Verification Options

NAME

pan - *Spin* generated source for a model-specific verifier.

DESCRIPTION

Overview of options that are available with the verifiers generated by *Spin* with *Spin*'s run-time option **-a**. There are two groups of options: those that are available after the verifier source in pan.c has been compiled, and those that are available at compilation time. The reason for the split is that knowledge of some options at compile time can be used to produce a more efficient verification system. Attached is also a brief explanation of the numbers that are printed by the verifiers at the end of a run.

See V5_Readme for options specific to multi-core verifications with version 5.0 and later.

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**
  reserved
- **-b**
  bounded search mode, makes it an error to exceed the search depth, triggering and error trail
- **-C**
  for models with embedded C code, reproduce error trail in columnated format
- **-c***N*
  stop at *N*th error (defaults to first error if *N* is absent)
- **-d**
  print state tables and stop (-d -d or -d -d -d will print versions of the state tables before additional optimizations are applied)
- **-E**
  suppress the reporting of invalid endstate errors (see also **-A**)
- **-e**
  create trails for all errors encountered (default is first one only)
- **-F***filename*
  when compiled with -DSC, names the file to be used for the stack data
- **-f**
  add weak fairness (to **-a** or **-l**)
- **-g**
  for models with embedded C code, reproduce error trail with msc gui support
- **-h***N*
  choose another hash-function, with *N*: 1..32 (defaults to 1)
- **-I**
  like **-i**, but approximate and faster
- **-i**
  search for shortest path to error (causes an increase of complexity)
- **-J**
  reverse the evaluation order of nested unless statements (to conform to the one used in Java)
- **-k***N*
  set the number of hashfunctions used in bitstate hashing mode to *N* (bitstate mode) The default is k=2. This option was introduced in version 4.2.0.
- **-L***N* when compiled -DSCHED, sets a restriction on the max nr of context switches to *N* (default 10). Introduced in version 5.1.5.
- **-l**
  find non-progress cycles (requires compilation with **-DNP**)
- **-M***N*
  use N Megabytes for bitstate hash array (bitstate mode)
- **-G***N*
  use N Gigabytes for bitstate hash array (bitstate mode)
- **-m***N*
  set max search depth to *N* steps (default *N*=10000)
- **-n**
  no listing of unreached states at the end of the run
- **-N** *n*
  if more than one LTL property or never claim is defined, use the n-th one. Instead of *n* also the name of the claim or the LTL property can be used. If only one claim/property appears, or if none are used, then this option is not available.
- **-P***N*
  for models with embedded C code, reproduce trail, but print only steps from the process with pid *N*
- **-Q***N*
  set time-limit on execution of N minutes (not in multicore mode)
- **-q**
  require empty channels in valid endstates
- **-RS***N*
  Use *N* as a seed for the random number generator.
  Can be used in combination with compilation directives **-DT_RAND** and **-DP_RAND** (defined under experimental options below) to randomize the search process. *N* can be any non-negative integer value.
- **-r**, **-P**, **-C**
  for models with embedded C code, play back error trail (optionally followed by the name of a trailfile, as the next argument)
- **-r***N*
  play back error trail numbered N
- **-r** *filename*
  play back error trail stored in filename (works also with options **-C**, **-P***N*, **-g** and **-S**).
- **-S**
  for models with embedded C code, replay in silent mode, printing only the output from the user-defined printf statements in the model
- **-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.
- **-T**
  create trail files in read-only mode (see also -x)
- **-t***suff*
  instead of .trail use .suf on trailfiles
- **-U**
  reserved
- **-V**
  prints the *Spin* version number, and shows how the pan.c was compiled, then stops.
- **-v**
  verbose mode
- **-W**
  reserved
- **-w***N*
  use a hashtable of 2^*N* entries(defaults to **-w***23* in bitstate mode and **-w***19* in exhaustive search mode)
- **-X**
  print all stderr output onto stdout instead
- **-x**
  do not overwrite an existing trail file
- **-Y** and **-Z**
  reserved
- **-z***N* [Version 5 and later]
  Set handoff depth for multi-core search to N (default is 20).

B Compile Time Options for *Spin* and *Pan*

The directives are grouped in eight sets, depending on their main purpose.

There are four directives that can be used for compiling the *Spin* sources itself. These should never be needed by *Spin* users, only (once) by someone recompiling and installing *Spin* from its sources.

Directives for Compiling *Spin* itself

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

The next tables give optional directives for compiling the verifiers that are generated by *Spin*. Traditionally these are stored in a file named pan.c (with a number of dependent files). Usage of the directives below is always optional, and typically of the form:

```
$ spin -a spec
$ cc -o pan -DNOBOUNDCHECK pan.c
```

Each directive modifies the default behavior of the verifier to achieve a specific effect noted in the tables below.

Directives Supported by *Xspin*

BITSTATE	use supertrace/bitstate instead of exhaustive exploration
MEMCNT=N	sets upperbound to the amount of memory that can be allocated usage, e.g.: -DMEMCNT=20 for a maximum of 2^20 bytes. Use of MEMLIM is preferred.
MEMLIM=N	sets 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)

Other Main Search and Compilation Modes

BFS	use breadth-first, instead of depth-first search
BFS_DISK	in BFS mode, store some of the data on disk
BFS_DSK_LIMIT=N	in BFS mode, max number of states stored per diskfile, default 1 million
BFS_LIMIT=N	in BFS mode, point beyond which states move to disk, default 100,0000
CYGWIN	compile pan.c for 32-bit cygwin
WIN32	compile pan.c for 32-bit Windows
WIN64	compile pan.c for 64-bit Windows

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 B\(u"chi acceptance rules
SAFETY	optimize for the case where no cycle detection is needed (faster, uses less memory, disables both -l and -a)
SFH version 5	faster verification of safety properties, sets also NOCOMP (faster, uses slightly more memory, disables both -l and -a)

Directives to Reduce Memory Use

BITSTATE	use supertrace/bitstate instead of exhaustive exploration
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)
FULL_TRAIL version 5	leaving this directive out significantly reduces memory in multi-core mode, but reduces error-trails to a suffix of the full trail only. adding it restores the capability to generate full error trails. Only relevant in mult-core verfications; no effect elsewhere.
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.
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.
SPACE	optimize for space not speed

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)
VMAX=N version 5	allocates memory (in bytes) for state vector queues in Multicore mode, e.g.: -DVMAX=500 (default is 256) (Can also safely be ignored -- but optimizing the values can increase the number of states that can be accomodated in the handoff queues)
PMAX=N version 5	allocates memory (in bytes) for process structures inside queues in Multicore mode, e.g.: -DPMAX=32 (default is 16) (Can also safely be ignored -- but optimizing the values can increase the number of states that can be accomodated in the handoff queues)
QMAX=N version 5	allocates memory (in bytes) for chan structures inside queues in Multicore mode, e.g.: -DQMAX=32 (default is 16) (Can also safely be ignored -- but optimizing the values can increase the number of states that can be accomodated in the handoff queues)

Directives For Debugging *Pan* Verifiers

CHECK	more frugal debugging printouts (see also -DVERBOSE)
SDUMP	if used in addition to CHECK: adds ascii dumps of state vectors to verbose output (i.e., an ascii version of SVDUMP)
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)
VERBOSE	adds elaborate debugging printouts (see also -DCHECK)

Directives For Experimental Use

ALIGNED	on some platforms, to avoid complaints from the runtime system about unaligned data access	
FREQ=N version 5	changes the frequency of snapshot updates during a run from once every 1000000 states to once every N states	
HASH32 version 5	force the use of 32-bit hash (on 64-bit machine)	
HASH64 version 5	force the use of 64-bit hash (on 32-bit machine)	
JOINPROCS	a variant of collapse	
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).	
NO_FAST_C version 5	supress loop unrolling in compress() function	
NO_RESIZE version 5	suppress automatic resize of hashtable (bitstate mode)	
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	
NSUCC version 5	computes the average fanout of states in the statespace and prints a summary at the end of the run	
ON_EXIT=fct()		optional function to call just before pan exits
P_RAND=N version 5.2.5	randomize order in which processes are scheduled, using (optional) seed N the seed can also be set (or changed) with *pan* runtime option **-RS***N* the value provided can be any non-negative integer (See also T_RAND)	
PRINTF	enables printfs during verification runs (Version 2.8 and later -- earlier versions always left these enabled)	
PUTPID version 5.1.5	include the process pid number in trailfile names	
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.)	
REVERSE version 5.1.4	reverse the order in which process interleavings are explored (see also T_RAND, P_RAND, SCHED, and T_REVERSE	
R_XPT	in combination with MA, restart a verification run from the last checkpoint file written, can be combined with W_XPT	
SCHED version 5.1.5	restrict the maximum number of process context switches (using pan runtime option -L)	
T_RAND=N version 5.2.5	randomize order in which transitions are explored, using (optional) seed N the seed can also be set (or changed) with *pan* runtime option **-RS***N* the value provided can be any non-negative integer (See also P_RAND)	
T_REVERSE version 5.1.5	reverse order in which transitions are explored (See also T_RAND, and REVERSE)	
W_XPT=N	in combination with MA, write checkpoint files every multiple of N states stored	
ZAPH	in bitstate mode, reset the hash array to empty each time it becomes half full	

C *Pan*'s Output Format

A typical printout of a verification run is as follows:

```
$ pan
(Spin Version 3.0 -- 21 July 1997)
        + Partial Order Reduction

Full statespace search for:
        never-claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid endstates       +

State-vector 32 byte, depth reached 13, errors: 0
      74 states, stored
      30 states, matched
     104 transitions (= stored+matched)
       1 atomic steps
hash conflicts: 2 (resolved)
(max size 2^18 states)

1.533   memory usage (Mbyte)

unreached in proctype ProcA
        line 7, state 8, "Gaap = 4"
        (1 of 13 states)
unreached in proctype :init:
        line 21, state 14, "Gaap = 3"
        line 21, state 14, "Gaap = 4"
        (1 of 19 states)
```

This is what each line in this listing means:

```
(Spin Version 3.0 -- 21 July 1997)
```

Identifies the version of *Spin* that generated the pan.c source from which this verifier was compiled.

```
+ Partial Order Reduction
```

The plus sign means that the default partial order reduction algorithm was used. A minus sign would indicate compilation for exhaustive, non-reduced, verification with option -DNOREDUCE .

```
Full statespace search for:
```

Indicates the type of search. The default is a full statespace search. Large models can also be verified with a Bitstate search, which is approximate.

```
never-claim             - (none specified)
```

The minus sign indicates that no **never** claim, or LTL fomrula was used for this run. If a **never** claim was part of the model, it could have been suppressed with the compiler directive -DNOCLAIM .

```
assertion violations    +
```

The plus indicates that the search checked for violations of user specified assertions, which is the default.

```
acceptance   cycles     - (not selected)
```

The minus indicates that the search did not check for the presence of acceptance or non-progress cycles. To do so would require a run-time option **-a** or compilation with -DNP combined with the run-time option **-l**.

```
invalid endstates       +
```

The plus indicates that a check for invalid endstates was done (i.e., for absence of deadlocks).

```
State-vector 32 byte, depth reached 13, errors: 0
```

The complete description of a global system state required 32 bytes of memory (per state). The longest depth-first search path contained 13 transitions from the root of the tree (i.e., from the initial system state). No errors were found in this search.

```
74 states, stored
```

A total of 74 unique global system states were stored in the statespace (each represented effectively by a vector of 32 bytes).

```
30 states, matched
```

In 30 cases did the search return to a previously visited state in the search tree.

```
104 transitions (= stored+matched)
```

A total of 104 transitions were explored in the search, which can serve as a statistic for the amount of work that has been performed to complete the verification.

```
1 atomic steps
```

One of the transitions was part of an atomic sequence, all others were outside atomic sequences.

```
hash conflicts: 2 (resolved)
```

In 2 cases the default hashing scheme (a weaker version than what is used in bitstate hashing) encountered a collision, and had to place the states into a linked list in the hash-table.

```
(max size 2^18 states)
```

The (perhaps default) argument that was specified for the size of the hash-table was 2^18; equivalent to a run-time option **-w18**.

```
1.533   memory usage (Mbyte)
```

Total memory usage was 1.533 Megabytes, including the stack, the hashtable, and all related data structures. Memory use would go down for smaller than the default choices for run-time options **-m** and **-w**, which could in this case, with only 74 reachable states of 32 bytes each, considerably reduce memory use.

```
unreached in proctype ProcA
        line 7, state 8, "Gaap = 4"
        (1 of 13 states)
unreached in proctype :init:
        line 21, state 14, "Gaap = 3"
        (1 of 19 states)
```

A listing of the state numbers and approximate line numbers for the basic statements in the specification that were not reached. Since this is a full statespace search that ran to completion this means that these transitions are effectively unreachable (dead code).