Scaling up from Micro Cognition to Macro Cognition: Using SGOMS to build Macro Cognitive Models of Sociotechnical Work in ACT-R

Robert L. West (Robert_west@carleton.ca)
Institute of Cognitive Science, 1125 Colonel By Drive
Ottawa, ON K1S 5B6 Canada

Sterling Somers (sterling@sterlingsomers.com)
Institute of Cognitive Science, 1125 Colonel By Drive
Ottawa, ON K1S 5B6 Canada

Abstract

SGOMS is a way of scaling up GOMS models to model multi agent work in complex sociotechnical systems. In particular, it allows GOMS to handle interruptions, multi-tasking, and re-planning. West and Pronovost (2009) discussed some of the theoretical issues around building an SGOMS model in ACT-R. This paper presents work in progress for two such models. Specifically, we used ACT-R to create a model of a worker in a sandwich shop and a model of a commercial airline pilot. Problems with scaling ACT-R up to the macro cognitive level are discussed and solutions are presented.

Keywords: ACT-R, GOMS, microcognition, macrocognition, sociotechnical systems, cognitive modeling.

As Simon (1962) pointed out in his story about the two watchmakers, when a system gets more complex it is more likely to be organized in a hierarchical way, especially if the system has to exist in a noisy environment. In the story, two watchmakers make identical watches but one goes out of business will the other one flourishes. The unsuccessful watchmaker does not have a hierarchical approach to making watches. Instead he makes a watch in one go, but if he is interrupted it all falls apart. The successful watchmaker makes various parts of the watch separately, and then assembles them together into higher-level parts, which are then assembled into even higher-level parts, and so on until the watch is made. If he is interrupted only the part he is working on falls apart. Because the watches are popular there are many calls to order them and therefore many interruptions, which are catastrophic for the first watchmaker but manageable for the second watchmaker.

SGOMS is a theory of Macro Cognition that is based on this idea. The argument behind SGOMS is that cognitive modeling designed to work at the level of psychology experiments (micro cognition) can scale up to higher-level tasks where interruptions are common, if additional hierarchical structure is added. If the system is truly hierarchical then the micro cognitive processes should not change when macro level processes are needed for a task. SGOMS is a theory about what sort of macro level processes need to be added. Specifically, SGOMS is a theory of what needs to be added to GOMS modeling (Card et al., 1983).

GOMS is based on the theory that all expert routine behaviors and cognitions can be explained using a limited set of control structures. These are: Goals, Operators, Methods, and Selection-Rules (implemented as Production Rules). In addition GOMS assumes that these elements are organized as Unit Tasks (Newell, 1990). Unit Tasks are a means of partitioning a task to avoid overloading the cognitive system (i.e., taking in information too fast) and downtime (i.e., taking in information too slowly). GOMS works well for modeling isolated, individual tasks. However, GOMS does not work well in environments where interruptions, multi tasking, and teamwork are common (West & Nagy, 2007). Simon's (1962) story suggests that this is due to insufficient hierarchical structure.

SGOMS augments GOMS by adding additional structure to allow the productive use of GOMS in complex sociotechnical environments. To accomplish this, SGOMS adds a higher-level control structure and introduces an additional constraint on unit tasks related to this structure. In addition to avoiding overload and downtime, unit tasks must also be small enough to avoid most interruptions (West & Nagy, 2007). By adding this constraint unit tasks become islands of work that will likely be completed without interruption and where normal GOMS modeling can be applied.

In SGOMS, control of the unit tasks is accomplished by *planning units*. Planning units are higher-level representations of the work that specify how to use unit tasks to accomplish a particular part of the task. In our experience so far, planning units can often be represented as a list of unit tasks. Planning units can be interrupted, bookmarked, and restarted; therefore planning units provide a cognitive mechanism to deal with task interruptions and multi-tasking.

West and Provnost (2009) characterized sociotechnical systems as having a tension between sticking to an existing plan and adapting to unforeseen local circumstances. Sociotechnical systems require that workers act in a coordinated way and this frequently involves having a plan. Planning meetings and briefing sessions are common in the sociotechnical workplace. However, it is often the case that workers have to adapt to or work around events or circumstances that were unforeseen in the plan. According to SGOMS theory, when a worker is interrupted and cannot continue on a planning unit, they use constraint-based decision making to switch to another planning unit. In an SGOMS model, each planning unit is associated with a set of constraints that can change based on events in the workplace. These constraints include constraints related to the plan; therefore, a worker can work around a problem (if possible) without disrupting the existing plan.

Planning units are also theorized to produce a *common ground* representation of the task (Klein, 2004), allowing workers to communicate what they are doing or to instruct others using the planning unit names. That is, planning units are meant to represent the level at which the cognitive representation of the work becomes shared. However, this does not imply that everyone's planning units would be the same; instead it would depend on their role in the planning unit. For example, a pilot and an air traffic controller might share a planning unit names for various parts of the landing sequence, but the contents of these planning units would not be the same.

ACT-R and GOMS

GOMS models are often implemented in ACT-R (Anderson & Lebiere, 1998). The reason for this is that there are similarities between ACT-R and GOMS, and ACT-R is capable of expressing all of the functionality needed for GOMS modeling. In fact, it is arguable that a GOMS model represents the simplest and most direct way to express something in ACT-R. However, it is important to note that ACT-R has more functionality than GOMS (the biggest difference being that ACT-R can learn) and that ACT-R has a detailed sub symbolic model for each type of functionality, which GOMS does not have. Therefore, expressing a GOMS model in ACT-R means adopting specific ACT-R assumptions defining the sub-symbolic systems. This is important to note because expressing a GOMS model in a different architecture could result in important differences.

It is also important to note that there are different versions of GOMS. The four most well known systems are the original version of GOMS, known as CMN-GOMS (Card et al., 1983); the keystroke level model, KLM (Card, Moran & Newell, 1980); NGOMSL (Natural GOMS Language, Kieras, 1996); and CPM-GOMS (Cognitive-Perceptual-Motor GOMS, John, 1988, 1990). To be clear about the relationship between ACT-R and GOMS in our model we defined our GOMS/SGOMS mechanisms in act-r terms. Please note that these are our definitions and may differ from definitions elsewhere in the GOMS literature.

Goals

In ACT-R the productions (i.e., selection rules) are triggered by the content of the buffers, which contain chunks. Chunks contain a limited number of predicate information bits (e.g., isa:dog name:rover color:brown). Productions fire if their *if* condition matches the buffer contents. ACT-R has a number of different buffers to represent the activity of different modules (e.g., vision, audition, declarative memory, motor, visual imagery). ACT-R also has a specific goal buffer, which represents where it is in a task. However, we defined the goal as the contents of all the buffers, which is the de facto definition of goals in most ACT-R models. In fact, the term goal is misleading as all of the buffers, including the goal buffer, represent the current state of the system, not a goal that the system wants to achieve. The goal directed behavior of the system is emergent from the contents of the buffers and the productions that they trigger

Operators

We defined operators as the lowest level at which a production in ACT-R can hand off an action to a module. In ACT-R, once an action has been handed off to a module, that action can occur in parallel with the actions of productions, so the level at which an operator is defined is important. However, there is no firm theory about this. Instead, judgments about the size of operators are based mainly on introspection about the task. For example, move hand to mouse, would be a typical GOMS operator because intuitively we feel that, in most cases, we do not consciously guide our hand to the mouse. However, we could break it down more if we wanted to. For example, I could consciously choose to guide my hand over the mouse and then consciously place my hand on the mouse, which would be two operators. To deal with this we propose that there is a level of natural operators that the motor system commonly uses and that, while it is possible to consciously control actions below this level, it requires a deliberate decision to do so and does not often occur. In addition to introspection, research on motor actions should be used to define these actions.

Methods

In GOMS, methods are sets of actions that are commonly repeated in a task. In ACT-R, if a set of actions is frequently repeated in the same order the actions will become compiled. What this means is that instead of using productions to retrieve information about what to do next (typically stored in declarative memory) the compiled version will consist of productions that directly call each other in a fixed order and therefore fire ballistically, as a set. This allows ACT-R to model the process of moving from deliberate actions to automatic behaviors, although in GOMS models the learning part is not needed since experts would be expected to have optimized sets of compiled productions. Therefore, we defined methods as compiled sets of productions. This puts a limit on the size of methods because in order to compile, the sequence must be exactly the same and must occur relatively quickly. Therefore, anything with variability or with delays introduced from the environment will not become a method.

Selection rules

Selection rules in ACT-R are production rules, so this is exactly the same. However, we found it useful to make a conceptual distinction between task-related productions, i.e., productions associated directly with the task, and system production rules, i.e., production rules that implement the general mechanisms for executing expert behaviors in complex environments. Although the ACT-R architecture treats all productions equally, it is the case that productions can be specific to a task or generic.

Unit tasks

In ACT-R, a unit task is a set of related productions, including compiled productions, or methods, that call perceptual, motor, and cognitive operators to complete the unit task. As in GOMS, unit tasks are about how the task is divided up

Planning units

We represented planning units as sequentially chained chunks in declarative memory. Each chunk represents a unit task and also the unit task that should follow it. A planning unit is executed by a set of productions that retrieves the next unit task in a series after the current unit task is finished. Each planning unit also separately stores a set of constraints, information about where the agent is in the planning unit, and information about the task that is non-routine. These are stored separately as chunks and allow the agent to judge if a planning unit is appropriate for the situation, to remember where they were in a task when they return to it, and to incorporate non-routine information, such as exceptions in the plan and unexpected events by altering the chunks.

Constraint based decision-making

The selection of productions in ACT-R is a form of constraint-based decision-making (i.e., the contents of the buffer provide the constraints for choosing a production). However, the time scale for ACT-R production selection seemed wrong. West and Nagy (2007) found that workers could spend a considerable amount of time ruminating on these decisions if the situation was complex. They also found a lack of uniformity at this level. Our belief is that this process is, itself, a form of expertise, and not a direct product of the architecture. In ACT-R terms, this means that the constraint based decision-making system would be built out of productions and chunks and could be considerably different across tasks and across individuals for non-routine decisions. For example, West and Nagy (2007) found that network maintenance workers often had to consider a large number of constraints when deciding how to respond to an unexpected event. In ACT-R terms this means that the information needed to make the decision exceeded the capacity of the buffers to hold it. Therefore, West and Nagy (2007) proposed that the workers were using memory-based heuristics to cope with this.

Models

The claim behind SGOMS is that all expert tasks can be described with these mechanisms. Creating an ACT-R model of how these processes operate and interact provides a much more rigorous and testable model. It also provides a test for ACT-R to see if it has the functionality to scale up to the macro level. West and Provnost (2009) discussed how this might be done. However, the key is to actually build and test models of different tasks. The goal is to show that our ACT-R model of SGOMS can parsimoniously model different types of expert tasks, or to falsify it by showing that this is inherently problematic. To do this we chose two tasks, working in a busy sandwich/wrap restaurant (located on our campus) and landing a large commercial jet airplane.

In addition to specifying how to build a model, SGOMS is also a way of observing and analyzing human behavior. Essentially, the SGOMS structure specifies what to pay attention to and what to analyze in order to get the data to build the model. In turn the model provides an organized, principled way of understanding the data. The first phase of data gathering involves the researchers tentatively filling in as much of the model as possible by reading manuals and interviewing experts. Later phases involve iteratively testing and adjusting or amending the model (see West & Nagy, 2007). In this paper we report our progress at modeling data from the first phase.

Sandwich making model

We began by trying to make a simple model of an expert sandwich maker working in a sandwich shop. The model was, on the surface, very simple. The worker could make different types of sandwiches, work the cash, or clean up (each of these was a planning unit). However, we immediately ran into an issue. SGOMS needs to keep track of a lot of information. In particular, at any given time a worker knows what planning unit they are in, what unit task they are in, what method they are doing, and where they are in the method. While this amount of information can be placed in a single chunk in the goal buffer, the size of the chunk is very large and this goes against ACT-R theory. Therefore, we created a set of goal buffers, with a buffer for each of the SGOMS levels (i.e., planning units, unit tasks, methods, and operators). This solution is very similar to Salvucci & Taatgen (2008) who added more goal buffers to allow ACT-R to multi-task. However, they added one buffer for each task, which is different from the way we did it. These differences are discussed below but here we wish to note the convergence of opinion that multiple goal buffers are needed for ACT-R to be flexible at the macro cognitive level.

A second issue that arose was the need to interrupt unit tasks with environmentally driven events. This requires that the system be open to bottom up information from different modalities (e.g., such as seeing the cheese on the floor, or hearing something drop). ACT-R is driven by the production system representing procedural memory, which can be thought of as a top down system. However, this system can receive bottom up information from the environment through the perceptual modules and react to it by firing a production. For example, there could be a production that fires when the auditory buffer contains the sound of a fire alarm. If this production has a higher utility it will fire instead of the top down, goal driven productions. However, using this approach means that the production system can be interrupted at any point, which creates an ACT-R programming nightmare. A second, related problem is what happens when two bottom-up interruptions occur close together. In this case the reaction to the first interruption is immediately canceled in order to react to the second interruption, which creates more chaos. Although it is possible to use ACT-R in this way, the problems created suggest that this aspect of the architecture is incomplete.

Our solution to this was to add a module to ACT-R representing the activity of the amygdala. The amygdala is widely believed to function as a monitoring system for negative events (Sergerie, Chochol, & Armony 2008; Damasio, 1994). However, in order to do this the amygdala module needed to function in a way similar to the procedural module. Therefore, we created a second, parallel production system to model the amygdala. However, the productions it contains are merely reactionary, with each production representing something the system finds alarming. The amygdala module monitors the buffers of the perceptual modules and if there is a match to a production it fires and places a chunk in the amygdala buffer indicating a problem and the modality that detected the problem (so attention can be directed there). This releases the procedural module production system from having to react to an interruption immediately when the interruption occurs. Under these conditions the procedural module production system can monitor for interruptions within a cycle of executing methods and/or operators. Furthermore, by making the system for dealing with interruptions interruptible in the same way, further interruptions can be dealt with by simply updating the representation of the situation and restarting the system for dealing with interruptions.

With these augmentations in place we were able to model the sandwich maker in a robust way so that they could make sandwiches, respond to unexpected events in intelligent ways, and work around problems. Although it was a relatively simple task we required the agent to be able deal with an interruption at any point and also to respond differently to different types of interruptions. As far as we can see, this would be difficult to achieve without our augmentations, and the model would certainly be less parsimonious.

Airplane pilot model

Our second task was to model the landing procedures on a commercial airliner. Our eventual goal is to produce a detailed model of the distributed cognition analysis presented in Hutchins (1995). To start on this model we used the Microsoft Flight Simulator and the X Plane flight simulator to gain a basic knowledge of landing procedures. We also discussed how we were modeling the procedure with actual pilots. This task was considerably more complex than making a sandwich and also had more real time components.

The task fit well into the planning unit/unit task framework, although we will follow this up by having real pilots evaluate the model. We found that unit tasks often reoccurred in different planning units, which was handled easily and conveniently by our model. However, we also found that the task involved a lot of monitoring and adjusting, which is different from a unit task that steps through a series of procedures towards a goal. For this we introduced “looping” unit tasks. These are unit tasks that continuously repeat until they are interrupted. Normally we think of interruptions as a bad thing, but in this task we found that task interruptions could function in a positive way, as a signal to change unit tasks. For example, a voice coming over the radio can act as signal to monitor the radio. However, decisions have to be made about how much information can be handled by the modules processing bottom up information. For example, we initially thought the auditory module should monitor the radio for the call sign of the airplane but our experience with the flight simulator indicated that novice pilots cannot do this. Instead, based on our own experience, we had the auditory module monitor for a voice over the radio, which could trigger a top down monitoring to listen for the call signal. The issue of how much intelligence can go into modules that monitor the environment for information is important as it can make a big difference in these models. An interesting question related to this is whether these modules can learn to handle more, as in the case of an expert, or if the limitations are fixed.

Our discussions with real pilots also revealed some differences in terms of the constraint satisfaction process used to select planning units. West and Nagy (2007) had found that network maintenance workers often considered large numbers of constraints and used heuristics to make decisions. In contrast, pilots often have definite rules or decision-making schemes. For example, if a pilot is landing and an incursion occurs on the runway they must switch to aborting the landing. Because this rule is simple it can be modeled with a single production rule. Also, pilots are taught clearer priorities. For example, they learn first to aviate, then to navigate, and last to communicate. They are also taught rules for specific instances. For example, on the last part of the landing they explicitly ignore radio communications. This finding was consistent with the claim that constraint satisfaction methods will be task dependent (West and Nagy 2007).

Multi tasking

As mentioned above, our approach to multi tasking is similar to Salvucci & Taatgen (2008) in that we increased the number of goal buffers, but it is also different. One point that is important to keep in mind is that the Salvucci & Taatgen (2008) model is not a model of experts or of macro cognition. The Salvucci & Taatgen (2008) model does a good job of modeling how people learn to multi-task in labbased situations. We have no story about learning; instead we based our model on the broader requirements of performing complex tasks in busy environments. In Salvucci & Taatgen's (2008) model, each task gets one goal buffer. This allows switching between tasks without loosing track of where you are in the task. Salvucci & Taatgen (2008) also use a greedy polite resource-sharing model, where resources are the modules that receive instructions from the procedural production system. Greedy means that task productions will take over a module if it is available and needed. Polite means that if one task is controlling a module, when it is finished it will not re-take it, but instead will let another task use it.

As Salvucci & Taatgen (2008) note, there are other models of multi-tasking with different abilities. These include the ability to delay, interrupt, and prioritize (Freed, 1998), the ability to direct multi-tasking using schemas (Norman and Shallice, 1986), and the ability to adapt to constraints (Howes et al., 2004, 2007). Essentially our goal was to develop a model that could do all of these things. However, we also made things easier for ourselves by focusing only on experts, thus ignoring the problem of how people learn to do these things. Our way of doing this was to use the ACT-R production system and the ACT-R declarative memory system to build a hierarchical control structure based on SGOMS. Within this hierarchy, multi-tasking can occur in different ways.

At the level of switching planning units, multi-tasking involves a constraint-based decision each time the task is switched. However, if the there are simple rules for this, as sometimes occurs with pilots, the switch can be achieved by a single production rule and is therefore quite fast. The more complex the constraint based decision process, the longer the decision would take. As noted above, we believe that the constraint based decision process is specific to the task.

Multi-tasking can also take place within a planning unit. This could occur if a planning unit required two or more distinct tasks to be completed. This could involve using a deliberate schema for switching between the unit tasks associated with each of the tasks. In the simplest case, this could be expressed by the order of the unit tasks in the planning unit.

Finally, multi-tasking can take place within a unit task if the multi-tasking is a normal routine part of the unit task. In this case we would expect the multi-tasking to be managed by productions in the unit task in a way that is appropriate to the task. The question of what happens below this brings up an interesting question, which is, when does something cease to be multi-tasking and become a single task? Methods involve coordinating different activities, as do operators, so really we are always multi-tasking.

Micro versus Macro

Psychology experiments are designed to get at micro cognitive processes and to avoid the influence of macro cognitive processes. To do this, experiments often involve abstract and artificial tasks. This is useful and good because it allows us to get at the micro mechanisms that underlie macro level behavior. Salvucci & Taatgen's (2008) model of multi-tasking is a good example of a micro theory. It is a way of understanding and experimenting on some of the factors that underlie real world task switching. SGOMS is a macro level theory. It is aimed instead at understanding how real world multitasking can be understood in the context in which it occurs.

In our view the macro and micro levels should be complementary. For example, consider someone making a cell phone call while driving, if it is the first time they have done this then they are not experts, SGOMS does not apply, and the Salvucci & Taatgen model would be a good choice. However, if they frequently and routinely socialize on the cell phone while driving (e.g., some taxi drivers) then SGOMS predicts that they could have acquired expertise at switching between their planning unit for driving and their planning unit for socializing via cell phone. This would take the form of a more sophisticated understanding of the constraints involved in phoning and driving.

Alternatively, consider a husband who picks up his wife everyday from a busy downtown office building where there is no stopping allowed. To get around this he uses his cell phone to call her just before he gets there so she can come out, but otherwise he does not use the cell phone in the car. In an SGOMS model the cell phone call and the driving would be part of the same planning unit – picking up his wife. Because there is no switching between planning units SGOMS does not predict the use of constraint based switching, instead SGOMS predicts the use of rules specific to that situation (e.g., phoning when in a particular merging lane that is always slow). However, in all cases, the micro cognitive bottlenecks for resource allocation that underlie the Salvucci & Taatgen model, and ACT-R in general, would still apply. SGOMS is a theory of what else may apply in specific contexts. In other words, it is a way of scaling up micro cognition to macro cognition.

In terms of buffers, Salvucci & Taatgen (2008) use different buffers to represent the goals of different tasks whereas we use different buffers to represent different levels of information about the task. As noted above, the most important thing is the consensus that a single goal buffer is not enough. If we were to allow for multiple buffers the most natural place would be within planning units to represent different unit tasks.

Conclusion

We have reported our progress so far in building two SGOMS models of very different tasks. Building the models is part of the SGOMS process, which involves iterating between model building and gathering real world data. The claim of SGOMS theory is that one cognitive framework can be used to model all expert behavior. To test this and to make SGOMS more specific we want to try SGOMS on many different tasks. So far, we have not found a task that cannot be easily accommodated by the SGOMS structure.

In terms of implementing SGOMS in ACT-R, two issues were found. The first was the need for a hierarchical goal structure instead of the single goal buffer that ACT-R uses. However, it is important to note that the extra buffers are to keep track of activity within disruptive, multi-agent environments and are not needed to model Psychology experiments, which almost always involve simple, isolated tasks. The other thing we found was the need for a system to evaluate bottom up information. We created such a system by modeling the amygdala as a production system, with the caveat that it can contain only simple, reactive productions. Something like this is needed to allow the ACT-R procedural memory production system to deal with bottom up interruptions, otherwise, in a noisy environment, the procedural memory production system would be constantly interrupted in order to evaluate bottom up information.

References

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

Card, S., Moran, T., & Newell, A. (1983) *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Damasio, A.R. (1994). *Descartes' Error: emotion, reason, and the human brain.* New York: Grosset/Putnam.

Freed, M. (1998). Managing multiple tasks in complex, dynamic environments. In *Proceedings of the 1998 National Conference on Artificial Intelligence*, Madison, Wisconsin, 1998.

Howes, A., Vera, A., Lewis, R.L., and McCurdy, M. (2004). Cognitive constraint modeling: A formal approach to supporting reasoning about behavior. In Proc. Cognitive Science Society.

Howes, A., Vera, A., Lewis, R.L. (2007). Bounding rational analysis: Constraints on asymptotic performance. In W. D. Gray (Ed.) *Integrated Models of Cognitive Systems* (pp. 403-413). Oxford University Press.

Hutchins, E. (1995). How a Cockpit Remembers Its Speeds. *Cognitive Science, 19*, 265-288.

Klein, G., Woods, D. D., Bradshaw, J. D., Hoffman, R. R., and Feltovich, P. J. (November/December 2004). Ten challenges for making automation a "team player" in joint human-agent activity. IEEE: Intelligent Systems, pp. 91-95.

Newell, A. (1990). *Unified theories of cognition.* Cambridge: Harvard University Press.

Norman, D. A., & Shallice, T. (1986). Attention to action: Willed and automatic control of behavior. In R. J. Davidson, G. E. Schwartz, & D. Shapiro (Eds.), *Consciousness and Self-Regulation* (pp. 1-18). New York: Plenum.

Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review*.

Sergerie, K., Chochol, C., & Armony, J. L. (2008). The role of the amygdala in emotional processing: A quantitative meta-analysis of functional neuroimaging studies. Neuroscience and Biobehavioral Reviews 32, 811–830.

Simon, H. (1962). The Architecture of Complexity. *Proceedings of the American Philosophical Society, 106:6*, 467-482.

West, R.L., & Nagy, G. (2007). Using GOMS for modeling Routine Tasks Within Complex Sociotechnical Systems: connection Macrocognitive Models to Microcognition. *Journal of Cognitive Engineering and Decision Making.*

West, R.L. & Pronovost, S. (2009). Modeling SGOMS in ACT-R. Linking macro and micro cognition. *Journal of Cognitive Engineering and Decision Making.*