NAME

parsepan - script to parse SPIN verification reports

SYNOPSIS

parsepan [**options...**] [*pan_verification_report(s)*]

or more specifically:

parsepan [**-h**, **--help**] [**-a**, **--allnames**] [**-d**, **--delimiter** *{c|t|=}*] [**-k**, **--keyfile** *key_file*] [**-l**, **--lines**] [**-q**, **--quotes**] [**-s**, **--summary**] [**-t**, **--header**] [**-v**, **--version**] [**-x**, **--debug**] [*pan_verification_report(s)*]

DESCRIPTION

parsepan is a script to parse the results of SPIN's pan verifier. **parsepan** writes the information as data values to the standard output. In its default setting, all data values found are written on a single line delimited by tabs. The format of the ouptut can be customized through several options. Apart from SPIN's standard verification report, **parsepan** recognizes the output added by the **runspin** script. If no input files are specified on the command-line, the input is read from the standard input.

parsepan makes it easy to collect specific information from SPIN's verification reports and to export a comma separated values (csv) file, which can be imported into analysis software such as a spreadsheet. In connection with runspin, **parsepan** can serve as unit test tool for Promela specifications.

To identify parts of the verification report, **parsepan** uses names for specific items in the report, e.g., state_vector, errors, atomic_steps, etc. A verification report can thus be seen as a list of *(key,value)* pairs, where *key* is the name of the item, and *value* is the specific value as found in the verification report.

When multiple verification reports are specified (or when the input is the standard input), **parsepan** will print the values of each report on a separate line. **parsepan** will use the first verification report as reference point: for all subsequent verification reports only the items as found on the first report will be included (and in the same order). For missing items, a dash "-" will be included. This means that all *columns* of the different lines (the rows) will correspond with the same *key.*

OPTIONS

-h, **--help**

Prints the synopsis of the script and the possibly options.

-a, **--allnames**

The script dumps all *key* names that are recognized by parsepan. This list can serve as a starting point to build a custom *keyfile* (see below), to specify the keys that one is interested in.

-d, **--delimiter** c | t | =

Specifies the delimiter between the data values. Only three possible delimiters are currently supported. The character *c* corresponds with the comma, the character *t* corresponds with the tab character, and the character = is simply the equal sign. By default, the delimiter is a tab.

-k , --keyfile *keyfile*

By default, parsepan will output *all* values that it finds in the verification report. One is usually only interested in a subset of these values. When using this option, parsepan will only report on the values of the keys that are in the *keyfile.* parsepan will respect the order of the keys in the *keyfile*, when printing the values.

-l, **--lines**

Instead of printing all values on a single line, this option will dump all *(key,value)* pairs to the standard output; each pair will be printed on a single line, and the *key* and *value* will be delimited by an equal sign (=).

-q, **--quotes**
This option will make sure that all values that are being printed will be enclosed in double quotes. This is useful when the output of parsepan (e.g., csv format) is used as input for a third party program (e.g., a spreadsheet) that requires quoted values.

-s, **--summary**
Prints a very concise summary of the verification report. It only prints the values for the following keys: *rs_promela_file, rs_config_name, errors, states_stored, state_vector.* This is useful when parsepan is used as unit test tool.

-t, **--header**
Includes all *key* names as a header row before the first line of values. This is useful when exporting the results to a third party program.

-v, **--version**
Prints the version number of parsepan and exits.

-x, **--debug**
The debug option will print all non-empty lines in the verification report that parsepan did not recognize. This option was useful in identifying all lines that can be printed by a pan verifier.

EXAMPLES

`runspin -p -n deadlock peterson.prom | parsepan --header --delimiter c`
The Promela model *peterson.prom* is verified according to configuration *deadlock*. The verification reports are fed to parsepan, which shows all data values on a single line, headed by the names of the keys. The (abbreviated) output would look like this:

```
rs_promela_file,rs_verification_date,rs_spin_version, ...
peterson.prom,16-Apr-2014 13:00:00,Spin Version 6.2.7 -- 2 March 2014, ...
```

`runspin -a -p peterson.prom | parsepan --summary`
First all configurations found in *peterson.prom* are verified. The output of these verification reports is read by parsepan, which then reports a summary of the verification reports:

```
peterson.prom   deadlock    0   32  20
peterson.prom   mutex       0   32  36
peterson.prom   progress    0   62  36
```

DEPENDENCIES

parsepan is a Python script and (apart from SPIN and optionally runspin) it only requires Python to be installed.

FILES

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

ADDITIONAL NOTES

To automate the verification process with SPIN there is also an utility called **runspin**, which has been developed hand-in-hand with **parsepan**. **parsepan** recognizes the information added by the **runspin** script.

SEE ALSO

spin(1) -- SPIN website: http://www.spinroot.com
runspin(1) -- automatically verify Promela models

HISTORY

parsepan is (loosely) based on the perl script **ppr** (parse pan reports) which was developed in 2000/2001. Its purpose was to batch process SPIN verification reports. **ppr** has been used extensively for the experiments in [Ruys 2001].

VERSION

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

AUTHOR

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