On Modeling Intentions for Prospective Memory Performance

Renée Elio (ree@cs.ualberta.ca)
Department of Computing Science, University of Alberta
Edmonton, Alberta T6G 2E8

Abstract

Four models of intention handling were implemented and evaluated for their fit to a set of prospective memory reaction time data. The models instantiated either a top-down intention monitoring scheme or a bottom-up intention cueing scheme, corresponding to two types of descriptive accounts for prospective memory performance. Top-down models yielded reaction time patterns that more closely matched observed patterns. In these models, the cost of managing a delayed intention during a primary task is a function of the increased number of competing, possibly relevant intentions. Issues surrounding task-independent processing and representational commitments for what it means 'to intend' and to manage multiple intentions are discussed.

Keywords: intentions, goals, cognitive architecture, prospective memory, ACT-R, executive control

Introduction

This work is concerned with expanding the scope of cognitive theories to better address elements of the 'human executive control function' (Kieras et al., 2000; Schorr et al., 2003; Anderson et al., 2004; Salvucci, 2005). For example, the ACT-R modeling framework (Anderson & Lebiere, 1998; Anderson et al., 2004) has recently evolved to allow cognitive modelers to develop constructs like an 'intention module.' There are no theoretical commitments yet about any constraints on such a module and its interaction with other elements of the architecture.

Given its historical focus on single-task learning and memory performance, it is not surprising that ACT-R is silent about how the cognitive architecture might manage and allocate resources to multiple, unrelated goals. Such architectural commitments would need to come from experimental data that push the theory and architecture in some particular direction. For example, earlier versions of the architecture employed a "goal stack" which automatically created, maintained, and destroyed goal/sub-goal relationships. This was arguably a programming convenience in which there was little theoretical investment, but it had the consequence of indirectly giving an agent perfect memory for all her goals and subgoals. Experimental work (Altmann & Trafton, 2002) showing imperfect memory for subgoals forced the abandonment of the goal stack data structure, and its associated processes, from the modeling architecture. The current position is that goals are just like any other memory chunk, insofar as they need to be explicitly retrieved and are subject to forgetting.

To say that goals in the ACT-R 5.0 architecture are just like any other memory chunk (e.g., a declarative fact like 2+2 = 4) is not quite true. For example, there is special activation emanating from a *goal type* memory element, once it is retrieved into a special *goal buffer*. This is important for directed retrieval of task relevant information. Beyond that, however, the theory and architecture do not have further commitments about the representation of goal memory elements or how resources are allocated to competing goals.

There are, of course, several ACT-R models of dual-task performance, in which a person switches between alternative tasks in response to different stimuli (e.g., Sohn & Anderson, 2001). A typical ACT-R dual-task production model accomplishes this 'at the program level', i.e., through a task-specific, goal switching production set. Environmental cues for goal switching are referenced directly in the production rules; the representation assumptions about declarative knowledge for goal execution are also referenced in task-specific production rules. These are not disparaging observations. Indeed, in lieu of architectural commitments about how to manage and monitor competing, one *must* model multiple goal handling at the program level, i.e., with production rules.

Henceforth, I will use the term 'intention' rather 'goal', which has a particular meaning within frameworks like ACT-R. The work reported here focuses on computational accounts of what it means 'to intend' and what implications that might have for defining representational and processing constraints that distinguish 'intentions' and 'intentional action' from any other cognitive construct (e.g., from semantic or episodic memory traces) or process.

Dual-task performance, including complex skills like driving (see Salvucci, 2005), is relevant to addressing such questions. So too is the prospective memory literature. The term *prospective memory* is used to denote a type of memory task in which an individual has to remember to perform a particular action at some designated point in the future, either after a specified time lapse or when a critical event occurs. Interestingly, this literature describes this sort of task as remembering to execute a 'delayed intention.' Everyday examples involve intending to remember to buy milk on the way home or to relay a message to friends upon seeing them. The typical laboratory paradigm for studying prospective memory requires a person to perform an on-going (primary) task, with a further requirement to execute the prospective memory (secondary) task, when a particular cue appears. A feature that distinguishes prospective memory tasks from the typical dual-task paradigm is that the secondary (prospective memory) task must be executed relatively infrequently, unexpectedly, or sometimes not at all, relative to the primary task(s).

Descriptive accounts of prospective memory appeal to notions such as *delayed intention, intention monitoring*, *intention cueing,* or *preparatory attention* for executing intentions. My primary interest is giving some computational meat to these descriptive phrases, hopefully in a way that identifies possible architectural-level approaches to intention handling.

My approach is influenced by Bratman's (1990) analysis of the notion of intention, its semantics, and the role it plays in directing resources (see also Cohen & Levesque, 1990). Informally stated, a key notion is that to *intend (x)* is to have a persistent commitment to actions to bring about *x*, just as long as *x* is not satisfied, still deemed possible, and still deemed 'relevant' (the last feature prevents fanatical pursuit of intentions that, by some reasoning process, ought to be dropped). A corresponding computational perspective would view 'having an intention' as having an execution thread for some process, which has associated conditions for its suspension, resumption, or termination. There can be multiple (but, as per Bratman's account, not semantically conflicting) intentions and some executive must manage how they consume and direct processing resources, and recognize intention conflicts.

In this regard, 'having an intention' does *not* map to 'having a goal memory chunk.' It instead entails all the representational and processing constraints that surround the formation and manipulation of such structures, including the allocation of resources. Within a framework like ACT-R, we can regard a typical cognitive model of some task performance as a single execution thread. This execution thread directs and consumes resources (e.g., visual attention, memory retrieval) in ways that yield reaction time measures, as dictated by the theory's assumptions concerning the time to execute primitive processing steps.

An important question, therefore, is how the cognitive architecture handles multiple execution threads, in a task-independent manner. This can first be investigated at the program level, by developing general intention handling productions that operate over general declarative structures for representing information about intentional action. The simulation of particular performance data can then be considered within this generic framework. It would be most informative if there were qualitatively different *control* accounts for some particular task performance, because intention handling is a matter of an executive control scheme interacting with some representation of intention information.

It so happens that two qualitatively different control accounts have been proposed for prospective memory performance. Under the automatic-retrieval account, an intention *i* is represented in memory with its cue and associated action. However, as long as the cue for executing intention *i* is not present in the environment, there is no impact, as it were, of 'having' intention *i* on the execution of some other, ongoing task *t*. It is only when intention *i'*s cue appears that there is capacity overhead for its automatic retrieval and the retrieval of its action. An alternative view is that the successful execution of a prospective memory task *always* requires ongoing capacity that impacts the execution of the primary, on-going task *t* (see Smith, 2003; Marsh & Hicks, 2003). McDaniel and Einstein (2000) offer a hybrid account and the reader is directed to these citations for experimental results that support both theoretical perspectives.

Prospective memory performance and these sorts of accounts offer another vehicle for exploring intention handling. Here, I focus on a single set of experimental results as a first step in this matter. The aim is not to develop an account of this single set of results—that would be limited in value. Rather, the aim is to explore a space of alternative intention-handling control models. The *relative* success of the alternative models in simulating observed prospective memory data is useful for understanding the computational assumptions that characterize descriptive accounts like the ones above. The best of these alternative models can then be investigated for its extendibility to other results. The overall exercise is useful for understanding what elements of task-independent intention handling are better viewed as features of the cognitive architecture, rather than as features of task-specific programs. The next section presents the experimental results that are the focus for this effort.

Capacity Impacts for Delayed Intentions

Smith (2003, exp. 1) instructed subjects how to perform a lexical decision task in the usual manner, by giving one response to signal a non-word, and a different response to signal a word. She also had subjects memorize 12 *target* words, and instructed them to make a third response, if one of these target words appeared. Remembering to make this distinct response to target words constituted the prospective memory task. Within every block of 38 trials, trials 1-18 presented a *non-word* string; trial 19 presented a *non-target* word (a.k.a. a control word), trials 20-37 presented non-words again, and trial 38 presented a *target* word. Each of 12 blocks had this overall structure. Note that the prospective memory task (the distinct response to each of 12 target words) occurred relatively infrequently: once every 38 trials. One group of subjects (the *embedded intention* group) understood their task as described: the appearance of a target word during the lexical decision task should elicit the 'is a target' response instead of the 'is a word' response. A second group of subjects (the *delayed intention* group) also memorized the same set of target words and the third, distinctive action to make whenever these words appeared on the screen. However, they were told that the task of responding in a special way to target words would come sometime *after* the lexical decision task. By design, Smith never asked the delayed-intention subjects to do this task—that was just the cover story for memorizing the target words and the special response at the start of the experiment. However, the target words appeared in stream of lexical decision stimuli for the delayed-condition subjects, just as they did for the embedded-condition subjects.

Smith's rationale for this manipulation was as follows. If the intention to respond "specially" to target words during the lexical decision task requires capacity *only* when that intention is formed (e.g., as a product of understanding the experimental instructions), and then again *only* when a cue for its execution appears (when the target word shows up every 38th trial), then there should be no reaction time differences on *control* trials between embedded intention and delayed intention conditions. After all, any intention about *targets* is not being cued on trials presenting *non-targets.* And so the embedded condition subjects should not have any overhead for 'having' this intention during the lexical decision task, relative to the delayed intention subjects who do not 'have it'— in some sense—until after the lexical decision task is over. However, Smith reports that embedded intention subjects took approximately 300 msec longer than delayed intention subjects on *control* trials; they also took 150 msec longer than delayed subjects on *non-word* trials. She argues that some conscious strategic capacity is allocated to 'remembering to remember' a prospective memory task, even in the absence of cues for its execution.

Simulation Models

Memory Structures

The class of models explored for this task was implemented within the ACT-R modeling framework, which embodies certain theoretical principles that yield reaction time predictions. The theory assumes that each elemental, serial cognitive step requires 50 msec. These are encoded as productions. Each production can be viewed as a primitive 'fetch', 'store', 'compare' or 'modify' operation, to use an assembly language analogy. 'Fetch' operations are pattern based retrieval requests to a declarative memory, which brings some pattern (chunk) into a buffer for subsequent processing. The latencies associated with these operations, and probability that any particular memory chunk is returned, are determined by the theory's spreading activation equations.

The four models developed here were task-independent control schemes for handling intentions. They all operated on identical declarative memory representations, shown in Table 1. In ACT-R's declarative memory, each memory chunk is an instance of some class (specified with the *isa* relation), which defines its possible feature slots. The intention chunk had these features (and possible values): *object_type* (string), status (unsatisfied | satisfied), *lexicalstatus* (word | notword), *targetstatus (target |nottarget)* and *world (now |later)*. With the exception of the *status* feature, all the other feature-value pairs constitute satisfaction conditions for the intention. The action associated with an intention is represented in a separate *action* memory chunk (e.g., the *signalnonword* memory chunk in Table 1).

The declarative memory designed for both delayed and embedded conditions used exactly three intention structures. The *nonword-intention* was identical in both declarative memories. However, the two other intention chunks for responding to words and to targets were necessarily different in these two memories. In the embedded model, the *word-intention* had two conditions: that the string was a word but not a target (*lexstatus=word* and *targetstatus=not-target*). In the delayed condition model, the *word-intention* required only that the string was a word; it did not include any specification for *targetstatus*. For both the embedded and delayed models, the *target-intention* chunk had conditions that the string be both a word and a target. What distinguished this intention under the two experimental manipulations was a third condition, the *world* feature. This represented the experimental manipulation that embedded subjects knew to respond to targets during the lexical decision task (*world=now*), while delayed intention subjects knew to respond to targets after the lexical decision task was over (*world=later*).

The representation of a single lexical decision trial was held in the *onetrial* chunk (Table 1, bottom). Informally put, each single trial represents a 'world' in which intentions might be relevant and satisfied. The *onetrial* chunk became elaborated with an encoding of the presented string and other features, under the control of the intention-handling productions (described in the next sections). The declarative memories for both the delayed and embedded conditions had identical memory chunks for words and targets.

A few remarks about this representational scheme are in order. First, an intention's conditions are specified within a single memory element; that is a matter of programming convenience that does not bear on the results presented here. Distributing these conditions across distinct memory chunks opens up avenues of empirically testable predictions, which I mention later. Second, the control structures described below implicitly use *object_type=string* as a contextual relevance condition. The assumption is that all intentions are 'about' something, and the intentions for this task were 'about strings in the experiment.*'* In principle, declarative memory could contain many other intention structures related to unsatisfied intentions 'about' anything (buying milk, informing friends); such structures would not make it past the contextual relevance check the control schemes use. Third, an intention's associated action is held in a separate memory chunk. This allows for intended *actions* to be forgotten, even if the intention *to* act is remembered.

It is fair to use the same *number* of intentions in the declarative memories for both the embedded and delayed conditions: Smith's delayed condition subjects could recite both the action to take when a target word appeared, as well as all the target words themselves, at the end of the lexical decision task. However, the details of these three intention structures across the two experimental manipulations *must* be different, given the aim to evaluate general intention-handling control schemes. Put another way,

Table 1: Declarative Memory Structures under Embedded vs. Delayed Instructional Conditions

	(nonword-intention *isa* intention *name* respondnonword *object_type* string *status* unsatisfied *lexstatus* notword *world* now)
<u>*embedded*</u>:	(word-intention *isa* intention *name* respondword *object_type* string *status* unsatisfied *lexstatus* word *targetstatus* nottarget *world* now)
<u>*delayed*</u>:	(word-intention *isa* intention *name* respondword *object_type* string *status* unsatisfied *lexstatus* word *world* now)
<u>*embedded*</u>:	(target-intention *isa* intention *name* respondtarget *status* unsatisfied *object_type* string *lexstatus* word *targetstatus* target *world* now)
<u>*delayed:*</u>	(target-intention *isa* intention *name* respondtarget *status* unsatisfied *object_type* string *lexstatus* word *targetstatus* target *world* later)

(signalnonword *isa* action keypress "k" *fo*r nonword-intention) (signalword *isa* action keypress "d" *for* word-intention) (signaltarget *isa* action keypress "m" *for* target-intention)
(onetrial *isa* world *object_type* string o*bject* "dog" lexstatus unknown *targetstatus* unknown *world* now)

Table 2: Intention Handling Control Algorithms

```
(a) top down intention monitoring
  1. Detect and encode string s and put it in trial t's representation
  2. Repeat
       2.1 Retrieve an unsatisfied intention i about strings (object_type= string),
             spreading activation from trial t's representation
       2.2 Repeat
                 Select an condition of intention i that is unknown in trial t's representation
                       Elaborate t's representation with a value for that condition
            Until      t's representation has a specification for all of i's conditions
       2.3 If any condition of i is not matched by the t's elaborated representation,
             mark the intention i unsatisfied; else mark it satisfied
         Until there is a satisfied intention about strings
  3.  Retrieve the action associated with the satisfied intention
  4.  If no action is retrieved, then respond randomly; else execute the retrieved action

(b) bottom up intention cuing
  1. Detect and encode string s and put it in trial t's representation
  2. Elaborate t's representation by determining either the lexical status or target status of string s
  3. Repeat
            3. 1 Retrieve an unsatisfied intention i about strings (object_type =string),
                   spreading activation from trial t's representation
            3.2 While t's representation is still missing a value for one of i's conditions
                        Select an condition of intention i that is unknown in trial t's representation
                        Elaborate the representation of trial t with a value for that condition
            3.3  If any condition of i is not matched by the trial's elaborated representation,
                   mark the intention i unsatisfied; else mark it satisfied
          Until there is a satisfied intention about strings
   4.  Retrieve action associated with satisfied intention
   5.  If no action is retrieved, then respond randomly, else execute the retrieved action
```

Smith's instructional manipulations about intended actions are reflected as differences in declarative knowledge about intentional actions, processed by general control schemes.

Intention Handing Control Schemes

Table 2 presents pseudo-code for a top-down intention handling control scheme and for a bottom-up intention cueing control scheme. A key difference between top-down monitoring and bottom-up cueing concerns what is in the subject's representation of the world (the current trial) when the model makes a retrieval request for an unsatisfied intention. In top-down intention monitoring (Table 2(a)), the three relevant intentions are equally likely to be retrieved on step 2.1, at least on the first retrieval request: all intentions about *strings* are unsatisfied at the start of a new trial. The retrieved intention's feature slots cause the model to determine those features' values for the current trial.

Suppose the trial presents the string *"table",* but the model retrieves the *nonword-intention* structure on step 2.1. This intention requires *lexstatus=nonword,* so the model allocates processes to determine the lexical status of the string. As a consequence of these processes, the representation of the current trial is changed to be *lexstatus=word*. The model continues identifying features of the world for all the conditions specified in the just-retrieved intention structure. It then checks (step 2.3) to see if the conditions of the just-retrieved intention structure match the world state. If it does, the intention is satisfied and there can be resources allocated to retrieving its action; otherwise, the intention structure is marked as unsatisfied.

The latter case would cause another retrieval request to fetch an unsatisfied intention about strings. However, using our current example, the world representation has already been elaborated with the feature *lexstatus=word*. This increases the probability, via spreading activation, of retrieving intentions that include *word* as a condition value. So even in this top-down perspective, there is an influence of bottom-up cueing on subsequent retrieval requests for unsatisfied intentions. A pure top-down approach would reset the trial representation after each unsatisfied intention, and re-elaborate the world representation all over again.

In the bottom-up cueing algorithm, the world is elaborated first (some feature is selected to be determined, in step 2 of bottom-up cueing) and then unsatisfied intentions are retrieved. Spreading activation will favor intention information that includes the just-determined feature as a required world condition. However, just as the top-down version has a bit of bottom-up cueing, the bottom up version has a top-down elaboration flavor on step 3.1. The just-retrieved intention structure may require additional world tests, and so the algorithm will commence elaborating the world in a manner specified by the intention structure, to determine if it is satisfied.

The bold-faced steps in both control schemes implement a kind of full-elaboration policy: consider *all* the required conditions in an intention structure and elaborate the world accordingly, *before* deciding whether the intention is mismatched or not. A so-called partial-mismatch version of each model was also tested. This allowed an intention to be marked as unsatisfied as soon as a single mismatch was detected.

Implementation in the ACT-R framework

The Table 2 algorithms were implemented as production systems, where each primitive action (an encoding, a retrieval request, a comparison between memory chunks held in buffers) maps to a single production. Paralleling the Smith experimental design, all models included an encoding of 24 words, half of which were also encoded as target words. A model received 10 non-word trials, a control word trial, 10 non-word trials, and then a target word trial, until all the 24 words had appeared. The parameters and activation levels for all declarative memory chunks were the same across both conditions and all four models. Each model was run 10 times and the response latencies were averaged over these 10 runs. Accuracy in these models was perfect; it is straightforward to adjust certain parameters to model the high (but imperfect) observed accuracy.

Results and Discussion

Table 3 shows that the top-down/partial mismatch model was the best at simulating the size of the observed reaction time differences between the embedded and delayed conditions on control trials (observed: 335 msec; simulated: 277) and on non-word trials (observed: 154 msec; simulated 143 msec). The top-down models also do a better job of simulating the within-conditioin reaction time patterns, namely (a) the longer reaction times on control trials than on non-word trials in the embedded condition, and (b) the slightly longer latencies for non-word trials than for control trials in the delayed condition. The partial-mismatch/top down model is the best of these four control schemes. This is due to shorter reaction times on the nonword trials and this makes perfect sense: the model can reject any intention it retrieves about words, as soon as it determines that the current trial is presenting a non-word.

Table 3: Simulated and Observed Reaction times (msec)

observed (Smith, 2003, exp. 1)				
embedded	nonword	936	control	1061
delayed	nonword	782	control	726
top-down full elaboration				
embedded	nonword	1205	control	1237
delayed	nonword	985	control	941
top-down partial mismatch				
embedded	nonword	1124	control	1225
delayed	nonword	981	control	948
bottom-up full elaboration				
embedded	nonword	1218	control	1074
delayed	nonword	1118	control	1101
bottom-up partial mismatch				
embedded	nonword	1106	control	1113
delayed	nonword	1042	control	1021

One reason that top-down models perform better than bottom-up models is related to the representation of the *onetrial* chunk for the current trial, which includes the feature *world=now* for each trial. For the delayed condition, there is a lower probability that the representation of *target-intention,* marked with *world=later,* will be retrieved: spreading activation from *world=now* in the trial's elaboration favors retrieval of *word-intention* and *nonword-intention*. For bottom-up models, step 2 elaborates the world representation with either the lexical feature or the target feature (the *world=now* feature is already known for the trial). These elaborated features of the trial will favor just those intentions that have one of these features, plus the feature *world=now*. This is more likely to retrieve 'just the right' intention for the embedded condition, since, by definition, the world is cueing 'just the right intention.' This serves to reduce the set size of competing intentions that is retrieved as potentially relevant, removing the embedded vs. delayed condition effect.

The top-down models thus instantiate Smith's preparatory attention account as a kind of interference mechanism: there is only a small chance that the model using the delayed intention representation will accidentally (and incorrectly) 'think of' the intention about targets during the lexical decision task; for the embedded case, this chance is much higher.

A final remark about the bottom-up algorithm is important. This algorithm seems to 'know' that it should elaborate the world with information about whether a string is a word or a target (see bottom-up algorithm, step 2). But of course, it cannot do that that unless it also 'knows' that such features are crucial to current, unsatisfied intentions. And this in turn seems to suggest that we cannot get possibly get away from some element of top-down intention monitoring at the executive level. This seems right, and consistent, with Bratman's analysis: the functional role that intentions play for resource-bounded agents is to direct which of many features in a changing world are important to encode and monitor.

Summary

There are three contributions from this exercise. First, from four models tested, one simulated the relative reaction time patterns observed between and within Smith's (2003, exp. 1) experimental conditions. This model instantiates Smith's preparatory attention account as a kind of interference effect among competing intentions. Hence, it would predict a greater capacity overhead as a function of the size of the competing intention set and the overlap of conditions associated with those competing intentions. Second, this best fitting model can be used as a starting point for constraints on intention handling at the architecture level. To implement the Table 2 algorithms with the necessary activations from retrieved intention structures, an additional buffer was used in ACT-R 5.0. However, there is latency overhead resulting from production rules that shift these intention structures among these buffers. It may be possible to define some of these operations as coming 'for free' in the architecture (possible in ACT-R 6.0). The *absolute* simulated reaction times would thereby decrease and be more in line with the observed reactions times. Third, the declarative representation of intention information here allows for the forgetting of unsatisfied intentions (they must have a minimum activation level to be retrieved), for remembering an intention but forgetting the associated action, and for the role of cue salience and similarity in retrieval effects (not in play for this particular modeling effort, but admitted by the representational commitments).

These ACT-R models predict that the size of the non-word block in the stimulus stream determines the size of the reaction-time difference between delayed and embedded conditions on *non-word* trials (this emerges from how highly activated this intention becomes from repeated access). A second testable prediction concerns the psychological reality of a contextual relevance condition, which functions to define the upper bound on the set of competing, and hence interfering, intentions. Another avenue for experimental investigation concerns predictions emerging from a distributed representation of intention conditions across separate memory chunks. This leads to wondering whether declarative information about intentional action is governed by retrieval and decay functions that are different from those functions governing semantic memory. These issues all speak to how to embody intention-handling assumptions at the architectural level. Current work concerns applying the general intention handling schemes on a wider set of experimental results.

Acknowledgements

This work was supported by an NSERC discovery grant to R. Elio.

References

Altmann, E.M., & Trafton, J.G. (2002). Memory for goals: An activation-based model. *Cognitive Science, 26,*39–83.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*, 1036-1060

Anderson, J. R. & Lebiere, C. (1998). *The atomic components of thought. Mahwah*, NJ: Erlbaum.

Bratman, M.E. (1990). What is intention? In P.R Cohen, J. Morgan,& M.A. Pollack (Eds.), *Intentions in Communication*. Cambridge, MA: MIT Press.

Cohen, P. R. & Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence, 42*, 213-261.

Kieras, D. E., Meyer, D. E., Ballas, J.A., and Lauber, E. J. (2000). Modern Computational Perspectives on Executive Mental Processes and Cognitive Control: Where to from here?. In S. Monsell & J. Driver (Eds.), *Control of Cognitive Processes: Attention and Performance* XVIII. Cambridge, MA: MIT Press.

McDaniel, M.A. & Einstein, G. O. (2000). Strategic and Automatic Processes in Prospective Memory Retrieval: A Multiprocess Framework. *Applied Cognitive Psychology*, *14*, 127-144.

Marsh, R. L. & Hicks, J. L. (2003) Interference to Ongoing Activities Covaries With the Characteristics of an Event-Based Intention. *JEP: Learning, Memory, and Cognition*, *29*, 861–870.

Salvucci, D. D. (2005). A multitasking general executive for compound continuous tasks. *Cognitive Science*, *29*, 457-492.

Schorr, T., Gerjets, P. & Scheiter, K. (2003). Analyzing Effects of Goal Competition and Task Difficulty in Multiple-Task Performance: Volitional Action Control within ACT-R (pp 1053-1058). *Proceedings of the 25th Annual Meeting of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Smith, R. E. (2003). The Cost of Remembering to Remember in Event-Based Prospective Memory: Investigating the Capacity Demands of Delayed Intention Performance. *JEP: Learning, Memory, and Cognition, 29,* 347–361.

Sohn, M-H., & Anderson, J. R. (2001). Task preparation and task repetition: Two-component model of task switching. *Journal of Experimental Psychology: General,* 130, 764–778.