Liveness:

```
spin -a file
gcc -o pan pan.c
pan -a -f or ./pan -a -f
spin -t -p -l -g -r -s file
```

Spin arguments

`-a`	generate verifier and syntax check
`-i`	interactive simulation
`-I`	display Promela program after preprocessing
`-nN`	seed for random simulation
`-t`	guided simulation with trail
`-tN`	guided simulation with Nth trail
`-uN`	maximum number of steps is N
`-f`	translate an LTL formula into a never claim
`-F`	translate an LTL formula in a file into a never claim
`-N`	include never claim from a file
`-l`	display local variables
`-g`	display global variables
`-p`	display statements
`-r`	display receive events
`-s`	display send events

Compile arguments

`-DBFS`	breadth-first search
`-DNP`	enable detection of non-progress cycles
`-DSAFETY`	optimize for safety
`-DBITSTATE`	bitstate hashing
`-DCOLLAPSE`	collapse compression
`-DHC`	hash-compact compression
`-DMA=n`	minimized DFA with maximum n bytes
`-DMEMLIM=N`	use up to N megabytes of memory

Pan arguments

`-a`	find acceptance cycles
`-f`	weak fairness
`-l`	find non-progress cycles
`-cN`	stop after Nth error
`-c0`	report all errors
`-e`	create trails for all errors
`-i`	search for shortest path to error
`-I`	approximate search for shortest path to error
`-mN`	maximum search depth is N
`-wN`	2^N hash table entries
`-A`	suppress reporting of assertion violations
`-E`	suppress reporting of invalid end states

Caveats

- Expessions must be side-effect free.
- Local variable declarations always take effect at the beginning of a process.
- A `true` guard can always be selected; an `else` guard is selected only if all others are false.
- Macros and `inline` do *not* create a new scope.
- Place labels before an `if` or `do`, *not* before a guard.
- In an `if` or `do` statement, interleaving can occur between a guard and the following statement.
- Processes are activated and die in LIFO order.
- Atomic propositions in LTL formulas must be identifiers starting with lowerase letters and must be boolean variables or symbols for boolean-valued expressions.
- Arrays of `bit` or `bool` are stored in bytes.
- The type of a message field of a channel cannot be an array; it can be a `typedef` that contains an array.
- The functions `empty` and `full` cannot be negated.

References

- G. J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*, Addison-Wesley, 2004. `http://spinroot.com`.
- M. Ben-Ari. *Principles of the Spin Model Checker*, Springer, 2008. `http://www.springer.com/978-1-84628-769-5`.

Spin Reference Card

Mordechai (Moti) Ben-Ari

September 29, 2008

Datatypes

`bit` (1 bit)
`bool` (1 bit)
`byte` (8 bits unsigned)
`short` (16^* bits signed)
`int` (32^* bits signed)
`unsigned` ($\leq 32^*$ bits unsigned)
  * - for a 32-bit machine.
`pid`
`chan`
`mtype` = { name, name, ... } (8 bits)
`typedef` typename { sequence of declarations }

Declaration - type var [= initial value]
Default initial values are zero.
Array declaration - type var[N] [= initial value]
Array initial value assigned to all elements.

Operators (descending precedence)

```
()   []   .
!    ~    ++   --
*    /    %
+    -
<<   >>
<    <=   >    >=
==   !=
&
^
```

```
|
&&
||
( ... -> ... : ... ) conditional expression
=
```

Predefined
Constants - `true`, `false`
Variables (read-only except `_`):
- `_` - write-only hidden scratch variable
- `_nr_pr` - number of processes
- `_pid` - instantiation number of executing process
- `timeout` - no executable statements in the system?

Preprocessor
`#define` name (arguments) string
`#undef`, `#if`, `#ifdef`, `#ifndef`, `#else`, `#endif`
`#include` "file name"
`inline` name (arguments) { ... }

Statements
Assignment - var = expression, var++, var--
`assert`(expression)

`printf`, `printm` - print to standard output
- `%c` (character), `%d` (decimal), `%e` (`mtype`),
- `%o` (octal), `%u` (unsigned), `%x` (hex)

`scanf` - read from standard input in simulation mode

`skip` - no operation
`break` - exit from innermost `do` loop
`goto` - jump to label
Label prefixes with a special meaning:
- `accept` - accept cycle
- `end` - valid end state
- `progress` - non-progress cycle

`atomic` { ... } - execute without interleaving
`d_step` { ... } - execute deterministically (no jumping in or out; deterministic choice among true guards; only the first statement can block).

{ ... } `unless` { ... } - exception handling.

Guarded commands
`if` :: guard -> statements :: ... `fi`
`do` :: guard -> statements :: ... `od`
`else` guard - executed if all others are false.

Processes
Declaration - `proctype` procname (parameters) { ... }
Activate with prefixes - `active` or `active`[N]
Explicit process activation - `run` procname (arguments)
Initial process - `init` { ... }
Declaration suffixes:
- `priority` - set simulation priority
- `provided` (e) - executable only if expression e is true

Channels
`chan` ch = [capacity] `of` { type, type, ... }

ch ! args	send
ch !! args	sorted send

ch ? args	receive and remove if *first* message matches
ch ?? args	receive and remove if *any* message matches
ch ? <args>	receive if *first* message matches
ch ?? <args>	receive if *any* message matches
ch ? [args]	poll *first* message (side-effect free)
ch ?? [args]	poll *any* message (side-effect free)

Matching in a receive statement: constants and `mtype` symbols must match; variables are assigned the values in the message; `eval`(expression) forces a match with the current value of the expression.

`len`(ch) - number of messages in a channel
`empty`(ch) / `nempty`(ch) - is channel empty / not empty?
`full`(ch) / `nfull`(ch) - is channel full / not full?

Channel use assertions:
- `xr` ch - channel ch is receive-only in this process
- `xs` ch - channel ch is send-only in this process

Temporal logic

`!`	not
`&&`	and
`||`	or
`->`	implies
`<->`	equivalent to
`[]`	always
`<>`	eventually
`X`	next
`U`	strong until
`V`	dual of U defined as `pVq <-> !(!pU!q)`

Remote references
Test the control state or the value of a variable:
- process-name `@` label-name
- proctype-name [expression] `@` label-name
- process-name : label-name
- proctype-name [expression] : label-name

Never claim
`never` { ... }.
Predefined constructs that can only appear in a never claim:
- `_last` - last process to execute
- `enabled`(p) - is process enabled?
- `np_` - true if no process is at a progress label
- `pc_value`(p) - current control state of process
- remote references

See also `trace` and `notrace`.

Variable declaration prefixes
`hidden` - hide this variable from the system state
`local` - a global variable is accessed only by one process
`show` - track variable in Xspin message sequence charts

Verification
Safety:
```
spin -a file
gcc -DSAFETY -o pan pan.c
pan or ./pan
spin -t -p -l -g -r -s file
```