The Soar Cheat Sheet

For Soar 9.6

Written by Bryan Stearns
October 31, 2022

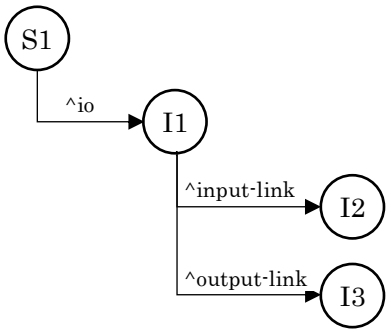
Rule Syntax (see manual page 49)

```
sp {production-name
    "Documentation string"
    :type
    (CONDITIONS)
    -->
    (ACTIONS) }
```

Type	Description (see manual page 202)
`:o-support`	Specifies that all RHS actions be given o-support
`:i-support`	Specifies that all RHS action be given i-support
`:default`	Specifies that this is a default production (for `production excise` and `trace` commands)
`:chunk`	Specifies that this is a chunk (for `production excise` and `trace` commands)
`:interrupt`	Interrupts Soar when this rule matches but before it fires
`:template`	Specifies that this be used to generate RL rules

Working Memory Element (WME) Terminology (see manual page 14)

A WME is a tuple of three elements: **identifier**, **attribute**, **value**
The syntax for referencing a WME within rules is
(<id-var> ^attribute value)



The memory graph on the right depicts 3 WMEs:

- (S1 ^io I1)
- (I1 ^input-link I2)
- (I1 ^output-link I3)

A group of WMEs that share an identifier are referred to as an **object** in working memory. The WMEs of an object are called that object's **augmentations**.

The memory graph on the right depicts 2 objects:

- S1, with 1 augmentation: (S1 ^io I1)
- I1, with 2 augmentations: (I1 ^input-link I2) and (I1 ^output-link I3)

SMEM

Common interface commands (see manual page 242):

Command	Description
`smem -e, --enable, --on`	Enables semantic memory
`smem -d, --disable, --off`	Disable semantic memory
`smem -c, --clear`	Deletes all semantic memories
`smem -i, --init`	Deletes all semantic memories if append is off
`smem -a, --add`	Add concepts to SMEM. Format: `smem -a {(<id> ^foo bar) (<id2> ^food bard)}`
`smem -r, --remove`	Remove concepts from SMEM. Format: `smem -r {(@34 ^foo bar)}` To remove all WMEs with ID:attr pair: `smem -r {(@34 ^foo )}` To remove all WMEs from an ID: `smem -r {(@34)}`
`smem -q, --query [<cue> [<num>]]`	Print concepts from SMEM matching some cue. Optional `<num>` will display the top `<num>` most-activated LTIs that match the cue.
`smem -b, --backup`	Create a backup of the SMEM database on disk
`smem --set append [on,off]`	Controls whether the database is overwritten or appended when loading. Default: `off`
`smem --set database [file,memory]`	Database storage method. Default: `memory`
`smem --set path [`*`empty, some path`*`]`	Location of the database file. Default: *`empty`*
`smem --set activation-mode [recency,frequency,base-level]`	Sets the ordering bias for retrievals that match more than one memory. Default: `recency`
`smem --set base-decay [>0]`	Sets the decay parameter for base-level activation computation. Default: `0.5`
`smem --set base-inhibition [on,off]`	Sets whether base-level activation has a short-term inhibition factor. Default: `off`
`smem --set spreading [on,off]`	Enables/Disables spreading-activation. Note: `activation-mode` must first be set to `base-level` to enable spreading.
`smem --set spreading-limit [0,1,…]`	How many LTIs can receive spread from an activation source. Default: `300`
`smem --set spreading-depth-limit [0,…,10]`	Limits depth of spread from any LTI. Default: `10`
`smem --set spreading-continue-probability [0.0,…,1.0]`	The multiplier that decays activation with each layer of spread depth. Default: `0.9`
`smem --set spreading-loop-avoidance [on,off]`	Controls whether any given spread traversal can loop back onto itself. Default: `off`

WM Link Schema (see manual page 145):

Command	Syntax	Description
Store	`<s> ^smem.command.store <id>`	Add the given object to SMEM or update it if already there
	`^smem.command.store-new <id>`	Add the given object as a new LTI in SMEM. If the given ID is already associated with an existing LTI, the ID in WM is not updated to be associated with the new LTI unless the `link-to-new-LTM yes` command is also given.
	`^smem.command.link-to-new-LTM yes`	See the description for the `store-new` command above.
	`^smem.result.success <id>`	Points to the ID used in the store command after the store completes
Retrieve	`<s> ^smem.command.retrieve <id>`	Retrieve the full object associated with the given ID
	`^smem.result.failure <id>`	If the id used in the retrieve command was not associated with an LTI
	`^smem.result.success <id>` `^smem.result.retrieved <lti>`	If an associated LTI was found for the given ID. The `<lti>` ID points to the newly-retrieved object.
Query	`<s> ^smem.command.query <q>`	The query pattern to search for in SMEM
	`^smem.command.neg-query <nq>`	Avoid retrieving an LTI that matches the given neg-query pattern.
	`^smem.command.prohibit <id>`	Do not retrieve the LTI associated with the given ID.
	`^smem.command.math-query .<attr>.<cond-type> <value>`	Conditional math query: Require that the retrieved LTI contains a WME with the given attribute and a value that meets the given math condition. Values must be numeric. Supported condition types are: {`less, greater, less-or-equal, greater-or-equal, max, min`}
	`^smem.command.math-query .<attr>.<cond-type>`	Superlative math query: Same as a Conditional math query, but does not use a specific value. For example, a superlative `foo.greater` command would require the retrieved LTI to have a `^foo <value>` that was higher than any other `^foo <value>` among candidate LTIs.
	`^smem.result.failure <q>`	If no LTI matching the query was found in SMEM
	`^smem.result.success <q>` `^smem.result.retrieved <lti>`	If an LTI was found that matched the given query. The `<lti>` structure points to the newly-retrieved object.
Any	`<s> ^smem.result.bad-cmd <scmd>`	If the contents of the smem.command object do not represent a valid command. The `<scmd>` variable links to the smem.command object.
	`<s> ^smem.command.depth <integer>`	For a retrieve or query command, retrieve an LTI and its descendants to the given depth.

EPMEM

Common interface commands (see manual page 252):

Command	Description
`epmem -e, --enable, --on`	Enables episodic memory
`epmem -d, --disable, --off`	Disable episodic memory
`epmem -b, --backup`	Creates a backup of the episodic database on disk
`epmem -v, --viz`	Print episode in graphviz format
`epmem --set append [on,off]`	Controls whether database is overwritten or appended when opening or re-initializing
`epmem --set database [file,memory]`	Database storage method
`epmem --set exclusions [`*`any string`*`]`	Toggle exclusion of a named attribute from stored episodes. Defaults: "`epmem`", "`smem`"
`epmem --set force [ignore,remember,off]`	Forces episode encoding/ignoring in the next storage phase. Default: `off`
`epmem --set learning [on,off]`	Enables episodic memory learning. (Automatically enabled when epmem is enabled.)
`epmem --set path [`*`empty,some path`*`]`	Location of database file. Default: *`empty`*
`epmem --set trigger [dc,output,none]`	How episode encoding is triggered. Default: `output`
`epmem --set cache-size [`*`some integer`*`]`	Number of memory pages used in the SQLite cache. Default: `10000`
`epmem --set graph-match [on,off]`	Graph matching enabled. Default: `on`
`epmem --set lazy-commit [on,off]`	Delay writing semantic store changes to file until agent exists. Default: `off`

WM Link Schema (see manual page 155):

Command Type	Syntax	Description
Retrieve	`<s> ^epmem.command.retrieve <time>`	Retrieve the episode captured at the given time.
Query	`<s> ^epmem.command.query <q>`	The query pattern to search for in an episode
	`^epmem.command.neg-query <nq>`	Avoid retrieving an episode that matches the given neg-query.
	`^epmem.command.prohibit <time>`	Requires the retrieved episode not be from the given time.
	`^epmem.command.before <time>`	Requires the retrieved episode come from before the given time.
	`^epmem.command.after <time>`	Requires the retrieved episode come from after the given time.

	`^epmem.result.success <q> <nq>`
	`^epmem.result.retrieved <ep>`
	`^epmem.result.memory-id <time>`
	`^epmem.result.present-id <present>`
	`^epmem.result.match-score <match>`
	`^epmem.result.cue-size <size>`
	`^epmem.result.normalized-match-score <norm>`
	`^epmem.result.match-cardinality <card>`
		If an episode matched the query (and optional neg-query):
		The `<ep>` ID points to the retrieved episode.
		`<time>` is an integer indicating the time of the retrieved episode.
		`<present>` is an integer indicating the current time.
		`<match>` indicates how closely the episode matches the query.
		`<size>` indicates how many leaf WMEs were in the query.
		`<norm>` is `<match>` divided by `<size>`.
		`<card>` is the count of leaf WMEs matched from `<q>`
		minus the count of WMEs matched from `<nq>`.

	`^epmem.result.graph-match <gmatch>`	If there is a result and the `graph-match` parameter is on: `<gmatch>` equals 1 if graph-match succeeded, and 0 otherwise.
	`^epmem.result.mapping <m-root>`	If the `graph-match` parameter is on and structural match passed: `<m-root>` provides a mapping from IDs in `<q>` to those in `<ep>`.
	`^epmem.result.failure <q> <nq>`	If no episode matched the given query (and optional neg-query).
Any	`<s> ^epmem.present-id <present>`	An integer indicating the current time, as used for episode IDs.
	`^epmem.command.next <n>`	Retrieve the episode that comes right after the most recently retrieved episode. The ID `<n>` can be any arbitrary identifier.
	`^epmem.command.previous <p>`	Retrieve the episode that comes right before the most recently retrieved episode. The ID `<p>` can be any arbitrary identifier.

RL

Common interface commands (see manual page 237):

Command	Description
`rl --set learning on`	Turns on RL
`rl --set learning-rate (0-1)`	Sets the learning rate (alpha). Default=0.3.
`rl --set discount-rate (0-1)`	Sets the discount rate (gamma). Default=0.9.
`rl --set decay-mode (normal, exponential, logarithmic, delta-bar-delta)`	How the learning rate changes over time. Default=`normal`.
`rl --set learning-policy (sarsa, q-learning, off-policy-gq-lambda, on-policy-gq-lambda)`	Value update policy. Default=`sarsa`.
`rl --set temporal-discount (on, off)`	Discount RL updates over gaps. Default=on.

WM link schema (see manual page 133):

Command Type	Syntax	Description
Reward	`<s> ^reward-link.reward.value <float>`	A reward signal added to the current state's total reward

RL rules (see manual page 131):

A production is a RL rule if and only if its left-hand side tests for a proposed operator, its right-hand side creates *only* a single numeric-indifferent preference, and it is not a template rule.

```
sp {this*is*an*rl*rule
    (state <s> ^name task-name
               ^operator <o> +)
    (<o> ^name move-left)
    -->
    (<s> ^operator <o> = 1.5) }
```

GP command (see manual page 203):

The gp command can be used to generate productions based on simple patterns. Patterns in gp are specified with sets of whitespace-separated values in square brackets. Every combination of values across all square-bracketed value lists will be generated. Values with whitespaces can be used if wrapped in pipes. Characters can also be escaped with a backslash (so string literals with embedded pipes and spaces outside of string literals are both possible).

```
gp {this*generates*rl*rules
    (state <s> ^name task-name
               ^operator <o> +
               ^row [ 1 2 3 ])
    (<o> ^name [ left right ])
    -->
    (<s> ^operator <o> = 1.0) }
```

Rule templates (see manual page 140):

Rule templates have variables that are filled in to generate RL rules as the agent encounters novel combinations of variable values. A rule template is valid if and only if it is marked with the :template flag and, in all other respects, adheres to the format of an RL rule. However, whereas an RL rule may only use constants as the numeric-indifference preference value, a rule template may use a variable.

```
sp {sample*rule*template
    :template
    (state <s> ^operator <o> +
               ^value <v>
    -->
    (<s> ^operator <o> = <v>) }
```

```
sp {rl*sample*rule*template*1
    (state <s> ^operator <o> +
               ^value 3.2
    -->
    (<s> ^operator <o> = 3.2) }
```

Preferences (see manual page 20)

Preference Name	Syntax	Description
Acceptable	`+`	An acceptable preference states that a value is a candidate for selection. All values, except those with require preferences, must have an acceptable preference to be selected. If there is only one value with an acceptable preference (and none with a require preference), that value will be selected as long as it does not also have a reject or a prohibit preference.
Reject	`-`	A reject preference states that the value is not a candidate for selection.
Better, Worse	`> value,` `< value`	A better or worse preference states, for the two values involved, that one value should not be selected if the other value is a candidate. Better and worse are simple inverses of each other, so that "A better than B" is equivalent to "B worse than A."
Best	`>`	A best preference states that the value may be better than any competing value (unless there are other competing values that are also "best"). If a value is best (and not rejected, prohibited, or worse than another), it will be selected over any other value that is not also best (or required). If two such values are best, then any remaining preferences for those candidates (worst, indifferent) will be examined to determine the selection. Note that if a value (that is not rejected or prohibited) is better than a best value, the better value will be selected.
Worst	`<`	A worst preference states that the value should be selected only if there are no alternatives. The semantics of the worst preference are similar to those for the best preference.
Unary Indifferent	`=`	When two or more competing values both have unary indifferent preferences, by default, Soar chooses randomly from among the alternatives. (The `decide indifferent-selection` function can be used to change this behavior as described on page 196 in the manual.)
Binary Indifferent	`= value`	A binary indifferent preference behaves like a unary indifferent preference, except that the operator value given this preference is only made indifferent to the operator value given as the argument.
Numeric-Indifferent	`= number`	A numeric-indifferent preference includes a unary indifferent preference and behaves in that manner when competing with another value having a unary indifferent preference. But when a set of competing operator values have numeric-indifferent preferences, the decision mechanism will choose an operator based on their numeric-indifferent values and the exploration policy. (The available exploration policies and how they calculate selection probability are detailed on page 196 in the manual.) When a single operator is given multiple numeric-indifferent preferences, they are either averaged or summed into a single value based on the setting of the `numeric-indifferent-mode` command (see page 196 in the manual). Numeric-indifferent preferences that are created by RL rules can be adjusted by the reinforcement learning mechanism.
Require	`!`	A require preference states that the value must be selected if the goal is to be achieved. A required value is preferred over all others. Only a single operator value should be given a require preference at a time.
Prohibit	`~`	A prohibit preference states that the value cannot be selected if the goal is to be achieved. If a value has a prohibit preference, it will not be selected for a value of an augmentation, independent of the other preferences.

Impasse State Structures (see manual page 84)

State no-change impasse

- When the proposal phase runs to quiescence and no operators are proposed

State structure	Value type	WME Count	Notes
^impasse no-change	string	1	
^choices none	string	1	
^attribute state	string	1	

Operator no-change impasse

- When the proposal phase runs to quiescence and either no new operator has been selected

State structure	Value type	WME Count	Notes
^impasse no-change	string	1	
^choices none	string	1	
^attribute operator	string	1	

Tie impasse

- When there is a collection of equally eligible operators competing for selection

State structure	Value type	WME Count	Notes
^impasse tie	string	1	
^choices multiple	string	1	
^attribute operator	string	1	
^item <o>	id	2+	A tied operator from the superstate
^item-count <count>	int	1	The number of tied operators
^non-numeric <o>	id	0+	The tied operators that do not have numeric-indifferent preferences
^non-numeric-count <count>	int	1	The number of tied operators that do not have numeric-indifferent preferences

Conflict impasse

- When two or more objects are better than each other and not dominated by a third operator

State structure	Value type	WME Count	Notes
^impasse conflict	string	1	
^choices multiple	string	1	
^attribute operator	string	1	
^item <o>	id	2+	A conflicting operator from the superstate
^item-count <count>	int	1	The number of conflicting operators
^non-numeric <o>	id	0+	The conflicting operators that don't have numeric-indifferent preferences
^non-numeric-count <count>	int	1	The number of conflicting operators that don't have numeric-indifferent preferences

Constraint-failure impasse

- When there are conflicting necessity preferences

State structure	Value type	WME Count	Notes
^impasse constraint-failure	string	1	
^choices <choices>	string	1	Either "constraint-failure" or "none"
^attribute operator	string	1	
^item <o>	id	2+	A conflicting operator from the superstate
^item-count <count>	int	1	The number of conflicting operators
^non-numeric <o>	id	0+	The conflicting operators that do not have numeric-indifferent preferences
^non-numeric-count <count>	int	1	The number of conflicting operators that do not have numeric-indifferent preferences

RHS Functions (see manual page 71)

Stopping and pausing Soar:

Function	Example	Description
halt	(halt)	Irreversibly terminates the running of a Soar program and returns to the user prompt. It should not be used if the agent is to be restarted. It will prevent agent output from being sent during the same cycle.
interrupt	(interrupt)	Causes Soar to stop at the end of the current phase and return to the user prompt. The run may be continued by issuing a run command from the user interface.
wait	(wait 1000)	Causes the current Soar thread to sleep for the given integer number of milliseconds.
write	(write |State ID is: | <s>)	Takes any number of arguments and writes them to standard output, concatenated and converted to string format. It does not automatically insert blanks, linefeeds, or carriage returns.
crlf	(write <x> (crlf) <y>)	Short for "carriage return, line feed", can be called only within write. It forces a new line at its position in the write action.
log	(log 3 |agent-logs channel 3 enabled|)	Equivalent to the write function, except that it specifies a "log channel" for output. The output will only show if that channel is active. The function takes two arguments. First is an integer corresponding to the channel level for output, second is the message to print.

Mathematical functions:

Function	Example	Description
+, -, *, /	(<s> ^sum (+ <x> <y>) ^product-sum (* (+ <v> <w>) (+ <x> <y>)) ^big-sum (+ <x> <y> <z> 402) ^neg-x (- <x>))	Prefix notation mathematical functions. They will take either integer or real-number arguments. The +, –, and * functions return an integer when all arguments are integers and otherwise return a real number, and / always returns a real number. These functions can each take any number of arguments and will return the result of sequentially operating on each argument. The – symbol is also a unary function which, given a single argument, returns the product of the argument and -1. The / symbol is also a unary function which, given a single argument, returns the reciprocal of the argument (1/x).
div, mod	(<s> ^quotient (div <x> <y>) ^remainder (mod <x> <y>))	Prefix notation binary mathematical functions (they each take two arguments). They take only integer arguments (using reals results in an error) and return an integer: div takes two integers and returns their integer quotient; mod returns their remainder.
abs, atan2, sqrt, sin, cos	(<s> ^abs-value (abs <x>) ^sqrt (sqrt <x>))	Prefix notation unary mathematical functions (they each take one argument). They will take either integer or real-number arguments. The abs function returns an integer when its argument is an integer and otherwise returns a real number, and the other functions always return a real number. The atan2 function returns, in radians, the arctangent of (first arg / second arg). The sin and cos functions take as arguments an angle in radians.
min, max	(<s> ^max (max <x> 3.14 <z>) ^min (min <a> <b> 42 <c>))	N-ary mathematical functions (they each take a list of symbols as arguments). They take either integer or real-number arguments and return a real-number value if any of their arguments are real-numbers. Otherwise they return integers.
int	(write (+ 2 (int sqrt(6))))	Converts a single symbol to an integer constant. This function expects either an integer constant, symbolic constant, or floating point constant. The symbolic constant must be a string which can be interpreted as a single integer. The floating point constant is truncated to only the integer portion.
float	(write (float (+ 2 3)))	Converts a single symbol to a floating point constant. This function expects either an integer constant, symbolic constant, or floating point constant. The symbolic constant must be a string which can be interpreted as a single floating point number.
ifeq	(write (ifeq <a> <b> equal not-equal))	Conditionally return a symbol. This function takes four arguments. It returns the third argument if the first two are equal and the fourth argument otherwise. Note that symbols of different types will always be considered unequal. For example, 1.0 and 1 will be unequal because the first is a float and the second is an integer.

Generating and manipulating symbols:

Function	Example	Description
capitalize-symbol	(capitalize-symbol foo)	Capitalizes the first character of a given string symbol.
compute-heading	(<s> ^heading (compute-heading 0 0.5 32.5 28))	Takes four real-valued arguments of the form (x1, y1, x2, y2), and returns the direction (in degrees) from (x1, y1) to (x2, y2), rounded to the nearest integer. The example to the left would result in a heading value of 48.
compute-range	(<s> ^distance (compute-range 0 0.5 32.5 28))	Takes four real-valued arguments of the form (x1, y1, x2, y2), and returns the distance from (x1, y1) to (x2, y2), rounded to the nearest integer. The example to the left would result in a distance value of 42.
concat	(<s> ^name (concat foo bar (+ 2 4)))	Given an arbitrary number of symbols, this function concatenates them together into a single constant symbol. The example to the left creates the WME (S1 ^name foobar6).
deep-copy	(<s> ^tree-copy (deep-copy <t>))	Returns a copy of the given symbol along with linked copies of all descendant symbols. In other terms, a full copy is made of the working memory subgraph that can be reached when starting from the given symbol. All copied identifiers are created as new IDs, and all copied values remain the same.
dc	(<s> ^dc-count (dc))	Returns the integer number of the current decision cycle.
@	(<s> ^lti-num (@ <l1>))	Returns an integer matching the LTI number of the given ID. If the given ID is not linked to an LTI, it returns nothing. (In the example to the left, if the function returns nothing, no WME would be created.)
link-stm-to-ltm	(link-stm-to-ltm <l1> 42)	Takes two arguments. It links the first given symbol to the LTI indicated by the second integer value.
make-constant-symbol	(<s> ^new-symbol (make-constant-symbol))	Returns a new constant symbol guaranteed to be different from all symbols currently present in the system. With no arguments, it returns a symbol whose name starts with "constant", such as "constant1". With one or more arguments, it takes those argument symbols, concatenates them, and uses that as the prefix for the new symbol. (It may also append a number to the resulting symbol, if a symbol with that prefix as its name already exists.)
rand-float	(<s> ^fate (rand-float 1000))	If no argument (or a negative argument) is given, it returns a random, real-valued number in the range [0.0, 1.0]. Otherwise, given a value *n*, it returns a number in the range [0.0, *n*].
rand-int	(<s> ^fate (rand-int 1000))	If no argument (or a negative argument) is given, it returns a random integer number in the range $[-2^{31},2^{31}]$. Otherwise, given a value *n*, it returns a number in the range [0, *n*].
round-off	(<s> ^pie (round-off <pi> 0.1))	Returns the first given value rounded to the nearest multiple of the second given value. Values must be integers or real-numbers. If pi = 3.142, the example to the left would create (S1 ^pie 3.1).
round-off-heading	(<s> ^true-heading (round-off-heading <dir> 0.5))	The same as round-off, but additionally shifts the returned value by multiples of 360 such that $-360 \leq$ value ≤ 360.
size	(<s> ^augs (size <n>))	Returns an integer symbol whose value is the count of WME augmentations on a given ID argument. Providing a non-ID argument results in an error.
strlen	(<s> ^message-len (strlen <m>))	Returns an integer count of the characters in the given string symbol.
timestamp	(write (timestamp))	Returns a symbol whose print name is a representation of the current date and time, such as "8/1/96-15:22:49".
trim	(<s> ^trimmed (trim <m>))	Takes a single string symbol argument and returns the same string with leading and trailing whitespace removed.

Interface commands:

Function	Example	Description
exec	(exec MakeANote <o> 1)	Used to call user-defined registered functions. Arguments are concatenated without spaces.
cmd	(write (cmd print -d 2 <s>))	Used to call built-in Soar commands. Spaces are inserted between concatenated arguments.

Controlling chunking:

Function	Example	Description
dont-learn	(dont-learn <s>)	When chunking is set to unflagged, by default chunks can be formed in all states; the dont-learn RHS action will cause chunking to be turned off for the specified state.
force-learn	(force-learn <s>)	When learning is set to flagged, by default chunks are not formed in any state; the force-learn RHS action will cause chunking to be turned on for the specified state.

Common Soar Terminal Commands (see manual page 184 ff.)

Command	Common Aliases	Description
soar init	init, is	Re-initializes the current agent. (No productions are removed.)
soar stop	stop, ss, interrupt	Stops any running Soar agents on this kernel and returns control to the user.
soar wait-snc on	waitsnc on	Wait instead of state no-change impasse.
run [*N*]	step [*N*], d [*N*]	Run for *N* decision cycles. *N* defaults to 1.
run -e [*N*]	e [N]	Run for *N* elaboration cycles. *N* defaults to 1.
run -o [*N*]		Run until the N^{th} time the agent sends ouput. *N* defaults to 1.
exit	stop	Terminates Soar and exits the kernel.
decide indifferent-selection (-b, --boltzmann, -g, --epsilon-greedy, -x, --softmax, -f, --first, -l, --last)	inds (-b,-g,-x,-f, -l)	Set how Soar selects among indifferent operators. (Default: Softmax) Boltzmann: Tempered softmax (uses temperature) Epsilon-greedy: Tempered greedy (uses epsilon) Softmax: Random, biased by numeric-indifferent value First: Deterministic; select first indifferent preference Last: Deterministic; select last indifferent preference
decide indifferent-selection (-e, --epsilon, -t, --temperature)	inds (-e, -t)	Sets exploration policy parameters. Epsilon is for Epsilon-greedy. Range: [0,1]. Default=0.1. Temperature is for Boltzmann. Range: (0,inf). Default=25.
decide set-random-seed [*X*]	srand [*X*]	Set the integer seed for the random number generator. Can be used to reproduce randomized decision behavior.
print (-f, --full) (-c, --chunks, -j, --justifications, -r, --rl)	p (-f) (-c,-j,-r)	Prints all loaded production of the specified types. (See pg 216 for other print types.) Omit the "--full" option to print just the production names.
print <ID> [-d *N*]	p <ID> -d *N*	Prints the WME under the given ID to the given integer depth *N*.
print @	p @	Prints the contents of SMEM
preferences -N [id [attr]]	pref -N [id [attr]]	Prints details about what past rule firings are keeping the given ID (or named attributes under that ID) in WM.
chunk always	learn always	Turns on chunking
chunk singleton <type> <attr> <type>		Tells chunking to combine instances of the given WME patterns into one WME pattern if duplicates appear in a generated chunk. Valid <types> are "state", "identifier", "operator", and "constant"
trace (-l, --level) (0-5)	trace (0-5) watch (0-5)	Set a specific trace level using an integer 0 to 5. Default=1.
trace (-L, --learning) (noprint, print, fullprint)	trace -L (0, 1, 2) watch -L (0, 1, 2)	Controls the printing of chunks/justifications as they are created. Default=noprint.
output enabled (on, off)		Globally toggle all output. Default=on.
output console (on, off)		Send output to std::out. Default=off.
output callbacks (on, off)		Send output to std print callback. Default=on.
output log [-A, --append] <filename>	clog [-A, --append] <filename>	Log all output to file.
output log [-c, --close]	clog [-c, --close]	Close the output log file.
explain all (on, off)		Record both justifications and chunks. Default=off.
explain list-chunks		List all chunks learned.
explain chunk [<chunk name> | <chunk id>]	c [<name> | <id>]	Start explainer discussing rule
explain instantiation <inst>	i <inst>	Explain the rule instantiation with the given ID.
explain identity	ei	Display identity to identity set mappings
explain explanation-trace	et	Switch explainer to explanation trace
explain wm-trace	wt	Switch explainer to WM trace
visualize [wm, smem, epmem] [<id>] [<depth>]	vis […]	Visualize from memory system
visualize [identity_graph, ebc_analysis]	vis [id, ebc]	Visualize explainer analysis
visualize color-identities (on, off)		Colorize identities in visualizations. Default=off.
visualize file-name <name>		The name of the file to generate. Default=soar_viz.
visualize image-type <ext>		Image file type that will be generated. Default=svg. See your Graphviz or DOT documentation for legal types.
stats (-s, --system)	st, stats	Report the system (agent) timing stats. (Default)
stats (-t, --track)		Begin tracking per-cycle max stats. (Time recorded in ns.)
stats (-c, --cycle)		Print collected per-cycle max stats saved by --track.
stats (-C, --cycle-csv)		Print collected per-cycle max stats in csv form.
load file <filename> (-a, --all, -v, --verbose)	source <filename> (-a,-v)	Load and evaluate the contents of a file. Use --all to print separate summaries for each file. Use --verbose to print names of any overwritten rules.
save agent <filename>		Writes procedural + semantic memory to file plus common settings.