NAME

terminator – deadlock detection

SYNOPSIS

bcg_open [*bcg_opt*] *spec*[**.bcg**] [*cc_opt*] **terminator** [*terminator_opt*] *terminator_param*

or:

exp.open [*exp_opt*] *spec*[**.exp**] [*cc_opt*] **terminator** [*terminator_opt*] *terminator_param*

or:

fsp.open [*fsp_opt*] *spec*[**.lts**] [*cc_opt*] **terminator** [*terminator_opt*] *terminator_param*

or:

lnt.open [*lnt_opt*] *spec*[**.lnt**] [*cc_opt*] **terminator** [*terminator_opt*] *terminator_param*

or:

lotos.open [*lotos_opt*] *spec*[**.lotos**] [*cc_opt*] **terminator** [*terminator_opt*] *terminator_param*

or:

seq.open [*seq_opt*] *spec*[**.seq**] [*cc_opt*] **terminator** [*terminator_opt*] *terminator_param*

DESCRIPTION

This program attempts to detect deadlocks (i.e. states without successors) in 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**. It is based on the ‘‘bit state space’’ verification technique proposed by Gerard Holzmann.

When the program detects deadlock states, it displays diagnostic sequences, i.e., execution sequences leading from the initial state to the deadlock states. Diagnostic sequences are displayed using the simple SEQ format (see the **seq**(LOCAL) man page for a description of this format).

Note: the deadlock detection performed using this technique is partial: due to potential hash-code collisions, some states may not be visited. Therefore, this program is likely to find only a subset of the existing deadlocks. It might even fail to find deadlocks when they exist.

Note: in its first form (i.e., when applied to the BCG graph *spec***.bcg**), this program is probably not the best way to perform deadlock detection.

Note: the **exhibitor**(LOCAL) program can also be used to find the shortest sequence leading to a deadlock.

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 *terminator_options* are currently available:

-depth *depth*
Consider only execution sequences whose number of transitions is less or equal than *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.

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

-all Print all diagnostic sequences. Not a default option.

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

-decr
Print only those diagnostic sequences which are shorter than the last diagnostic sequence previously diplayed. 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).

by default
(in absence of **-none**, **-all**, **-first**, **-decr**)
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).

The parameters *terminator_param* have the following formats, where *order* is an integer denoting the order in which the transitions will be fired (see functions **`CAESAR_CREATE_EDGE_LIST()`** and **`CAESAR_CREATE_STACK_1()`**, in the **caesar_edge**(LOCAL) and **caesar_stack_1**(LOCAL) manual pages), and where *size*, *size_1*, and *size_2* are integer numbers denoting the sizes of bitmap tables:

* If *terminator_param* = *order size*
=> automatic search using a single bitmap table with *size* entries. If *size* is equal to zero, then the bitmap table will be as large as possible, within the limits of the available computer memory.

* If *terminator_param* = *order size_1 size_2*
=> automatic search using two bitmap tables with *size_1* and *size_2* entries, respectively.

* If *terminator_param* is empty
=> interactive mode.

EXIT STATUS

Exit status is 0 if everything is alright, 1 otherwise.

DIAGNOSTICS

When the source is erroneous, error messages are issued.

AUTHOR

Hubert Garavel (INRIA Rhone-Alpes)

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)

FILES

The source code of this tool is available in file **$CADP/src/open_caesar/terminator.c**

SEE ALSO

OPEN/CAESAR Reference Manual, **bcg**(LOCAL), **bcg_open**(LOCAL), **caesar**(LOCAL), **caesar.adt**(LOCAL), **exhibitor**(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