? Reference

Here is a short summary of top-level 4CAPS commands. In general, commands apply to all modules. However, affixing an "@" to the command name generally allows you to execute the command in the specified modules.

1 Managing Modules

1.1 Listing Modules

(MODS): Print the names of all modules..

1.3 Adding Modules

(ADD-MOD mod-name): Create a a new module with name mod-name. If a module of this name already exists, then do nothing.

1.4 Deleting Modules

(DEL-MODS): Delete all modules.

(DEL-MODS@ mod-name): Delete the module with name mod-name. If no such module exists, this command is ignored.

2 Individual Modules

2.1 Module Working Memory

(WM {wm-class}^*): Prints all wmes in the system that are of the same class or a sublcass of at least one of the wm-class arguments. If no wm-class arguments are supplied, all wmes that are sublcasses of BASE-WME (i.e., all wmes) are printed.

(WM@ mod-name {wm-class}^*): Prints the wmes of the module named mod-name that are of the same class or a subclass of one of the wm-class arguments, or all wmes of subclass BASE-WME of no wm-class arguments are supplied. If no such module exists, no action is taken.

(GET-WME num): Prints/returns the wme with an id of num, or NIL if no such wme exists in any module's working memory. Useful with top-level calls to MODIFY.

2.2 Module Productions

(MATCHES): Print the LHSs of all instantiations of all modules that will be fired on the next cycle (assuming no changes to working memory in the interim).

(MATCHES@ mod-name): Print the LHSs of all instantiations in the module name mod-name that will be fired on the next cycle. If no such module exists, no action is taken.

2.3 Module Capacity

(CAPS): Print the activation capacities of all modules.

(CAPS@ mod-name): Print the activcation capacity of the module named mod-name. If no such module exists, no action is taken.

(SET-CAPS num): Set the activation capacities of all modules to num.

(SET-CAPS@ mod-name num): Set the activation capacity of the module named mod-name to num. If no such module exists, no action is taken.

2.4 Module Specializations

(SPECS {wm-class}^*): Print the specialization of each module for each wm-class. If no arguments are supplied, the specializations for BASE-WME and all its sublcasses (i.e., all wm-classes) are printed. Note that if a module is not specialized for a wm-class (i.e., the specialization is NIL), then this is not printed in the interest of succintness.

(SPECS@ mod-name {wm-class}^*): Print the specialization of the module named mod-name for the supplied wm-classes, or for BASE-WME and all its subclasses if no wm-class arguments are given. If no such module exists, no action is taken.

(SET-SPECS {wm-class num}^*): Set the specialization of all modules for the specified wm-classes (and all of its subclasses) to be the corresponding nums.

(SET-SPECS@ mod-name {wm-class num}^*): Set the specialization of the module named mod-name for the specified wm-classes to be the corresponding nums. If no such module exists, no action is taken.

3 Working Memory

3.1 Working Memory Classes

(DEFWMCLASS wm-class ({superclass}*) {[attribute | (attribute default)]}*): Define a new working memory class with name wm-class. It inherits from each listed superclass. (If no superclasses are explicitly provided, the new class will inherit from BASE-WME, as do all wm-classes.) In addition to whatever attributes it inherits from its superclasses, it will also contain any listed attributes. When a wme of class wm-class is created, any

attributes not explicitly given a value will be assigned a default value, the one indicated in the DEFWMCLASS form or, if none was supplied, NIL.

3.2 Working Memory Elements

(MODIFY wme attribute value): Modify the attribute of the wme to have the new value. (GET-WME is useful for retrieving a wme to be modified at the top-level.)

(SPEW {T | wme}^1 {wme-spec | wme}^1 weight): Spew activation from one wme to another. The first argument is the activation of the source. If T, the source is the "eternal source," which has an activation of 1.0. If it's a wme, then the wme's activation serves as the source. The second argument is the target of the spew. If a specification list describing a wme, then the specified wme is created. Otherwise, it is must be an existing wme. (Note that GET-WME can be useful for retrieving source and target wmes at the top level. Further note that when GET-WME is used with SPEW at the top-level, bugs seem to surface.) The weight argument modulates the proportion of the source activation that is spewed to the target.

4 Productions

5 Parameters

5.1 Default WME Threshold

(DEFAULT-WME-THRESH): Print the current default wme threshold, i.e., the activation threshold on wmes trying to match production LHSs assumed if none is explicitly supplied.

(SET-DEFAULT-WME-THRESH num): Set the default wme threshold to be num.

5.2 Delete Threshold

(DELETE-THRESH): Print the current delete threshold, i.e., the activation level below which wmes are automatically deleted from modules. This implements a form of garbage collection.

(SET-DELETE-THRESHOLD num): Set the delete threshold to be num.

5.3 Default Specialization

(DEFAULT-SPEC): Print the default specialization of modules for wm-classes. This default can be overriden with SET-SPEC and SET-SPEC@.

(SET-DEFAULT-SPEC num): Set the default specialization to be num.

5.4 Maximum Activation

(MAX-ACT): Print the maximum activation that wmes are allowed to accumulate. This value is a saefguard against positive feedback allowing a single wme to command to many resources.

(SET-MAX-ACT num): Set the maximum activation to be num.

5.5 Front WME Activations when Printing

(FRONT-ACT-P): Print whether the activations of wmes are printed before the wmes themselves ("fronted"), or whether the :ACT attribute is printed like the other attributes. Either T or NIL.

(SET-FRONT-ACT-P bool): Set whether wme activations are fronted to be bool.

5.6 Summarize Embedded WMEs when Printing

(SUMM-EMBEDS-P): Print whether when printing a wme, if the value of an attribute is another wme, a summary of the wme (i.e., its wm-class and id number) or the entire wme is printed. Either T or NIL.

(SET-SUMM-EMBEDS-P bool): Set whether embedded wmes are printed to be bool.

5.7 Elide NIL WME Attributes when Printing

(ELIDE-NILS-P): Print whether when printing a wme, if the value of an attribute is NIL, the attribute and value are elided completely or printed like non-NIL attributes. Either T or NIL.

(SET-ELIDE-NILS-P bool): Set whether NIL-valued attributes are elided to be bool.

5.7 Modules Outermost when Printing Simulation Histories

(MODULES-OUTER-P): Print whether, when executing the HISTORY command, the output tables are organized with modules as the outermost dimension. Consider the situation when this command returns T. Then if there is a single table with modules on one dimension and wm-classes on the other, then modules will occupy the rows. If there are multiple tables, then each module corresponds to a table, with wm-class and time information occupying the dimensions of the table. If the command returns NIL, then the proceeding statements are true, with modules and wm-classes reversed.

(SET-MODULES-OUTER-P bool): Set whether modules are printed outermost in history tables as bool.

5.8 Module Tracing

(TRACING-P): Print the tracing status of all modules (i.e., whether the instantiations fired on each macrocycle are printed).

(SET-TRACING-P bool): Set the tracing status of all modules depending on the value of bool.

5.9 WM Tracing

(TRACING-WM-P): Print the tracing status of wmess (i.e., whether the modular activation contributions for each wme are printed on each macrocycle).

(SET-TRACING-WM-P bool): Set the tracing status of wmes depending on the value of bool.

6 Simulations

6.1 Initializing and Recording Simulation Histories

(INIT-ACT-HISTORY): Initialize the activation history, which tracks the capacity utilized for each cognitive function in each module over each subsequent macrocycle.

(INIT-SEGMENT-HISTORY): Initialize the segment history, which can be used via START-SEGMENT to collapse periods of simulation activity over macrocycles into a single epoch.

(END-SEGMENT symbol): Record in the segment history that the previous segment has ended and that all subsequent macrocycles of activity (until the next call to END-SEGMENT) are part of a segment denoted by symbol.

6.2 Module Resetting

(RESET): Resets the system by cleansing working memory, initializing both the activation and segment histories, resetting all modules (i.e., setting their macrocycle counts to zero and clearing their working memories, while leaving roduction memories intact), and resetting the macrocycle count to 0. It also resets the id counter from which newly created wmes draw their ids, easing comparisons of model activity across different simulation runs.

6.3 Running Simulations

(RUN [num]): Run the current model for num macrocycles. If the desired number of macrocycles is not specified, the model is run until activity in all modules ceases.

6.4 Printing Simulation Histories

(HISTORY [:WMES {wm-class | ({wm-class}^*)}^1] [:TIME {num | SEGMENT}^1] [:MEASURE {ACT | CAP}^1] [:COMBINATION {SUM | AVG}^1] [:MODULES-OUTER-P bool]): The history command summarizes the activation flows of a simulation for all modules.

If :WMES are not supplied, then the activation aggregated across all wm-classes is displayed. If a wm-class is supplied, then the activation devoted to that class is displayed, and if a list of wm-classes is supplied, then the activation consumed by each wm-class is shown.

If :TIME is not supplied, then activation is aggregated over all macrocycles of the simulation. If it is a number, then activation is aggregated over successive intervals, where each interval consists of num macrocycles (except possible the last one). The special case where num=1 shows activation flows at each macrocycle. Finally, if it is SEGMENT, then activation is aggregated over the macrocycles that define each segment of the simulation.

If :MEASURE is ACT, then the measure of interest is the activation consumed; if it is PROP, the the measure is the proportion of the current module's capacity consumed. If the current module has no fixed capacity, the PROP measure is always 0.0. If no :MEASURE is explicitly supplied, it defaults to ACT.

If :COMBINATION is SUM, then the measure of interest is summed over the time interval of interest. If it is AVG, then it is averaged over this interval. If not supplied, it defaults to SUM.

:MODULES-OUTER-P dictates the whether modules are listed on the outermost dimension of the history tables, or if wm-classes are. If not supplied, it defaults to the default indicated by MODULES-OUTER-P.

(HISTORY@ {nil | mod-name | ({mod-name}^*)}^1 [:WMES {wm-class | ({wm-class}^*)}^1] [:TIME {num | SEGMENT}^1] [:MEASURE {ACT | CAP}^1] [:COMBINATION {SUM | AVG}^1] [:MODULES-FIRST-P bool]): Same as the HISTORY command except that the history is printed for the module or modules indicated by the first argument. If the argument is NIL, then the table aggregates over all modules. If it is the name of a module, then the table is for this module. Finally, if it is a list of module names, then tables are produced for each corresponding module.