NAME

exhibitor – search for execution sequences matching a given pattern

SYNOPSIS

bcg_open [*bcg_opt*] *spec*[**.bcg**] [*cc_opt*] **exhibitor** [*exhibitor_opt*]

or:

exp.open [*exp_opt*] *spec*[**.exp**] [*cc_opt*] **exhibitor** [*exhibitor_opt*]

or:

fsp.open [*fsp_opt*] *spec*[**.lts**] [*cc_opt*] **exhibitor** [*exhibitor_opt*]

or:

lnt.open [*lnt_opt*] *spec*[**.lnt**] [*cc_opt*] **exhibitor** [*exhibitor_opt*]

or:

lotos.open [*lotos_opt*] *spec*[**.lotos**] [*cc_opt*] **exhibitor** [*exhibitor_opt*]

or:

seq.open [*seq_opt*] *spec*[**.seq**] [*cc_opt*] **exhibitor** [*exhibitor_opt*]

DESCRIPTION

exhibitor takes two different inputs:

- A Labelled Transition System, expressed either as a BCG graph *spec***.bcg**, a composition expression *spec***.exp**, an FSP program *spec***.lts**, a LNT program *spec***.lnt**, a LOTOS program *spec***.lotos**, or a sequence file *spec***.seq**.

- A "pattern" (or "goal"), expressed in the full SEQ format described in the **seq**(LOCAL) manual page. By default, this pattern is read from the standard input (under UNIX, the input stream should end up with a 'control-D' character; under Windows, it should end with a 'control-Z' character). Alternately, the pattern can be stored in a file, e.g., *filename***.seq** (the "**.seq**" suffix is not mandatory, but strongly advisable), in which case this file can be passed to **exhibitor** (under UNIX, the stream redirection notation "**<** *filename***.seq**" can be used for this purpose).

exhibitor performs an on-the-fly search in the Labelled Transition System (LTS), looking for execution sequences (also called "diagnostic sequences") that start from the initial state and match the specified pattern.

exhibitor displays on the standard output, in the simple SEQ format, the diagnostic sequence(s) found, if any. If no diagnostic sequence has been found, **exhibitor** displays an empty result (this case is covered by the simple SEQ format).

If the input pattern is empty (which is allowed by the syntax of the full SEQ format), **exhibitor** stops immediately, as no diagnostic sequence can be found.

If the input pattern contains more than one **`sequence`** (see the **seq**(LOCAL) manual page for details), **exhibitor** only searches for a single one, according to the number supplied with the **-seqno** option.

In the CADP toolbox, the SEQ format is the standard format for diagnostic sequences. Many CADP tools take their inputs and/or deliver their outputs in this format. Unfortunately, such diagnostic sequences are not necessarily as short as possible and some information might have been lost (e.g., sequences of **`i`**-transitions

have been compacted or eliminated, the original gate names corresponding to **i**-transitions have vanished, etc.).

In many cases, **exhibitor** solves these problems by allowing to find the shortest sequence matching a given pattern and by providing the source-level information which has been lost. It is also useful for verification and test purpose, because it can search and display automatically an execution scenario defined by a given pattern.

SEARCH ALGORITHMS

exhibitor implements two different search algorithms, one based on a breadth-first search (BFS) and another one based on a depth-first search (DFS). Both algorithms have their own merits. Here is a brief comparison:

- The BFS algorithm always produces the shortest diagnostic sequence(s) (if any) that match the given pattern.

- The DFS algorithm does not necessarily produce the shortest diagnostic sequence (because in '∗'-groups, the states are visited only once). Moreover, as this algorithm is programmed using a recursive C function, **exhibitor** may abort with a core dump if the execution stack overflows. In such case, the maximal depth should be limited using the **-depth** option (see below).

Two small examples will illustrate the differences between both algorithms. Let S be the sequence **'<until> "exit"'**.

- Let's consider the LTS corresponding to the LOTOS behaviour:
  L1; L2; exit [] L3; exit

  The BFS algorithm will find the optimal solution:
  L3; exit
  in a few steps, whereas the DFS algorithm might very well explore entirely the longest path:
  L1; L2; exit
  before finding a better solution. Also, if there are ∗-s in the sequence to be searched and circuits in the LTS, the sequence found by the DFS algorithm might not be minimal.

- Let's consider the LTS corresponding to the LOTOS behaviour:
  L1; exit ||| L2; exit ||| ... ||| Ln; exit

  The DFS algorithm will directly find a solution in n+1 steps, whereas the BFS will have to explore and store all the 2^n possible interleavings of transitions **L1, ..., Ln** before finding a solution.

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 the C compiler.

The following options *exhibitor_options* are currently available:

-seqno *number*
Select the *number*-th sequence in the input stream as the pattern to be searched for. *number* should be a non-zero, positive integer. It should be less than or equal to the number of sequences contained in the input stream. By default, the first sequence will be selected.

-depth *depth*
Consider only execution sequences whose number of transitions is less than or equal to *depth* (where *depth* is greater than zero). Prune the exploration of the graph when the distance from the initial state becomes greater than *depth* transitions. By default (if this option is not present on the command-line) or if *depth* is equal to zero, all sequences will be considered.

-conflict
When the search is done, print information about those labels that created non-determinism conflicts, solved in a deterministic way (see the **seq**(LOCAL) manual page for details). Not a default option.

-case
Preserve case-sensitivity in the **`string`**s and **`regular_expression`**s specified in the input stream. Not a default option. By default, all lower-case letters contained in **`string`**s and **`regular_expression`**s are turned to upper case, except for the special gates **`"i"`** and **`"exit"`** (see the **seq**(LOCAL) manual page for details).

-bfs Use the breadth-first search algorithm. Default option.

-dfs Use the depth-first search algorithm. Not a default option.

-none
Don't print any diagnostic sequence. Not a default option.

-all Print all diagnostic sequences. With the BFS algorithm, all the printed sequences are minimal. Not a default option.

-first
Print the first diagnostic sequence encountered and stop just after. Not a default option for the DFS algorithm.

-decr
For the BFS algorithm: this option is identical to **-first**.

For the DFS algorithm: print only those diagnostic sequences which are shorter than the last diagnostic sequence previously displayed. Prune the exploration of the graph in order to find

diagnostic sequences of decreasing sizes. It is not guaranteed that the final sequence is minimal (some shorter sequence might exist, which can not be found by the DFS algorithm). Not a default option for the DFS algorithm.

by default

(in absence of **-none**, **-all**, **-first**, **-decr**) For the BFS algorithm: default is identical to **-first**.

For the DFS algorithm: print only the shortest diagnostic sequence obtained. Prune the exploration of the graph in order to find diagnostic sequences of decreasing sizes. It is not guaranteed that the final sequence is minimal (some shorter sequence might exist, which can not be found by the program).

EXIT STATUS

When the source is erroneous, error messages are issued. Exit status is 0 if everything is alright, 1 otherwise.

AUTHORS

Version 1 of **exhibitor** was developed by Hubert Garavel (INRIA Rhone-Alpes).

Version 2 of **exhibitor** was developed by Hubert Garavel and Xavier Etchevers, with the help of Radu Mateescu.

OPERANDS

spec.**bcg**	BCG graph (input)
spec.**exp**	network of communicating LTSs (input)
spec.**lts**	FSP specification (input)
spec.**lnt**	LNT specification (input)
spec.**lotos**	LOTOS specification (input)
spec.**seq**	sequence file (input)
(standard input)	pattern in SEQ format (input)

FILES

The binary code of **exhibitor** is available in $CADP/bin.‘arch‘/exhibitor.a

SEE ALSO

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

Additional information is available from the CADP Web page located at http://cadp.inria.fr

Directives for installation are given in files **$CADP/INSTALLATION_*.**

Recent changes and improvements to this software are reported and commented in file **$CADP/HISTORY.**

BUGS

Please report new bugs to Hubert.Garavel@inria.fr