP RBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RBR-UPDATE-RECALL CONTINUE RECALL))
))

; Stop when we reach the end of the chain
(RBR-UPDATE-RECALL--FIND-NEXT--COVERT-STIMULUS-END
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL FIND NEXT)

  (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL)
  (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW)
  (AUDITORY SPEECH ITEM-ID ?ITEM-ID NEXT ?NEXT-ID SOURCE COVERT MARKER ??? TYPE ???)
  (TAG RBR-RECALL ?NEXT-ID IS COVERT STIMULUS-END)

  (MOTOR VOCAL PROCESSOR FREE)
 ) THEN (
  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS COVERT NEW))
  (ADDDB (TAG RBR-TRIAL ?ITEM-ID IS COVERT RECALLED))

  (DELDB (TAG RBR-TRIAL ?ITEM-ID IS NEXT-TO-RECALL))

  (DELDB (STEP RBR-UPDATE-RECALL FIND NEXT))
  (ADDDB (STEP RBR-UPDATE-RECALL END RECALL))
))

; Cleanup after recall: Remove the STIMULUS-END tag
(RBR-UPDATE-RECALL--CLEANUP--REMOVE-STIMULUS-END
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL END RECALL)

  (TAG RBR-RECALL ?ITEM-ID IS COVERT STIMULUS-END)
 ) THEN (
  (DELDB (TAG RBR-RECALL ?ITEM-ID IS COVERT STIMULUS-END))
))

; Cleanup after recall: Remove the ADD-CHAIN-START tag
(RBR-UPDATE-RECALL--CLEANUP--REMOVE-ADD-CHAIN-START
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL END RECALL)

  (AUDITORY SPEECH-TAG ?UNUM ??? IS ADD-CHAIN-START)
 ) THEN (
  (REMOVE-AUDITORY-SPEECH-TAG ?UNUM)
))

; Wrap up the update recall
(RBR-UPDATE-RECALL--END
 IF (
  (GOAL DO RBR-UPDATE-RECALL)
  (STEP RBR-UPDATE-RECALL END RECALL)
 ) THEN (
  (DELDB (STEP RBR-UPDATE-RECALL END RECALL))
  (DELDB (GOAL DO RBR-UPDATE-RECALL))
))
```

REFERENCES

REFERENCES

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language, 38*(4), 341-380.

Awh, E., Jonides, J., Smith, E. E., Schumacher, E. H., Koeppe, R. A., & Katz, S. (1996). Dissociation of storage and rehearsal in verbal working memory: Evidence from positron emission tomography. *Psychological Science, 7*(1), 25-31.

Baddeley, A. D. (1986). *Working memory*. New York: Clarendon Press.

Baddeley, A. D. (1996). Exploring the central executive. *Quarterly Journal of Experimental Psychology: Human Experimental Psychology: Special Issue: Working Memory, 49A*(1), 5-28.

Baddeley, A. D. (2000). The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences, 4*(11), 417-423.

Baddeley, A. D. (2002). Is working memory still working? *European Psychologist, 7*(2), 85-97.

Brooks, J. O., & Watkins, M. J. (1990). Further Evidence of the Intricacy of Memory Span. *Journal of Experimental Psychology-Learning Memory and Cognition, 16*(6), 1134-1141.

Brown, G. D. A., Preece, T., & Hulme, C. (2000). Oscillator-based memory for serial order. *Psychological Review, 107*(1), 127-181.

Burgess, N., & Hitch, G. J. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review, 106*(3), 551-581.

Buschke, H., & Hinrichs, J. V. (1968). Controlled rehearsal and recall order in serial list retention. *Journal of Experimental Psychology, 78*(3), 502-509.

Callicott, J. H., Mattay, V. S., Bertolino, A., Finn, K., Coppola, R., Frank, J. A., et al. (1999). Physiological characteristics of capacity constraints in working memory as revealed by functional MRI. *Cerebral Cortex, 9*(1), 20-26.

Cavanagh, J. P. (1972). Relation between the immediate memory span and the memory search rate. *Psychological Review, Vol. 79*(6), 525-530.

Cohen, J. D., Perlstein, W. M., Braver, T. S., Nystrom, L. E., Noll, D. C., Jonides, J., et al. (1997). Temporal dynamics of brain activation during a working memory task. *Nature, 386*(6625), 604-608.

Conway, A. R. A., Kane, M. J., Bunting, M. F., Hambrick, D. Z., Wilhelm, O., & Engle, R. W. (2005). Working memory span tasks: A methodological review and user's guide. *Psychonomic Bulletin & Review, 12*(5), 769-786.

Cowan, N. (1992). Verbal memory span and the timing of spoken recall. *Journal of Memory and Language, 31*(5), 668-684.

Cowan, N. (1999). An Embedded-Processes Model of working memory. In A. Miyake & P. Shah (Eds.), *Models of working memory: Mechanisms of active maintenance and executive control.* (pp. 62-101): Cambridge University Press.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24*(1), 87-+.

De Beni, R., & Palladino, P. (2004). Decline in working memory updating through ageing: Intrusion error analyses. *Memory, 12*(1), 75-89.

Dell, G. S., & Repka, R. J. (1992). Errors in inner speech. In B. J. Baars (Ed.), *Experimental slips and human error: Exploring the architecture of volition.* (pp. 237-262): Plenum Press.

Drewnowski, A., & Murdock, B. B. (1980). The role of auditory features in memory span for words. *Journal of Experimental Psychology: Human Learning & Memory, 6*(3), 319-332.

Farrell, S., & Lewandowsky, S. (2002). An endongeous distributed model of ordering in serial recall. *Psychonomic Bulletin & Review, 9*(1), 59-79.

Farrell, S., & Lewandowsky, S. (2004). Modelling transposition latencies: Constraints for theories of serial order memory. *Journal of Memory and Language, 51*(1), 115-135.

Frank, M. J., Loughry, B., & O'Reilly, R. C. (2001). Interactions between frontal cortex and basal ganglia in working memory: A computational model. *Cognitive, Affective & Behavioral Neuroscience, 1*(2), 137-160.

Garavan, H. (1998). Serial attention within working memory. *Memory & Cognition, 26*(2), 263-276.

Gilmartin, K. J., Newell, A., & Simon, H. A. (1976). A program modeling short-term memory under strategy control. In C. N. Cofer (Ed.), *The Structure of Human Memory*. San Francisco: W. H. Freeman and Company.

Hartman, M., Dumas, J., & Nielsen, C. (2001). Age differences in updating working memory: Evidence from the Delayed-Matching-To-Sample Test. *Aging, Neuropsychology, & Cognition, 8*(1), 14-35.

Hazy, T. E., Frank, M. J., & O'Reilly, R. C. (2006). Banishing the homunculus: Making working memory work. *Neuroscience, 139*(1), 105-118.

Henson, R. N. A. (1998). Short-term memory for serial order: The start-end model. *Cognitive Psychology, 36*(2), 73-137.

Henson, R. N. A., Norris, D. G., Page, M. P. A., & Baddeley, A. D. (1996). Unchained memory: Error patterns rule out chaining models of immediate serial recall. *Quarterly Journal of Experimental Psychology: Human Experimental Psychology: Special Issue: Working Memory, 49A*(1), 80-115.

Hornof, A. J. (2004). Cognitive Strategies for the Visual Search of Hierarchical Computer Displays. *Human-Computer Interaction, 19*(3), 183-223.

Hulme, C., Newton, P., Cowan, N., Stuart, G., & Brown, G. (1999). Think before you speak: Pauses, memory search, and trace redintegration processes in verbal memory span. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 25*(2), 447-463.

Jones, D. M., Macken, W. J., & Nicholls, A. P. (2004). The Phonological Store of Working Memory: Is It Phonological and Is It a Store? *Journal of Experimental Psychology: Learning, Memory, and Cognition, 30*(3), 656-674.

Jonides, J., Lacey, S., & Nee, D. (2005). Processes of working memory in mind and brain. *Current Directions in Psychological Science, 14*(1), 2-5.

Jonides, J., Schumacher, E. H., Smith, E. E., Lauber, E. J., Awh, E., Minoshima, S., et al. (1997). Verbal working memory load affects regional brain activation as measured by PET. *Journal of Cognitive Neuroscience, 9*(4), 462-475.

Kessler, Y., & Meiran, N. (2006). All Updateable Objects in Working Memory Are Updated Whenever Any of Them Are Modified: Evidence From the Memory Updating Paradigm. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 32*(3), 570-585.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*(4), 391-438.

Kieras, D. E., & Meyer, D. E. (1998). *The EPIC architecture: Principles of operation*. Ann Arbor, MI: University of Michigan.

Kieras, D. E., Meyer, D. E., Mueller, S. T., & Seymour, T. L. (1998). *An EPIC computational model of verbal working memory.* Paper presented at the 39th Annual Meeting of the Psychonomic Society.

Kieras, D. E., Meyer, D. E., Mueller, S. T., & Seymour, T. L. (1999). Insights into working memory from the perspective of the EPIC architecture for modeling skilled perceptual-motor and cognitive human performance. In A. Miyake & P. Shah (Eds.), *Models of working memory: Mechanisms of active maintenance and executive control* (pp. 183-223). New York, NY, US: Cambridge University Press.

Kirchner, W. K. (1958). Age differences in short-term retention of rapidly changing information. *Journal of Experimental Psychology, 55*, 352-358.

Krieger, S., Lis, S., Cetin, T., Gallhofer, B., & Meyer-Lindenberg, A. (2005). Executive function and cognitive subprocesses in first-episode, drug-naive schizophrenia: An analysis of N-back performance. *American Journal of Psychiatry, 162*(6), 1206-1208.

Kucera, H., & Francis, W. N. (1967). *Computational analysis of present-day American English*. Providence, RI: Brown University Press.

Lewandowsky, S., & Murdock, B. B. (1989). Memory for serial order. *Psychological Review, 96*(1), 25-57.

Marshuetz, C. (2005). Order Information in Working Memory: An Integrative Review of Evidence From Brain and Behavior. *Psychological Bulletin, 131*(3), 323-339.

McElree, B. (2001). Working memory and focal attention. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 27*(3), 817-835.

McElree, B., & Dosher, B. A. (1993). Serial retrieval processes in the recovery of order information. *Journal of Experimental Psychology: General, 122*(3), 291-315.

McEvoy, L. K., Smith, M. E., & Gevins, A. (1998). Dynamic cortical networks of verbal and spatial working memory: Effects of memory load and task practice. *Cerebral Cortex, 8*(7), 563-574.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: I. Basic mechanisms. *Psychological Review, 104*(1), 3-65.

Miyake, A., Friedman, N. P., Emerson, M. J., Witzki, A. H., & Howerter, A. (2000). The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: A latent variable analysis. *Cognitive Psychology, 41*(1), 49-100.

Monsell, S. (1984). Components of working memory underlying verbal skills - A distributed capacities view - A tutorial review. In *Attention and Performance* (pp. 327-350).

Moore, M. E., & Ross, B. M. (1963). Context effects in running memory. *Psychological Reports, 12(2)*, 451-465.

Morris, N., & Jones, D. M. (1990). Memory updating in working memory: The role of the central executive. *British Journal of Psychology, 81*(2), 111-121.

Mueller, S. T. (2002). *The roles of cognitive architecture and recall strategies in performance of the immediate serial recall task.* Unpublished dissertation, University of Michigan, Ann Arbor, MI.

Mueller, S. T., Seymour, T. L., Kieras, D. E., & Meyer, D. E. (2003). Theoretical implications of articulatory duration, phonological similarity, and phonological complexity in verbal working memory. *Journal of Experimental Psychology-Learning Memory and Cognition, 29*(6), 1353-1380.

Nairne, J. S. (2002). Remembering over the short-term: The case against the standard model. *Annual Review of Psychology, 53*, 53-81.

Newell, A. (1973a). Production systems: Models of control structures. In W. G. Chase (Ed.), *Visual information processing.*: Academic.

Newell, A. (1973b). You can't play 20 questions with nature and win: projective comments on the papers of this symposium. In W. G. Chase (Ed.), *Visual Information Processing* (pp. 283-308). New York: Academic Press.

O'Reilly, R. C., & Frank, M. J. (2006). Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia. *Neural Computation, 18*(2), 283-328.

O'Reilly, R. C., & Soto, R. (2002). A model of the phonological loop: Generalization and binding. In T. G. Dietterich, S. Becker & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Page, M. P. A., & Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review, 105*(4), 761-781.

Palladino, P., Cornoldi, C., De Beni, R., & Pazzaglia, F. (2001). Working memory and updating processes in reading comprehension. *Memory & Cognition, 29*(2), 344-354.

Pollack, I., Johnson, L. B., & Knaff, P. R. (1959). Running Memory Span. *Journal of Experimental Psychology, 57*(3), 137-146.

Postle, B. R., Berger, J. S., Goldstein, J. H., Curtis, C. E., & D'Esposito, M. (2001). Behavioral and neurophysiological correlates of episodic coding, proactive interference, and list length effects in a running span verbal working memory task. *Cognitive, Affective & Behavioral Neuroscience, 1*(1), 10-21.

Roelofs, A. (2003). Goal-referenced selection of verbal action: Modeling attentional control in the Stroop task. *Psychological Review, 110*(1), 88-125.

Rohde, D. L. T. (2002). *A connectionist model of sentence comprehension and production.* Unpublished dissertation, Carnegie Mellon University, Pittsburgh, PA.

Ross, P., & Segalowitz, S. J. (2000). An EEG coherence test of the frontal dorsal versus ventral hypothesis in N-back working memory. *Brain and Cognition, 43*(1-3), 375-379.

Ruiz, M., Elosúa, M. R., & Lechuga, M. T. (2005). Old-fashioned responses in an updating memory task. *Quarterly Journal of Experimental Psychology A: Human Experimental Psychology, 58*(5), 887-908.

Schneider, W., Eschman, A., & Zuccolotto, A. (2002). *E-Prime User's Guide*. Pittsburgh: Psychology Software Tools Inc.

Schumacher, E. H., Lauber, E., Awh, E., Jonides, J., Smith, E. E., & Koeppe, R. A. (1996). PET evidence for an amodal verbal working memory system. *Neuroimage, 3*(2), 79-88.

Shimamura, A. P. (2000). Toward a cognitive neuroscience of metacognition. *Consciousness and Cognition, 9*(2), 313-323.

Smith, E. E., & Jonides, J. (1999). Storage and executive processes in the frontal lobes. *Science, 283*(5408), 1657-1661.

Sternberg, S. (1966). High-Speed Scanning in Human Memory. *Science, 153*(3736), 652-654.

Sternberg, S., Monsell, S., Knoll, R. L., & Wright, C. E. (1978). The latency and duration of rapid movement sequences: Comparisons of speech and typing. In G. E. Stelmach (Ed.), *Information processing in motor control and learning* (pp. 118-152). New York: Academic Press.

Thermenos, H., Goldstein, J., Buka, S., Poldrack, R., Koch, J., Tsuang, M., et al. (2005). The effect of working memory performance on functional MRI in schizophrenia. *Schizophrenia Research, 74*(2-3), 179-194.

Van der Linden, M., Collette, F., Salmon, E., Delfiore, G., Degueldre, C., Luxen, A., et al. (1999). The neural correlates of updating information in verbal working memory. *Memory, 7*(5-6), 549-560.

Verhaeghen, P., & Basak, C. (2005). Ageing and switching of the focus of attention in working memory: Results from a modified N-Back task. *Quarterly Journal of Experimental Psychology A: Human Experimental Psychology, 58*(1), 134-154.

Waugh, N. C., & Norman, D. A. (1965). Primary memory. *Psychological Review, 72(2)*, 89-104.

Wilson, M. (2001). The case for sensorimotor coding in working memory. *Psychonomic Bulletin & Review, 8*(1), 44-57.

Wilson, M. D. (1988). The MRC psycholinguistic database: Machine readable dictionary, Version 2. *Behavioural Research Methods, Instruments and Computers, 20*(1), 6-11.