NAME

runspin - script to automatically verify Promela models

SYNOPSIS

runspin [**options...**] *promela_file*

or more specifically:

runspin [**-a**] [**-f** *cfgs_file*] [**-h**] [**-n** *config_name*] [**-o** *out_file*] [**-p**] [**-s**] [**-v**] [**-V**]
[**-x** *"configuration"*] *promela_file*

DESCRIPTION

runspin is a bash script to automate the verification of a Promela model using the model checker SPIN.

runspin automates the complete verification of the Promela model *promela_file*. First, **runspin** invokes *spin* to generate the verification program pan.c. Then it invokes the *C* compiler to compile pan.c. Finally it runs the compiled *pan* verifier.

Apart from verifying the Promela model, **runspin** adds valuable extra information to *pan*'s verification report, e.g., the Promela file name, exact commands used, date of verification, unix time, etc.

For each of the three phases, **runspin** needs the necessary commands and options to execute the different programs (i.e., spin, (g)cc and pan). A complete set of commands and options for a single verification run is called a *configuration*. There are three ways to tell **runspin** what configuration to use:

-f *cfgs_file*
runspin retrieves the configuration from *cfgs_file*

-p
runspin retrieves the configuration from *promela_file*

-x "*configuration*"
the configuration is specified verbatim on the command line

A specific configuration can be named using the option **-n** *config_name*, where *config_name* is the name of the configuration. If none of the three configuration setting options (i.e., **-f**, **-p**, or **-x**) is used, **runspin** will perform a (standard) safety verification run.

OPTIONS

-a Verify **all** configurations in *promela_file*.
This option is only allowed in conjunction with the **-p** option.

-f *cfgs_file*
Get the configuration from *cfgs_file* (cfgs stands for configurations). See below for the expected format of this *cfgs_file*. If the **-n** option is also present, **runspin** will use this name of the configuration to search for in *cfgs_file*. Otherwise, *promela_file* will be used as the name of the configuration in *cmds_file*.

-h Print usage message.

-s No extra information: generate standard SPIN verification report; **runspin** will *not* add extra information regarding the verification.

-n *config_name*
Use *config_name* as the name for the configuration to be used.
This option is only allowed in conjunction with the options **-f** or **-p**.

-o *out_file*
Write the verification report to *out_file*.

-p Get the configuration from *promela_file*.
If the **-n** option is also present, **runspin** will use *config_name* as the name of the configuration to

search for in *promela_file*. Otherwise, the first configuration in *promela_file* will be used for the verification.

-v Print version number and exit.

-V Let **runspin** print verbose messages on what it is doing.

-x "*configuration*"
Forces **runspin** to use the commands in *configuration* for the verification of the Promela model. The double quotes " around the *configuration* are mandatory.

FORMAT FOR CONFIGURATIONS

The complete set of commands to verify a single Promela model is called a *configuration*. A configuration thus consists of a tuple of three command lines: the command line for *spin*, the command line for the *C* compiler and the command line for the *pan* verifier.

The command lines for the *C* compiler and *pan* verifier have to be complete: **runspin** will not add any parameters or information to the commands. The command line for *spin* is treated differently: **runspin** will add the (obligatory) *promela_file* as the last argument. This means that the typical command for the first command of the configuration is "spin -a". As an example of a configuration, consider the following three lines, where each line corresponds to a separate command line.

```
spin -a
gcc -o pan -DSAFETY pan.c
./pan -m10000 -c1 -w19
```

Below we explain how to format configurations for **runspin**.

command line: **-x** "*configuration*"

The configuration can be specified on the command line using the option **-x** "*configuration*". The string "*configuration*" specifies the three different command lines. Each command in the *configuration* string should be prefixed by "**%**". Consequently, a *configuration* contains exactly three **%** symbols. For example:

```
runspin -x "%spin -a %gcc -o pan pan.c %./pan -c1" foobar.prom
```

The configuration looks like a flattened version of the three command lines, where the **%** symbol acts as the shell prompt.

within the promela file: **-p**

Configurations can also be stored within the Promela file. runspin will search *promela_file* for the first line that contains the string "runspin". On this line, after the string "runspin", the configuration should follow the same format as for the **-x** option. That is, each command should be prefixed by "**%**". Moreover, the last command (i.e. the *pan* command) should be terminated with "**%**", if after the last command, the line contains additional non white space characters, that are not part of the command. For example, *promela_file* could contain the following comment:

```
/* runspin: %spin -a %gcc -o pan pan.c %./pan -c1 % */
```

The double colon is not necessary, but makes the line more readable. The fourth "**%**" is necessary to prevent that "*/" will be added to the *pan* command.

In addition, it is also possible to have several named configurations in a Promela file. The different configurations could then be selected using the **-t** option. To name a configuration, to the string "runspin" one should append an underscore (_) and the name , e.g., "runspin_safety". For example, *promela_file* could contain the following comments:

```
/* runspin_safety: %spin -a %gcc -o pan -DSAFETY pan.c %./pan    % */
/* runspin_ltl:    %spin -a %gcc -o pan -DNOFAIR pan.c %./pan -a % */
```

configurations file: **-f** *cfgs_file*

Configurations can also be stored in a separate file (*cfgs_file*). All configurations in a *cfgs_file* have to be named. A complete configuration in *cfgs_file* consists of four lines. The first line is the name of the configuration enclosed in square brackets: "[" and "]". Next, the three command lines of the configuration should follow.
It is common practice to separate configurations by empty lines, but this is not necessary.
A configuration file can have C++/Java style end-of-line comments: on a line, everything after "//" will be discarded.
An example of a fragment of a *cfgs_file* is the following:

```
[safety]            // standard safety run
spin -a
gcc -o pan -DSAFETY -DNOCLAIM pan.c
./pan -m10000 -w19 -c1

[foobar.prom]       // configuration to verify foobar.prom
spin -a
gcc -o pan -DSAFETY -DMEMLIM=1024 pan.c
./pan -m100000 -w24 -c1
```

EXAMPLES

`runspin foobar.prom`
Performs a standard verification run.

`runspin -p foobar.prom`
No configuration name is provided, so **runspin** will use the first configuration in ‘foobar.prom’ to verify the Promela model. This is probably the most common way to use **runspin**.

`runspin -p -n liveness foobar.prom`
Use the configuration named ‘liveness’ as defined in ‘foobar.prom’.

`runspin -f runspin.cfgs -n safety foobar.prom`
Use the configuration named ‘safety’ as defined in ‘runspin.cfgs’ to verify ‘foobar.prom’.

`runspin -f runspin.cfgs -o foobar.prom.out foobar.prom`
No configuration name is provided, so use the promela name ‘foobar.prom’ as the configuration name in ‘runspin.cfgs’. The verification report will be saved in ‘foobar.prom.out’.

`runspin -x "%spin -a %gcc pan.c %./a.out -c1" foo.prom`
The configuration is specified on the command line. Here we do not generate the verifier "pan", so we have to use "a.out", the default name for gcc generated executables. The **-x** option is typically used from within a shell script or makefile.

DEPENDENCIES

runspin relies on the presence of several (standard) UNIX utilities:
bash, spin, (g)cc, grep, sed, awk, time

FILES

runspin - the **runspin** shell script
runspin.cfgs - sample file with configurations

ADDITIONAL NOTES

The output of **runspin** (and SPIN) can be parsed by **parsepan**, a utility which has been developed hand-in-hand with **runspin**.

SEE ALSO

spin(1) -- SPIN website: http://www.spinroot.com
parsepan(1) -- parse pan verification reports

HISTORY

The original **runspin** script was developed in 2000/2001. Its purpose was to ease the execution of large batches of verification runs with SPIN. It has been used extensively for the experiments in [Ruys 2001]. The original version, however, was very limited in scope and functionality: it hardly supported the verification of a single Promela model.

VERSION

This documentation describes the first public version **runspin**: version 0.9 (19-Apr-2014).

AUTHOR

SPIN is developed by Gerard J. Holzmann (http://spinroot.com/).
The **runspin** script is written by Theo Ruys (theo dot ruys at gmail dot com).