Toward Unifying Cognitive Architecture and Neural Task Set Theories

Bryan Stearns (stearns@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

John E. Laird (laird@umich.edu)
University of Michigan, 2260 Hayward Street
Ann Arbor, MI 48109-2121 USA

Abstract

PRIMs theory describes a computational foundation for understanding task-general human learning and transfer using rule-based cognitive architectures. Integration with ACT-R has yielded Actransfer, a model that replicates human learning and transfer across many tasks. However, this model requires task-specific latency scaling parameters from ACT-R to model different tasks, implying that there is missing computation in the theory. Neuroscience literature has separately defined the "task set" as the neural encoding that configures stimulus-response rule behavior in working memory. The process of switching between different task sets is often used to explain human latency costs. This paper introduces an alternate instantiation of PRIMs theory that enacts task set processing to account for the missing computation via a novel memory structure called a procedure context. Human tasks of varying complexity are modeled across two experiments. Procedure contexts model human latencies and interference effects in all tasks by integrating latency, decision making, task representation, and learning as aspects of a single unified process. This approach offers promise for future modeling within cognitive science by uniting theories from neuroscience and cognitive architectures.

Keywords: task set; cognitive modeling; cognitive architecture; Soar; PRIMs; PROPs; task switching; learning

Introduction

PRIMs theory (Taatgen, 2013) has been integrated with the ACT-R cognitive architecture (Ritter, Tehranchi, & Oury, 2019) for a task-general model of learning called Actransfer, which results in transfer profiles similar to those observed in humans. PRIMs theory is based on composing computationally-primitive condition-action memory operations, or "PRIMs," through practice, and is associated with basal ganglia modeling of rule-driven behavior (Taatgen, 2019). Actransfer is an architecture, with computational definitions for decision making, working memory (WM), and long-term memory that come from ACT-R, including a function for simulating latency from long-term declarative memory retrievals (Brasoveanu, 2015):

$$T_{retrieve} = F_r \times e^{-A} \tag{1}$$

F_r is the *latency factor* parameter that varies from model to model, and A is the activation of the retrieved memory.

Varying F_r for different task models is common in the ACT-R tradition, but this practice implies that some theory for the computational differences of different tasks is missing. Actransfer results rely on task-specific F_r values, varied sometimes by an order of magnitude, to substantiate model simulations, and this prevents it from being a truly task-general model of human learning and cognition.

One place to look for understanding task-specific processing differences is task set theory (Sakai, 2008). Task set theory describes how neural activity establishes links between perceived stimuli and task responses. A task set is often framed in terms of rule-driven behavior, and switching task sets for different task rule behaviors is thought to significantly contribute to task-specific latency.

This paper explores the unification of these lines of research in order to achieve a task-general model of learning that avoids task-specific F_r latency values. In this approach, PRIMs theory is implemented in the Soar cognitive architecture (Laird, 2012), resulting in a model called PROP$_3$. PROP$_3$ uses structures called "procedure contexts" that are theorized to correspond to task sets, and that provide a task-general approach to timing. Two modeling experiments are presented that use both PROP$_3$ and Actransfer to model the same tasks using identical task definitions and reasoning, and results from both implementations are compared. PROP$_3$ models are timed by treating procedure context switches as task set switches with uniform latency costs, while Actransfer models use Equation 1 and task-specific values of F_r.

Task Sets

A task set is a WM representation of a *set* of contextually associated stimulus-response rules, which make behaviors available to decision making (Oberauer, 2010). Active task sets are implied in the neural activity that is sustained while a subject comprehends instructed task behavior, waits to perceive task stimuli, recognizes them, and responds. Studies indicate a hierarchical organization of task sets across the pre-frontal cortex and to some degree the anterior cingulate cortex, distributed according to the abstractness of the represented procedures (Sakai, 2008).

Task set neural activity changes when switching tasks, or when performing different operations within a single task. Specific operations, such as comparing whether two objects are the same, have been linked with specific neurons (Sakai, 2008). The time required to establish a task set is a dominant component of task switch costs and of WM interference. By configuring stimulus-response mappings for current goals, task sets filter the scope of perceived stimuli and available re-

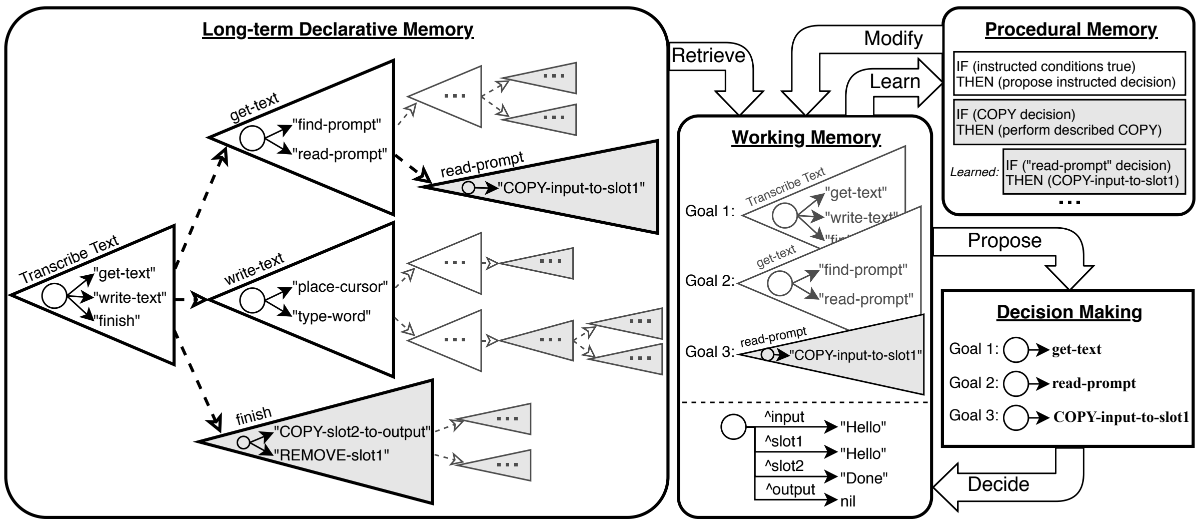


Figure 1: The PROP$_3$ procedure context model in Soar. Conditions are shown in white, actions in gray.

sponses, thereby shielding decision making from irrelevant information (Dreisbach & Haider, 2008). The cost of switching to a new task is considered the consequence of successful shielding during the prior task.

This paper proposes to represent a task set in Soar as *declarative* knowledge for describing task rules, or "D-Rules" here for clarity. These control the selection and firing of *procedural* production rules, or "P-Rules." Note that the declarative/procedural distinction in cognitive architectures is substantially based on computational aspects of the representations (such as how they are accessed and modified) in addition to psychological theory. In ACT-R or Soar, the distinction between procedural and declarative is effectively the distinction between P-Rules and what is manipulated by P-Rules. Since WM defines what knowledge is available for P-Rules to use, all WM content is called declarative in Soar.

PROP$_3$ and Procedure Contexts

The Primitive Operator ("PROP") model (Stearns, Assanie, & Laird, 2017) is a model of human learning and transfer that follows PRIMs theory. It is a set of P-Rules encoded in Soar that allow it to learn any task, once it is given a declarative representation of task D-rules. D-Rules are encoded in procedure contexts via a task-general "assembly language" of primitive conditions and actions, which are interpreted as instructions for task execution by condition and action P-Rules. Learning and transfer behavior arises during practice as D-rules are converted to more efficient P-rules. PROP$_3$ is a revision of previous models and introduces procedure contexts as a model of task set theory.

A *procedure context* is a symbolic representation of a *set* of contextually associated D-Rules, which makes behaviors available to decision making when in WM. Procedure contexts are hierarchically organized in Soar's long-term declarative memory, structured according to the abstractness of the procedures, following the Soar Problem Space Computational Model (PSCM) (Laird, 2012). Retrieving a context into WM corresponds to activating a task set.

Figure 1 shows example procedure contexts for a "Transcribe Text" task, where a subject must copy a prompted line of text into a computer text editor. Procedure contexts, shown as triangles, encode a subject's understanding of task procedures in long-term declarative memory. Each procedure context describes the decision options relevant to the subject for different task states. Each decision can lead to another context and set of possible decisions and procedures. Here, the subject must first read the text from the prompt before writing each word. Reading the prompt requires finding it and reading the text. Writing each word might require placing the edit cursor at the right location before typing the known word.

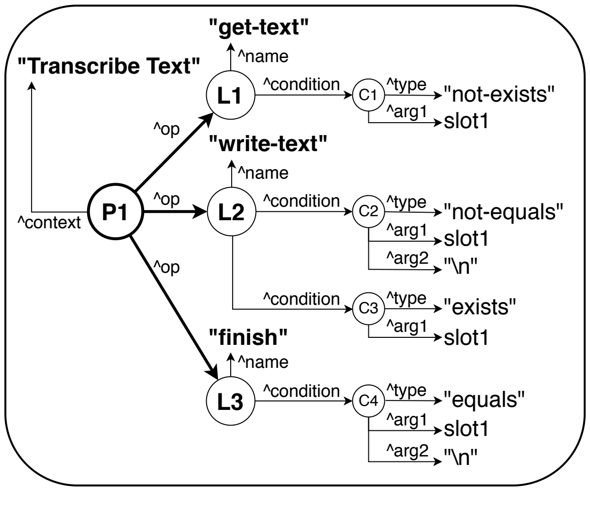


Figure 2: A procedure context for the "Transcribe Text" goal.

Figure 2 depicts the starting "Transcribe Text" procedure context in Soar's graph-based WM. It describes three D-

Rules in terms of conditions for pursuing deeper contexts. The "write-text" structure describes the D-Rule, `IF (slot1 is not equal to "\n" AND slot1 exists) THEN (propose "write-text" as an operation).` D-Rule conditions in a procedure context are evaluated once the context is retrieved into WM. Soar elaboration P-Rules evaluate condition structures in parallel, so that any operation is proposed if and only if its conditions are met. This is consistent with implications of recent basal ganglia neural modeling of PRIMs theory (Taatgen, 2019).

Once operations are proposed for all D-Rules with matching conditions, Soar decision making selects one to pursue as a subgoal. Soar maintains a stack of active goals in WM, as shown in Figure 1. As in the figure, if "Transcribe Text" was first retrieved into WM, Soar could select the "get-text" operation if its conditions were satisfied, which would result in retrieving the "get-text" procedure context as a subgoal for more detailed decision making. Similarly, if the "read-prompt" operation was then selected, the corresponding context would be retrieved as a deeper subgoal. All context hierarchies terminate in action operations, as shown in gray in the figure. Procedure contexts describe actions with structures comparable to the condition structures in Figure 2. Once an action D-Rule is retrieved, P-Rules execute the memory operation. Once actions are complete, or if the task state changes so that a different action is preferred, the irrelevant portion of the WM stack is removed and new goals are pursued.

Soar automatically learns new P-Rules that summarize subgoal actions, through a process called chunking. Learned P-rules can execute actions without the need for a subgoal. In Figure 1, a new P-Rule has just been learned from executing the "read-prompt" operation, so that the "read-prompt" procedure context does not need to be retrieved in future. PROP$_3$ uses a gradual version of Soar's chunking mechanism (Stearns & Laird, 2018).

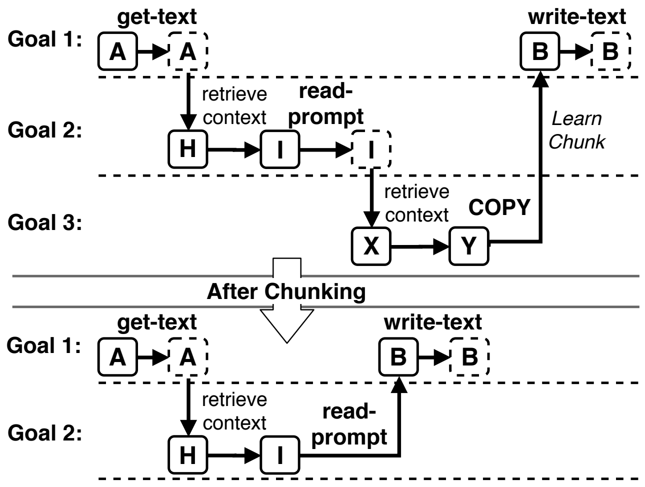


Figure 3: Soar chunking.

Figure 3 depicts the Soar chunking process. Boxed letters represent a WM state at different points in time, and arrows represent operations that modify WM. Dashed boxes represent where the selected change to WM requires subgoal decision making. The figure shows what the decision making in Figure 1 would look like in the future after learning.

When D-Rules entail multiple actions together (such as the "finish" context in Figure 1), these are composed in a hierarchy similar to conditions, in pairs, according to PRIMs theory for composing transferable P-Rules (Taatgen, 2013). For each iteration of practice, only the lowest subgoal is chunked, but eventually all actions will be chunked together. All executed D-Rule actions (those in gray in Figure 1) are gradually chunked into single P-Rules in this way, so that they can then fire in place of deliberate subgoals, but Soar does not learn chunks that replace condition-based procedure contexts such as those shown in white in Figure 1. These describe decision choices, not parallel actions for applying one decision.

In modeling human behavior with PROP$_3$, each decision corresponds to 50 msec. In modeling task sets with procedure contexts, time is required to add contexts to WM. Creating a subgoal requires one cycle, while retrieving knowledge from long-term declarative memory requires two. This leads to the following function for the latency of retrieving contexts:

$$T_{context} = F_g \times G + F_c \times C \tag{2}$$

where G is the number of times a subgoal is created, F_g is a *goal latency factor* (50 msec for one cycle), C is the number of times procedure contexts are retrieved, and F_c is a *context latency factor* parameter (100 msec for two retrieval cycles). C and G increase or decrease according to the number of subgoal branches in a task's PSCM structure.

Chunks speed up task performance by reducing G and C for task actions. But more procedure contexts for a task's decision making increases G and C. Thus, this model predicts greater latency for tasks that require more complex decisions.

In summary, PROP$_3$ models the latency of task set switching as the time required for switching procedure contexts, which depends on both the hierarchical structure of task decision making and on learning task actions.

Comparison with Actransfer

As a PRIMs model, Actransfer learns tasks by converting D-Rules to P-Rules, but there are several differences in comparison to PROP$_3$. First, Actransfer D-Rules are self-contained bundles of conditions and memory actions without the PROP$_3$ hierarchy. Only one D-Rule is in WM at a time, whereas PROP$_3$ allows one D-Rule hierarchy branch at a time. Actransfer retrieves the D-Rule with the highest activation.

Additionally, Actransfer models latency for retrieving D-Rules using Equation 1, and its F_r can vary significantly between tasks, whereas F_g and F_c in Equation 2 are the same across tasks. Actransfer matches PROP$_3$ in using 50 msec per decision, but it will use more decisions per D-Rule than PROP$_3$, because it evaluates D-Rule conditions deliberately via decision making rather than in the background in parallel.

Another difference is that Actransfer shields decision making from irrelevant procedures by conditioning rule logic on explicit *goal names* in WM, as is standard in ACT-R, Soar, and other architectures. PROP_3 shields decision making via the context-specific set of available D-Rules, without the need for explicit goal names. Further, Actransfer goal names are like any other WM value, and can change in parallel with other WM operations without adding latency as in Equation 2.

The most significant difference is that, where Equation 1 *adds* latency that is not otherwise part of the model, Equation 2 merely describes the computational cost of using procedure contexts in Soar.

The following experiments compare PROP_3 with Actransfer to test the implications of these differences. To avoid tailoring tasks or procedure context structures for PROP_3 results, the experiment tasks are chosen from those previously modeled by Taatgen (2013) using Actransfer, and the D-Rules of those models are imported into PROP_3 as procedure contexts. In order to define the procedure context hierarchy structure, all Actransfer D-Rules that condition on the same goal name are grouped into a single procedure context, and each change in goal name is replaced with a procedure context switch. Otherwise, the D-Rules are unchanged.

Experiment 1: Within-task switching

The first experiment uses two human tasks of different complexity to compare the PROP_3 and Actransfer timing models for within-task goal switch costs. The complex task is the text editors task from (Singley & Anderson, 1985), which required a deep hierarchy of task actions such as "edit-document," "edit-line," "find-cursor," and so on. In this task, humans took 20 to 120+ seconds to complete each assigned operation. The simple task is the mental arithmetic task from (Elio, 1986), which had a shallow hierarchy of operations for sequential mathematical routines, and in which humans took 2 to 12 seconds to complete the calculations.

The hypothesis is that latency from switching procedure contexts (as task sets) can model both time scales, according to the different amount of switching needed for each task. Below, each task is described before model results are presented.

Complex Task: Editors

In the editors task (Singley & Anderson, 1985), typists modified documents according to written edit directions, such as replacing one word with another or deleting a sentence. Three keyboard-only editors were used with which participants had no prior experience: ED, EDT, and EMACS. These each require different keyboard commands, with ED and EDT also differing from EMACs as single-line editors.

The 1985 experiment took place over six days. To test transfer, some participants switched editors after two days. If a participant spent two days each on ED, EDT, and EMACS in that order, this is called ED-EDT-EMACS. Figure 4a shows human data for transfer from ED to EDT-EMACS. Other configurations were tested, but are omitted here because results were similar. EDT performance after two days of ED is almost as fast as after two days of EDT, indicating substantial transfer. There is similar transfer to EMACS on day five. (EMACS-only users required 80 sec on day 1, not shown.)

Simple Task: Arithmetic

Elio's (1986) mental arithmetic task involved calculating hypothetical pollution rates based on water samples. Subjects repeatedly performed mental calculations such as shown in Table 1 using given input values. Human subjects were trained in a calculation sequence until they performed it on input data with perfect recall. They were then tasked with performing it 50 times on various inputs. Following training, subjects were assigned 50 trials using a different sequence of operations than the one memorized. Subjects were assigned one of three types of new procedures: transferred integrative, transferred component, and a control. Transferred integrative procedures shared with the memorized procedure all steps that required remembering intermediate results from previous steps. Transferred component procedures instead shared steps that required only checking input values directly. No steps were shared in the control.

Table 1: Example calculation sequence for the arithmetic task. Intermediate calculation values are marked with italics.

Step	Calculation	Op Type
1: *Particulate rating*	Solid $\times$ (lime$_4$ $-$ lime$_2$)	Component
2: *Mineral rating*	greater of $(\text{algea}/2)(\text{solid}/3)$	Component
3: *Index 1*	*Particulate + Mineral*	Integrative
4: *Marine hazard*	$(\text{toxin}_{\text{max}} + \text{toxin}_{\text{min}})/2$	Component
5: *Index 2*	*Index 1/Marine*	Integrative
6: **Overall quality**	*Index 2 - Mineral*	Integrative

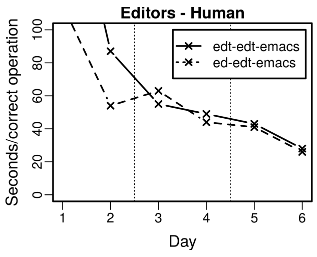
Human performance is shown in Figure 5a. Trials 1-50 show the reported power-law fit to training performance. Trials 51-100 show performance in the transfer conditions. Elio's reported transfer data show the mean from the first and last 25 trials per subject.[1] In the original study, results for component and integrative training calculations were reported separately. Only performance on component steps is shown here for brevity, as integrative results are comparable.

Transfer in the transferred component and transferred integrative conditions is evident by the faster initial performance for these cases, as expected due to the shared calculation steps. More interesting is the transfer to the control condition, which has no shared calculations. This indicates that calculation sequences share more than the specific operations.

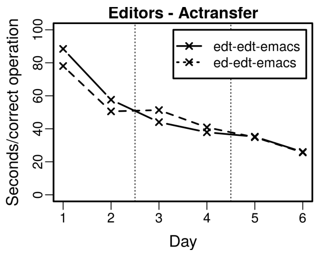
Actransfer models

The Actransfer models of both tasks arrange D-Rules so that similar operations, such as "ctrl-k" in EMACS or "d" in ED, which both delete the current line, or "algea / 2" and "Index1 / 2", which both divide, use similar primitive conditions and actions. This similarity allows transfer of P-Rules learned during practice, providing the results shown in Figure 4b and Figure 5b. Results match the general human transfer trends.

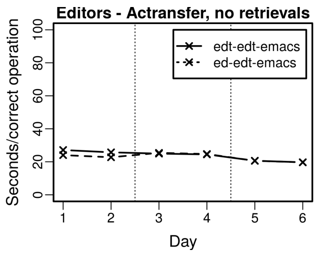
[1]The original 1986 human data is not available - only the data shown (R. Elio, personal communication, May 18, 2018).



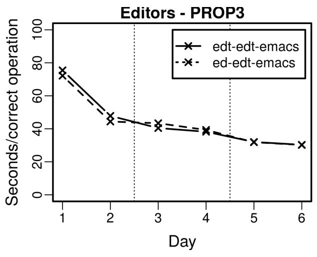
(a) Human task results from (Singley & Anderson, 1985).



(b) Actransfer model

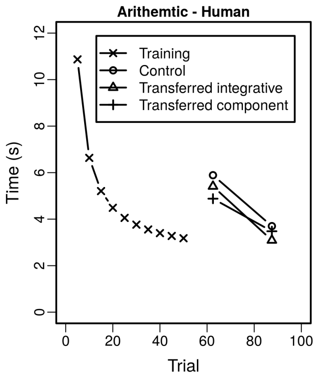


(c) Actransfer model, without declarative retrieval time

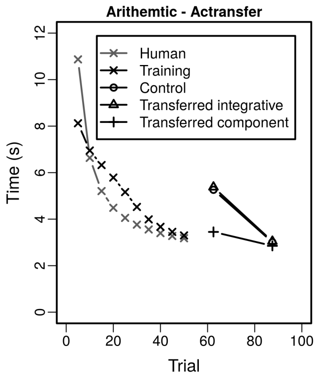


(d) PROP$_3$ model

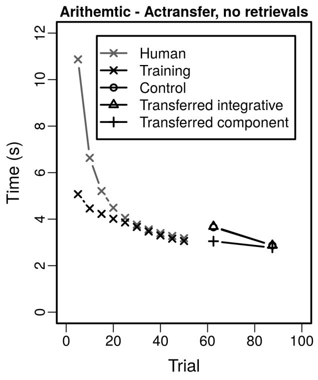
Figure 4: Editors task, human and model performance.



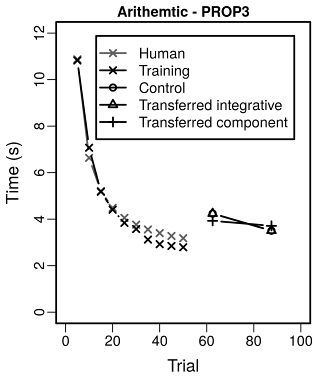
(a) Human task results from (Elio, 1986).



(b) Actransfer model



(c) Actransfer model, without declarative retrieval time



(d) PROP$_3$ model

Figure 5: Arithmetic task, human and model performance.

Figure 4c and Figure 5c show the same models, but when omitting latency due to Equation 1. The number of retrievals in these models decreases with practice, as the model learns P-Rules to execute the tasks more automatically and less by instruction, but the amount of simulated latency per retrieval also decreases as activation of retrieved memories increases with use. The figures show that decreasing latency per retrieval contributes significantly to the overall learning performance, especially in Figure 4b.

In the editors Actransfer model, $F_r = 1.5$, whereas in the arithmetic model, $F_r = 0.15$. The average latency per declarative retrieval in the editors task thus decreased from about 0.95 to 0.3 sec over days 1 to 6, whereas for the arithmetic model it decreased from about 0.07 to 0.03 seconds over the course of that task. This accounts for most of the difference in time scale between these two models, though the editors task also requires more decision making to complete.

PROP$_3$ models

The PROP$_3$ models use the differences in goal hierarchy structure to account for time scale differences in these tasks. Results are in Figure 4d and Figure 5d. Consistent with the hypothesis, the latency of switching contexts results in performance time scales comparable to both original Actransfer models, without requiring a task-specific scaling parameter.

In the arithmetic task, there is more transfer than exhibited by humans or the Actransfer model. This reflects how the model design treats identical calculation operations as identical subgoals, which are more fully transferred in PROP$_3$. This might indicate that humans do not mentally represent operations in quite so modular a fashion. This is consistent with Elio's (1986) findings, in which changes to integrative structure reduced component step transfer. A smoother power-law learning curve is also evident.

For the editors task, neither Actransfer nor PROP$_3$ models achieve human latency scaling in days 1-2. An early PROP model of this task explained this missing latency by modeling the learning of D-Rule structure during task trials (Stearns & Laird, 2018). PRIMs does not include a theory of the origin or modification of D-Rules in declarative memory, and Actransfer models do not include such behavior.

Experiment 2: Across-task switching

The second experiment tests the use of procedure contexts to model task switch costs by modeling the WM training experiment of Chein and Morrison (2010). Human subjects were trained for 20 days on a complex WM task. Subjects were sequentially shown single letters on a screen, and then asked to report the sequence. Between each letter was a distractor task of identifying if a word (e.g. "blick") was real.

Subjects were given a battery of cognitive assessments before and after training. Control subjects were assessed before and after 20 days of no training. One surprising result was from the Stroop task assessment, in which words describing colors (e.g. "red") were shown in fonts of a potentially different colors (e.g. blue), and subjects were asked to report the color of the font. Short pauses separated each trial word. WM interference was measured as the difference in average response time between when text and font colors were *incongruent* (e.g. "red" in blue font) and *congruent* (e.g. "red" in red font). The results are shown in Figure 6a. The horizontal axis shows the change in interference before and after the 20 day WM training period. Results showed surprisingly significant transfer from training toward reducing WM interference.

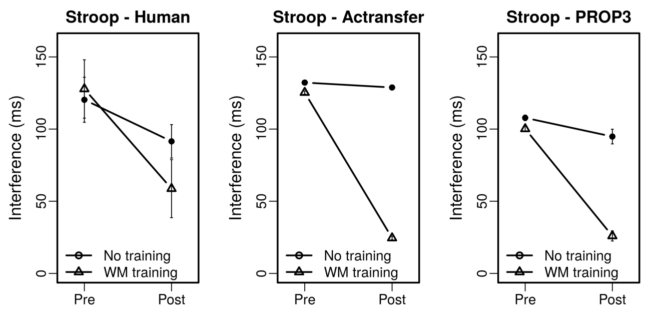
The hypothesis for Experiment 2 is that the Equation 2 model of task set switching can also account for the described Stroop interference effects. Each model is now presented.

Actransfer model

The Actransfer model uses transferred decision-making to explain this result. During pauses in the training task, the model prepares by rehearsing known letters. During pauses in the Stroop task, the model can either be idle or prepare to perceive font color. The practice of preparing during training biases the model to prepare in the Stroop task, since the prepare decisions in both tasks are composed of many of the same primitive operations, and both declarative instructions increase in activation during practice.

Interference is modeled via declarative retrieval latency, based on Equation 1. If there are conflicting stimuli in the model's visual WM buffer (the written color word and the font color), activation for declarative knowledge of each is diluted. But if the model chooses to prepare, only the presented font color enters its visual buffer. In that case, only that font color's memory is activated, and the undiluted higher activation leads to a faster retrieval time according to Equation 1.

Results in Figure 6b show substantial reduction in interference with training. Transfer is more extreme than that seen in human data, but the model demonstrates a working model of transferring decision making via practice.



(a) Humans (Chein & Morrison, 2010) (b) Actransfer model (Taatgen, 2013) (c) PROP$_3$ model

Figure 6: Stroop test results, before/after WM training.

PROP$_3$ model

The PROP$_3$ model represents WM interference as a failure to *prospectively* retrieve the correct procedure context before a trial prompt appears, consistent with the task set theory of building up a task set in preparation for stimuli (Sakai, 2008). A default Stroop response of reporting the written text is made available in the top-level procedure context, but reporting the font color requires retrieving a more detailed context as a subgoal. If the model prepares by retrieving the subgoal context before the prompt appears, performing this process does not delay the response once the prompt is shown.

As in the Actransfer model, between Stroop trials, the PROP$_3$ model has a choice between preparing or being idle. WM training teaches the model to prefer preparation. This is done using Soar's built-in Reinforcement Learning (RL) (Laird, 2012). An internal reward is given whenever a correct response is made, discounted over time. The faster a correct response, the greater the effect of the reward, and the more likely the choice is to be repeated.

Results are shown in Figure 6c. The model replicates human transfer trends, consistent with the hypothesis, though again, behavior is not an exact fit. It is clear, however, that the procedure context approach can achieve comparable behaviors to Actransfer by enacting task set-like computation.

Discussion and Conclusions

Though PROP$_3$ models replicate Actransfer behavior in both experiments without depending on long-term memory activation, Equation 1 has a long history of successful human modeling. There are two potential interpretations of these results. One is that procedure contexts and Equation 1 are two different computational explanations for the same phenomena, at different levels of abstraction. A second is that procedure contexts and Equation 1 explain different phenomena, where procedure contexts model latency related to procedural processes, and Equation 1 models more classical declarative processes. The tasks examined here primarily test procedure control and require few declarative fact recalls.

The idea of D-Rules as "declarative" knowledge comes from PRIMs theory as applied in Actransfer (Taatgen, 2013), but it is unclear whether D-Rules ought to be considered declarative in the psychological sense. It is important that D-Rules are accessible to the central executive in WM, as short-term structures that guide decision making. In the computational theories of ACT-R or Soar, any symbolic knowledge in WM has the form of "declarative" knowledge. However, in psychological theory, the term declarative usually refers to semantic or episodic knowledge that references facts or events in the outside world, not task procedures (Kump, Moskaliuk, Cress, & Kimmerle, 2015). Oberauer (2010) describes task sets as WM structures with procedural function, and calls them a form of procedural knowledge, although with a computational structure similar to that of declarative WM knowledge. This view of task sets as procedural WM knowledge is consistent with the idea that procedure contexts and Equation 1 model different processes. Regardless, results suggest Equation 1 need not account for as much of task latency as previously assumed in cognitive modeling, particularly with respect to procedural control tasks.

The PROP$_3$ models in this paper do not add substantial latency for retrieving declarative memories that are not procedure contexts, such as textual or mathematical facts relevant for tasks. The slightly faster time scale of PROP$_3$ results for the editors task might reflect the lack of fact retrieval latency in the model. The time scale for the arithmetic task model is not exceedingly fast, however. It is difficult to make definite conclusions without a more thorough investigation of how declarative fact knowledge should be represented and used in these models. Future work should explore whether procedure contexts imply any particular integration with more overtly declarative memory processes. Oberauer (2010) describes the theoretical declarative counterpart to task sets, "memory sets," which similarly mediate access to declarative knowledge in WM. The procedure context structure might be used in similar fashion to support memory set behavior. Future work might also explore on-line modification of procedure context structures as a model of subjects learning more accurate or efficient task set strategies.

One could attempt to encode hierarchical declarative structures in ACT-R comparable to procedure context structures. However, their use would seem problematic. First, ACT-R restricts WM buffers to hold only one declarative "chunk" at once, making *sets* of D-Rules difficult to represent concurrently in WM. Second, firing an individual ACT-R P-Rule corresponds to a decision, so that testing D-Rule conditions would require individual sequential *decisions* that update each set item every time the WM state changed. The extra processing would likely lead to unrealistic response times. Alternatively, in ACT-R one might plausibly replicate the net results of the procedure context model by treating goal name changes as a separate source of latency. This would effectively mean using Equation 2 in ACT-R as an extra equation that adds latency rather than describes existing computation.

The PROP$_3$ use of procedure contexts corresponds to representing procedural knowledge in WM, in the sense described by Oberauer (2010), demonstrating that it is possible to implement that approach to task sets in Soar. Moreover, this work demonstrates that the procedure context approach provides a single computational representation that generates human temporal profiles for procedural learning, transfer, and task switching, for a variety of tasks. This suggests a fruitful path of future research that might build upon the theory of procedure contexts to unify the fields of neuroscience, cognitive architectures, or beyond.

Acknowledgments

The work described here was supported in part by the ONR under Grant Number F048875-093099, and the AFOSR under Grant Number F050045-094301. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of ONR, AFOSR, or the U.S. Government.

References

Brasoveanu, A. (2015). *Intro to the ACT-R subsymbolic level for declarative memory.* Retrieved from `https://people.ucsc.edu/~abrsvn/ACT-R_subsymbolic_3.pdf`

Chein, J. M., & Morrison, A. B. (2010). Expanding the mind's workspace: Training and transfer effects with a complex working memory span task. *Psychonomic Bulletin & Review*, *17*(2), 193–199.

Dreisbach, G., & Haider, H. (2008). That's what task sets are for: shielding against irrelevant information. *Psychological Research*, *72*(4), 355–361.

Elio, R. (1986). Representation of similar well-learned cognitive procedures. *Cognitive Science*, *10*(1), 41–73.

Kump, B., Moskaliuk, J., Cress, U., & Kimmerle, J. (2015). Cognitive foundations of organizational learning: re-introducing the distinction between declarative and non-declarative knowledge. *Frontiers in Psychology*, *6*(1489), 1–12.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Oberauer, K. (2010). Declarative and procedural working memory: Common principles, common capacity limits? *Psychologica Belgica*, *50*(3-4), 277–308.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *10*(3), e1488.

Sakai, K. (2008). Task set and prefrontal cortex. *Annual Review of Neuroscience*, *31*(1), 219-245.

Singley, M. K., & Anderson, J. R. (1985). The transfer of text-editing skill. *International Journal of Man-Machine Studies*, *22*(4), 403 - 423.

Stearns, B., Assanie, M., & Laird, J. E. (2017). Applying primitive elements theory for procedural transfer in soar. In *Proceedings of the 15th international conference on cognitive modeling.*

Stearns, B., & Laird, J. E. (2018). Modeling instruction fetch in procedural learning. In *Proceedings of the 16th international conference on cognitive modeling.*

Taatgen, N. A. (2013). The nature and transfer of cognitive skills. *Psychological Review*, *120*(3), 439–471.

Taatgen, N. A. (2019). A spiking neural architecture that learns tasks. In *Proceedings of the 17th international conference on cognitive modeling.*