Submission to 2015 ACT-R Workshop for Demonstration

# ACT-Concurrently: Concurrency Work-Around for ACT-R

**Franklin P. Tamborello**
National Research Council Postdoctoral Research Associate
U. S. Naval Research Laboratory, 4555 Overlook Ave SW Washington, DC 20375, USA

## Abstract

As models grow more complex, parallelization of model runs becomes increasingly important for completing sets of model runs in a reasonable amount of time. Solutions exist to bring to bear high-performance computing resources to search parameter spaces, such as MindModeling@Home and Adaptive Mesh Refinement (Best et al., 2009). However, sometimes it is helpful to have many model runs to evaluate a model's algorithm early in development, before searching a parameter space. A parallelization scheme that is easy to configure and load could fill such a role.

The canonical ACT-R implementation, i.e., the Lisp version maintained by Dan Bothell that serves as a reference point for the ACT-R community, has much legacy code that would be time-consuming to rewrite such that ACT-R would be thread-safe, so that many model runs could execute in parallel within one loaded instance of ACT-R. I wrote a work-around which allows many ACT-R instances, each running within their own Lisp image, to work together to execute many model runs in parallel on one or many machines.

"ACT-Concurrently" loads with ACT-R when placed in the user-loads directory. Running in "manager" mode, it reads in a model file and parses it into Lisp objects. It then transmits that model to other ACT-R instances running in "worker" mode. The worker nodes, having already gone into a ready and listening state, read and execute the transmitted model. Each worker transmits back to the manager node the return value of the model run call.

With ACT-Concurrently, the effort needed by the modeler to parallelize model runs should be suitable for novices. All that is required is to place the loader file and ACT-Concurrently directory into the user-loads folder of each ACT-R instance and to set two visibly-commented global variables for each worker and two more for the manager. This makes available to the modeler all the cores on a modern machine as well as cores on other machines.

## References

Best, B. J., Furjanic, C., Gerhart, N., Fincham, J. M., Gluck, K. A., Gunzelmann, G., & Krusmark, M. (2009). Adaptive mesh refinement for efficient exploration of cognitive architectures and cognitive models. In *Proceedings of the 9th international conference of cognitive modeling*.

MindModeling@Home. (n.d.). MindModeling@home. [Web page] Retrieved from mindmodeling.org