ACT-Concurrently Reference Manual

Working Draft

Franklin P. Tamborello, II, PhD, CSP
National Research Council Research Associate
U. S. Naval Research Laboratory

Last revised 7/8/15 for act-concurrently.lisp revision 21.

Acknowledgments

- Dan Bothell (Carnegie-Mellon University) for his comments regarding random number generator seeding.

Introduction

This document describes the ACT-Concurrently concurrency work-around for ACT-R. The canonical ACT-R distribution is not thread-safe, but ACT-Concurrently provides a concurrency work-around to take advantage of a multi-core machine as well as other locally-networked machines. For both cases, ACT-Concurrently establishes a TCP socket connection from a user-designated ACT-R instance running as the "manager node" to one or more ACT-R instances, each running in their own Lisp images as "worker nodes."

Use of this software requires some basic knowledge of TCP/IP networking. You must know how to find the IP address of a machine and an available port for use. These topics are beyond the scope of this document. So far this software has only been tested with ACT-R 6.0 (revision 1,646) and Clozure Common Lisp on Macintosh OS 10.10.

Setup

Place the entire distribution in your ACT-R user-loads folder. ACT-Concurrently will thereafter load with ACT-R. If you want to save settings such as worker addresses and ports to the act-concurrently.lisp file so that it will be ready to go each time ACT-R loads, then for each node make a separate copy of your ACT-R installation and modify act-concurrently.lisp as follows. Note that ACT-R automatically compiles each file in **user-loads** before loading for the first time, so if you have already loaded ACT-Concurrently then you will need to delete the binary file compiled from **ac-loader.lisp** between making any changes to ACT-Concurrently and loading it. Otherwise you may use just one ACT-R installation and set the values of the following parameters after act-concurrently.lisp has loaded.

Worker Nodes

A worker node may be an ACT-R instance loaded into its own Lisp image either on the same machine as your manager node or on a remote machine. For each worker node, modify its copy of act-concurrently.lisp as follows. Find the line that says **(defvar *worker-address* nil)**. Replace **nil** with that machine's IP address. Format the IP address as a string of four numerals separated by periods, "127.0.0.1" as an example. Do the same for ***worker-port***, immediately below that. Set ***worker-port*** to an integer. Note that each combination of address and port must be unique to that worker node and that the port must be otherwise unused by that machine and the manager. If you opted to have a dedicated copy of ACT-R for this worker node, there is a commented call to **(worker-listen-for-job)** which you may uncomment (delete the leading "**;** ") so that when ACT-R loads the worker will start itself into its ready state. Your worker node should now be ready to load all its settings with loading of ACT-R.

You may leave your worker node listening for jobs for as long as you like. However, should it encounter an error when attempting to run your model you will probably have to respond to the error and then call **(worker-listen-for-job)** again to make it ready again.

Manager Node

For the manager node, scroll down a bit farther in act-concurrently.lisp. Set ***manager-p*** to **t**. Set ***model-run-call*** to a quoted expression, whatever you use to start your model running if after loading the model file you must call something, such as **(run-model :n 3)**. Otherwise, if you place your model run call in your model file then set this value to **nil**. Set ***worker-addresses-and-ports*** to a list of dotted pairs of addresses and ports of your workers as in the example in the comments in that section of act-concurrently.lisp. Place your model file into the **act-concurrently** directory within ACT-R's **user-loads** directory and name it **model.lisp**. Your manager node should now be ready to load all its settings and your model with loading of ACT-R.

Use

Call **(send-out-jobs)** to transmit your model to your worker nodes and have them run it. Whatever your model run function returns, each worker will return that to your manager. Your manager will save each worker's return value into a list. After all workers have completed their jobs, you can call **read-mail-data** to save these values into a global variable, ***data***.

Security Caveat

It should be noted that this software is not secure and for that reason I recommend that you run it only on machines within a local subnet behind a firewall. No encryption is performed on the messages transmitted between nodes, including the arbitrary Lisp code that the workers evaluate.