Run Time Options for PAN

-A	suppress the reporting of assertion violations (see also **-E**)
-a	find acceptance cycles (available if compiled *without* **-DNP**)
-b	bounded search mode, makes it an error to exceed the search depth, triggering and error trail
-cN	stop at *N* th error (defaults to first error if *N* is absent)
-d	print state tables and stop
-E	suppress the reporting of invalid endstate errors (see also **-A**)
-e	create trails for all errors encountered (default is first one only)
-f	add weak fairness (to **-a** or **-l**)
-hN	choose another hash-function, with *N*: 1..32 (defaults to 1)
-i	search for shortest path to error (causes an increase of complexity)
-l	like **-i**, but approximate and faster
-J	reverse the evaluation order of nested unless statements (to conform to the one used in Java)
-kN	set the number of hashfunctions used in bitstate hashing mode to *N* (requires compilation with **-DBITSTATE**) The default is k=2. This option was introduced in version 4.2.0.
-l	find non-progress cycles (requires compilation with **-DNP**)
-mN	set max search depth to *N* steps (default *N*=10000)
-n	no listing of unreached states at the end of the run
-q	require empty channels in valid endstates
-r, -P, -C	play back error trail with embedded C code statements
-s	use 1-bit hashing (default is 2-bit hashing, assumes compilation **-DBITSTATE**). In version 4.2.0 and later, the option -s is equivalent to -k1.
-V	print *Spin* version number and stop
-wN	use ahashtable of 2^*N* entries(defaults to **-w***18*)

Compile Time Options for PAN

Directives Supported by Xspin

BITSTATE	use supertrace/bitstate instead of exhaustive exploration
MEMCNT=N	set upperbound to the amount of memory that can be allocated usage, e.g.: -DMEMCNT=20 for a maximum of 2^20 bytes
MEMLIM=N	set upperbound to the true number of Megabytes that can be allocated; usage, e.g.: -DMEMLIM=200 for a maximum of 200 Megabytes (meant to be a simple alternative to MEMCNT)
NOCLAIM	exclude the never claim from the verification, if present
NOFAIR	disable the code for weak-fairness (is faster)
NOREDUCE	disables the partial order reduction algorithm
NP	enable non-progress cycle detection (option -l), replacing option -a for acceptance cycle detection
PEG	add complexity profiling (transition counts)
SAFETY	optimize for the case where no cycle detection is needed (faster, uses less memory, disables both -l and -a)
VAR_RANGES	compute the effective value range of variables (restricted to the interval 0..255)
CHECK	generate debugging information (see also DEBUG)

Directives Related to Partial Order Reduction

CTL	allow only those reductions that are consistent with branching time logics like CTL (i.e., the persistent set contains either one or all transitions)
GLOB_ALPHA	consider process death a global action (for compatibility with versions of Spin between 2.8.5 and 2.9.7)
NIBIS	apply a small optimization of partial order reduction (sometimes faster, sometimes not...)
NOREDUCE	disables the partial order reduction algorithm
XUSAFE	disable validity checks of x[rs] assertions (faster, and sometimes useful if the check is too strict, e.g. when channels are passed around as process parameters)

Directives to Increase Speed

NOBOUNDCHECK	don't check array bound violations (faster)
NOCOMP	don't compress states with fullstate storage (faster, but not compatible with liveness unless -DBITSTATE)
NOFAIR	disable the code for weak-fairness (is faster)
NOSTUTTER	disable stuttering rules (warning: changes semantics) stuttering rules are the standard way to extend a finite execution sequence into and infinite one, to allow for a consistent interpretation of Büchi acceptance rules
SAFETY	optimize for the case where no cycle detection is needed (faster, uses less memory, disables both -l and -a)

Directives to Reduce Memory Use

BITSTATE	use supertrace/bitstate instead of exhaustive exploration
HC	a state vector compression mode; collapses state vector sizes down to 32+16 bits and stores them in conventional hash-table (a version of Wolper's hash-compact method -- new in version 3.2.2.) Variations: HC0, HC1, HC2, HC3 for 32, 40, 48, or 56 bits respectively. The default is equivalent to HC2.
COLLAPSE	a state vector compression mode; collapses state vector sizes by up to 80% to 90% (see Spin97 workshop paper) variations: add -DSEPQS or -DJOINPROCS (off by default)
MA=N	use a minimized DFA encoding for the state space, similar to a BDD, assuming a maximum of N bytes in the state-vector (this can be combined with -DCOLLAPSE for greater effect in cases when the original state vector is long)
MEMCNT=N	set upperbound to the amount of memory that can be allocated usage, e.g.: -DMEMCNT=20 for a maximum of 2^20 bytes
MEMLIM=N	set upperbound to the true number of Megabytes that can be allocated; usage, e.g.: -DMEMLIM=200 for a maximum of 200 Megabytes (meant to be a simple alternative to MEMCNT)
SC	enables stack cycling. this will swap parts of a very long search stack to a diskfile during verifications. the runtime flag -m for setting the size of the search stack still remains, but now sets the size of the part of the stack that remains in core. it is meant for rare applications where the search stack is many millions of states deep and eats up the majority of the memory requirements.

Directives Reserved for Use When Prompted by PAN

NFAIR=N	allocates memory for enforcing weak fairness usage, e.g.: -DNFAIR=3 (default is 2)
VECTORSZ=N	allocates memory (in bytes) for state vector usage, e.g.: -DVECTORSZ=2048 (default is 1024)

Directives for Debugging PAN Verifiers

VERBOSE	adds elaborate debugging printouts
CHECK	more frugal debugging printouts
SVDUMP	if defined, adds an option -pN to the runtime verifiers to produce a file sv_dump at the end of the run, with a binary representation of all states, using a fixed size of N bytes per state. (see also SDUMP below)
SDUMP	if used in addition to CHECK: adds ascii dumps of state vectors to verbose output (i.e., an ascii version of SVDUMP)

Directives for Experimental Use

BCOMP	when in BITSTATE mode, this computes hash functions over the compressed state-vector (compressed with byte-masking) in some cases, this can improve the coverage
COVEST	no longer supported, see NOCOVEST
HYBRID_HASH	no longer supported
LC	to be used in combination with BITSTATE hashing only. it is automatically enabled when -DSC is used in BITSTATE mode. LC forces the use of hashcompact compression for stackstates (instead of the dedault which is full-state storage for states while they are on the search stack, even in bitstate mode). it slows down the search, but can save memory. it uses 4 bytes per state (giving very low probability of collision).
NOCOVEST	omits the coverage estimate that is generated at the end of BITSTATE runs.
NOVSZ	risky - removes 4 bytes from state vector - its length field. in most cases this is redundant - so when memory is tight in fullstate storage, try this mode. if the number of states stored changes when -DNOVSZ is used, the information wasn't redundant... (safety checks will still be valid, but liveness checks may then fail) NOVSZ cannot be combined with COLLAPSE
PRINTF	enables printfs during verification runs (Version 2.8 and later -- earlier versions always left these enabled)
RANDSTORE	when in BITSTATE mode, use for instance -DRANDSTORE=33 to reduce the probability of storing the bits in the hasharray to 33%. the value assigned must be between 0 and 99 low values increase the amount of work done (time complexity) and increase the effective coverage for large state spaces. most useful in sequential bitstate hashing runs to improve the accumulative coverage of all runs significantly
REACH	guarantee absence of errors within the -m depth-limit (described in more detail in Newsletter 4 and in the V2.Updates notes for Version 2.2.)
W_XPT=N	in combination with MA, write checkpoint files every multiple of N states stored
R_XPT	in combination with MA, restart a verification run from the last checkpoint file written, can be combined with W_XPT

Compile Time Options for SPIN

NXT	if defined, the NEXT operator X can be used in LTL formulae; risky, not compatible with partial order reductions
PC	required when compiling Spin on a PC
PRINTF	if defined, printf statements in the model are enabled during the verification process (not recommended)
SOLARIS	required when compiling Spin on a Solaris system

Run Time Options for SPIN

Simulation

-B	Suppresses the verbose printout at the end of a simulation run (giving process states etc.).
-b	Suppresses the execution of printf statements within the model (see also -T).
-c	Produce an ASCII approximation of a message sequence chart for a random or guided (when combined with -t) simulation run. See also option -M.
-g	Shows at each time step the current value of global variables (see also -w).
-i	Perform an interactive simulation, prompting the user at every execution step that requires a nondeterministic choice to be made. The simulation proceeds without user intervention when execution is deterministic.
-jN	Skip the first N steps of a random or guided simulation. (See also option -uN.)
-l	In combination with option -p, shows the current value of local variables of the process (see also -w).
-M	Produce a message sequence chart in Postscript form for a random simulation or a guided simulation (when combined with -t), for the model in file , and write the result into file.ps . See also option -c.
-nN	Set the seed for a random simulation to the integer value N. There is no space between the -n and the integer N. -p, shows the current value of local variables of the process.
-p	Shows at each simulation step which process changed state, and what source statement was executed.
-qN	In columnated output (option -c) and elsewhere, suppress the printing of output for send or receive operations on the channel numbered N.
-r	Shows all message-receive events, giving the name and number of the receiving process and the corresponding the source line number. For each message parameter, show the message type and the message channel number and name.
-s	Shows all message-send events.
-T	Suppress the default indentation of output from print statements. By default the output from process i is indented by i spaces. See also option -b.
-t[N]	Perform a guided simulation, following the error trail that was produces by an earlier verification run, see the online manuals for the details on verification. If an optional number is attached (no space between the number and the -t) the error trail with that sequence number is opened, instead of the default trail, without sequence number.
-uN	Stop a random or guided simulation after the first N steps. (See also option -jN.)
-v	Verbose mode, adds some more detail, and generates more hints and warnings about the model.
-w	Even more verbose output with options -l and -g (e.g., prints all variable values, not just those that change).

Verification Generation

-a	Generate a verifier (model checker) for the specification. The output is written into a set of C files, named pan.[cbhmt] that can be compiled, (e.g., cc pan.c) to produce an executable verifier. The online Spin manuals (see below) contain the details on compilation and use of the verifiers.
-A	Perform property-based slicing, warning the user of all statements and data objects that are likely to be redundant for the stated properties (i.e., in assertions and never claims).
-d	Produce symbol table information for the model specified in file . For each Promela object this information includes the type, name and number of elements (if declared as an array), the initial value (if a data object) or size (if a message channel), the scope (global or local), and whether the object is declared as a variable or as a parameter. For message channels, the data types of the message fields are listed. For structure variables, the 3rd field defines the name of the structure declaration that contains the variable.
-F	file This behaves identical to option -f but will read the formula from the file instead of from the command line. The file should contain the formula as the first line. Any text that follows this first line is ignored, so it can be used to store comments or annotation on the formula. (On some systems the quoting conventions of the shell complicate the use of option -f. Option -F is meant to solve those problems.)
-f	LTL Translate the LTL formula LTL into a never claim. This option reads a formula in LTL syntax from the second argument and translates it into Promela syntax (a never claim, qhich is Promela's equivalent of a Buchi Automaton). The LTL operators are written: [] (always), <> (eventually), and U (strong until). There is no X (next) operator, to secure compatibility with the partial order reduction rules that are applied during the verification process. If the formula contains spaces, it should be quoted to form a single argument to the Spin command. As the name suggests, a Spin never claim is used to specify behavior than is required to be impossible, i.e., behavior that would violate a user-specified property. This means that to check for compliance with an LTL formula, the formula must be negated explicitly before it is converted into a never claim. Negating an LTL formula is easy: just place the formula "f" in parentheses and negate it: "!(f)".
-J	Reverse the evaluation order of nested 'unless' statements (so that it conforms to the evaluation order of nested 'catch' statements in Java).
-m	Changes the semantics of send events. Ordinarily, a send action will be (blocked) if the target message buffer is full. With this option a message sent to a full buffer is lost.
-V	Prints the Spin version number and exits.

Source: http://spinroot.com/spin/Man/index.html *Prepared by A. Burak Gurdag on 15.03.2007*