NAME

tgv – Test Generation from transitions systems using Verification techniques

SYNOPSIS

bcg_open [*bcg_opt*] *spec*[**.bcg**] [*cc_opt*] **tgv** [*tgv_opt*] *tp*[**.bcg**|**.aut**]

or:

exp.open [*exp_opt*] *spec*[**.exp**] [*cc_opt*] **tgv** [*tgv_opt*] *tp*[**.bcg**|**.aut**]

or:

fsp.open [*fsp_opt*] *spec*[**.lts**] [*cc_opt*] **tgv** [*tgv_opt*] *tp*[**.bcg**|**.aut**]

or:

lnt.open [*lnt_opt*] *spec*[**.lnt**] [*cc_opt*] **tgv** [*tgv_opt*] *tp*[**.bcg**|**.aut**]

or:

lotos.open [*lotos_opt*] *spec*[**.lotos**] [*cc_opt*] **tgv** [*tgv_opt*] *tp*[**.bcg**|**.aut**]

or:

seq.open [*seq_opt*] *spec*[**.seq**] [*cc_opt*] **tgv** [*tgv_opt*] *tp*[**.bcg**|**.aut**]

DESCRIPTION

TGV allows the generation of an abstract test case from a specification and a test purpose. The generation is done "on-the-fly" on the synchronous product of the specification with the test purpose. It is based on Tarjan's algorithm. During the depth-first search (DFS), TGV performs abstraction and determinisation of this product. The DFS stops when an accepting state of test purpose is reached. During the backtracking, TGV synthesizes the transitions of the test case.

The specification and the test purpose are Inputs/Outputs Labeled Transition Systems (IOLTS). The specification is either the BCG graph *spec***.bcg**, the composition expression *spec***.exp**, the FSP program *spec***.lts**, the LNT program *spec***.lnt**, the LOTOS program *spec***.lotos**, or the sequence file *spec***.seq**.

The test purpose is described either in a BCG file *tp***.bcg** or in an **AUT** file *tp***.aut**.

TGV uses the **bcg_io**(LOCAL) tool to transparently convert a test purpose in AUT format into the BCG format, and to transform the generated test case or complete test graph into the AUT format.

BCG FORMAT

See the **bcg**(LOCAL) manual page.

AUT FORMAT

See the **aut**(LOCAL) manual page.

TEST PURPOSES

A test purpose is an abstract description of a subset of the specification, allowing to choose behaviors to test, and consequently allowing to reduce the specification exploration. Final states of the test purpose graph are either accepting states (the purpose is reached) or refusing states (parts of the specification are rejected).

Accepting (respectively refusing) states must be given as loop-transitions with the predefined label **`ACCEPT`** or **`accept`** (respectively **`REFUSE`** or **`refuse`**). The purpose must own one accepting state at least.

All action names are regular expressions (according to the definition given in the manual page of the POSIX **regexp**(LOCAL)). The test purpose is written in accordance with the specification label names (before renaming and hidding). It can also have transitions labelled with invisible actions of the specification.

If the predefined label "*" (which means **`otherwise`**) is present on a transition leaving some state *s*, this label represents all other actions than those already present on the outgoing transitions of state *s*.

So, for an AUT test purpose, a transition has the following grammar:

```
<transition> ::=
       '(' <from_state> ',' <action> ',' <to_state> ')'   |
       '(' <state> ',' 'ACCEPT'|'accept' ',' <state> ')' |
       '(' <state> ',' 'REFUSE'|'refuse' ',' <state> ')' |
       '(' <from_state> ',' '*' ',' <to_state> ')'

  <action> ::= <UNIX_regexp>
             | '"' <UNIX_regexp> '"'
```

Note: if the extension of the test purpose filename is omitted (or is different from **.bcg** and **.aut**), the file is first searched with **.bcg** extension and then with **.aut** extension.

TEST CASES

If the test purpose is valid, i.e. allows to select sequences of the specification leading to an accepting state, TGV produces an LTS that is the description of the test graph in the BCG format (**.bcg**) or AUT (**.aut**) format.

Actions labels of this LTS are based on those of the specification, plus some predefined labels as **`LOCK, OUTPUTLOCK, DEADLOCK, LIVELOCK`** used in lock transitions. A tag (**`INPUT`** or **`OUTPUT`**), seen from the tester's viewpoint, is added to each label. In some transitions, a verdict (**`PASS`**) or **`PASS`** or (**`INCONCLUSIVE`**) or **`INCONCLUSIVE`** is also added to the label.

A **`PASS`** verdict on a transition means that in the state reached by this transition, the tester has detected no implementation error in the IUT.

An **`INCONCLUSIVE`** verdict is present on a transition corresponding to a possible output of the specification that leads to a sequence not satisfying the test purpose.

There is no **`FAIL`** verdict, as fail transitions are implicit (from each state, unrecognized actions lead to a **`FAIL`** state).

A verdict not enclosed in parentheses means that the tester has reached a stable state from which no output of the implementation under test (IUT) is expected.

OPTIONS

The options *bcg_opt*, if any, are passed to **bcg_lib**(LOCAL).

The options *exp_opt*, if any, are passed to **exp.open**(LOCAL).

The options *fsp_opt*, if any, are passed to **fsp.open**(LOCAL).

The options *lnt_opt*, if any, are passed to **lnt.open**(LOCAL).

The options *lotos_opt*, if any, are passed to **caesar**(LOCAL) and to **caesar.adt**(LOCAL).

The options *seq_opt*, if any, are passed to **seq.open**(LOCAL).

The options *cc_opt*, if any, are passed to **cc**(1).

The options *tgv_opt*, if any, are passed to the TGV program.

TGV OPTIONS

The following *tgv_opt* are currently available:

-verbose
Verbose mode: TGV displays information about the calculations going on. Not a default option.

-io *file*[**.io**]
Specify the inputs/outputs of the specification, according to *file*[**.io**] (in case of renaming with **-rename** option, the **.io** file must contain *renamed* labels).

-rename *file*[**.ren**|**.rename**]
Use the renaming rules defined in *file*[**.ren**|**.rename**] to rename the labels of the product SPEC x TP.

-hide *file*[**.hid**|**.hide**]
Use the hiding rules defined in *file*[**.hid**|**.hide**] to hide some labels of the product SPEC x TP.

-tpprior
Priority to actions of the test purpose. By default, the actions of the specification are prior.

-outprior
Priority to the outputs. By default, the inputs are prior.

-self
For each state *s* of the test purpose such that *s* does not have, for each action *a* of the specification, an outgoing transition labelled with *a*, and *s* does not have an outgoing transition labelled with the predefined label "*", silently add on state *s* a self-loop transition labelled with "*". By default, such a self-loop transition is added too, but a warning message is emitted for each such state *s*.

-hash *n*
Fix the table hash size to *n* (by default, 100000).

-depth *n*
Fix the maximum preamble-body depth search to *n*.

-output *file*[**.bcg**|**.aut**] [**-parse**|**-unparse**]
Specify the name *file*[**.bcg**|**.aut**] of the output file in which TGV will display the test case. If the filename extension is **.aut**, the result is an AUT file. Otherwise, the result file is a BCG file with the **.bcg** extension. By default, the result is printed on the BCG file *tgv_result***.bcg**.

If the produced test case is in BCG format, options **-parse** and **-unparse** can be specified to control label parsing (see the **bcg_write**(LOCAL) manual page for a technical discussion about label parsing). Option **-parse** enables label parsing and option **-unparse** disables label parsing. By default, label parsing is enabled.

-label
Display all fired labels. Notice that displaying all labels of the specification can be obtained by using this option with the following test purpose:

```
des (0, 3, 2)
    (0, *, 0)
    (0, "dummy:", 1)
    (1, ACCEPT, 1).
```

-keeplock
By default, TGV computes lock transitions (`LOCK`) and prints those that remain after conflicts have been resolved. Use **-keeplock** (without **-csg**) to keep all the computed locks (the produced test graph is no more completely controllable).

-unlock|**-difflock**
Avoid the printing of lock transitions with option **-unlock**. Make difference between lock tags (`OUTPUTLOCK, DEADLOCK, LIVELOCK`) with option **-difflock**.

-csg|**-unloop**

Compute the complete test graph (option **-csg**) or compute a controllable test case without loop (option **-unloop**). By default, TGV computes controllable test cases with loops.

-post
Search a postamble from pass and inconclusive states (i.e. a path to stable states in which no message from the IUT is expected).

-postdepth *n*
Fix the maximum postamble depth search to *n*.

-random
Compute a random test case. Not a default option.

-randomseed *n*
Use *n* as seed for random test-case generation.

-verif
Do not annotate transition labels with input or output, nor with verdicts, but generate separate, self-looping verdict transitions for each verdict state. This option is incompatible with options **`-timer`**, **`-difflock`**, and **`-unlock`**.

-timer
Produce a test case with test timers (TAC and TNOAC). TAC is started when the tester is waiting for an output from the IUT. TNOAC allows to detect a lock state of the IUT.

-monitor
Open a window for monitoring in real-time the generation of the test case or complete test graph. Not a default option.

-parse
Enable label parsing when reading files in the AUT format and/or writing files in the BCG format (see the **bcg_write**(LOCAL) manual page for a technical discussion about label parsing). Note that this also applies to the implicit conversions of command-line arguments in the AUT format. Default option.

-unparse
Disable label parsing when reading files in the AUT format and/or writing files in the BCG format (see the **bcg_write**(LOCAL) manual page for a technical discussion about label parsing). Note that this also applies to the implicit conversions of command-line arguments in the AUT format. Not a default option.

I/O FILE

It is recommended to write an **.io** file, to distinguish between inputs and outputs. The **.io** file describes a set of actions, according to the following grammar:

```
<file.io> ::= 'input' | 'output' \n <regexp-list>

    <regexp-list> ::= <regexp> \n <regexp-list>
                    | "<regexp>" \n <regexp-list>
                    | <empty>
```

Semantically, if the first line is equal to `input` (respectively `output`), the body of the file describes all the inputs actions (respectively all outputs actions).

Note: If option **-io** is not given, TGV uses by default the file **$CADP/src/tgv/default.io** the contents of which are:

```
input
        [^!]*[?].*
```

HIDE FILE

This is an optional input file. See the **caesar_hide_1**(LOCAL) manual page for the grammar description and examples.

RENAME FILE

This is an optional input file. See the **caesar_rename_1**(LOCAL) manual page for the grammar description and examples.

Note 1: renaming patterns are applied *before* hiding patterns. Renaming and hiding patterns are applied after the synchronous product SPECxTP.

Note 2: the **.io** file must be written in accordance with the **.ren** (or **.rename**) and **.hid** (or **.hide**) files (because the **.io** file is read after the renaming and hiding).

TGV MESSAGES

If TGV produces a test case, the execution finishes with the following message: "`File` *file* `produced`". Otherwise, the message "`No test case`" is printed: may be it is because the test purpose is not valid, or because of an error in the description of files **.io** or **.hid** or **.ren**

AUTHORS

Thierry Jéron (Thierry.Jeron@irisa.fr), Pierre Morel, and Séverine Simon. A few patches were brought by Wendelin Serwe and Hubert Garavel in December 2004. Since then, TGV has been ported to 64-bit architectures and regularly maintained by Wendelin Serwe and Hubert Garavel.

SEE ALSO

OPEN/CAESAR Reference Manual, **bcg**(LOCAL), **bcg_open**(LOCAL), **caesar**(LOCAL), **caesar.adt**(LOCAL), **exp**(LOCAL), **exp.open**(LOCAL), **fsp.open**(LOCAL), **lnt.open**(LOCAL), **lotos**(LOCAL), **lotos.open**(LOCAL), **seq**(LOCAL), **seq.open**(LOCAL)

BUGS

Please report any bug to cadp@inria.fr