NAME

caesar_diagnostic_1 – the “diagnostic_1” library of OPEN/CAESAR

PURPOSE

The “diagnostic_1” library provides primitives for managing diagnostics.

USAGE

The “diagnostic_1” library consists of:

- a predefined header file **`caesar_diagnostic_1.h`**;
- the precompiled library file **`libcaesar.a`**, which implements the features described in **`caesar_diagnostic_1.h`**.

Note: The “diagnostic_1” library is a software layer built above the primitives offered by the “standard”, “edge” and “stack_1” libraries, and by the *OPEN/CAESAR* graph module.

Note: The “diagnostic_1” library relies on the “edge” and “stack_1” libraries. Therefore, when using the “diagnostic_1” library, there are restrictions concerning the use of the “edge” and “stack_1” library primitives. These restrictions are listed in the sequel.

DESCRIPTION

A “diagnostic” is an execution sequence, i.e., a list of states and transitions starting from the initial state of the graph and leading to a given state.

The “size” of a diagnostic is defined to be the number of states (not the number of transitions) in the corresponding execution sequence. This definition is compatible with the convention used for function **`CAESAR_DEPTH_STACK_1()`**.

Diagnostics are to be used during depth-first graph exploration. For this reason, this library uses (and is compatible with) stacks provided by the “stack_1” library. Let’s consider, for instance, an *OPEN/CAESAR* user program that searches for deadlocks. Every time a deadlock state is detected, the appropriate diagnostic is the contents of the stack, i.e., the execution sequence leading from the initial state to the current deadlock state.

The “diagnostic_1” allows to control the way diagnostics are printed. For instance, it allows to display only the shortest diagnostic sequence detected. Other “strategies” are also available. The data structure used to store diagnostics and related information is called a “diagnostic structure”.

FEATURES

....

`CAESAR_TYPE_DIAGNOSTIC_1`

```
typedef CAESAR_TYPE_ABSTRACT (...) CAESAR_TYPE_DIAGNOSTIC_1;
```

This type denotes a pointer to the concrete representation of a diagnostic structure, which is supposed to be “opaque”.

....

`CAESAR_TYPE_STRATEGY_DIAGNOSTIC_1`

```
typedef enum {
     CAESAR_NONE_DIAGNOSTIC_1,
     CAESAR_ALL_DIAGNOSTIC_1,
     CAESAR_FIRST_DIAGNOSTIC_1,
     CAESAR_DECREASING_DIAGNOSTIC_1,
     CAESAR_SHORTEST_DIAGNOSTIC_1
}       CAESAR_TYPE_STRATEGY_DIAGNOSTIC_1;
```

This enumerated type defines the ‘‘strategy’’ used to print diagnostics during graph exploration. Two steps are to be distinguished: every time an execution sequence of interest is detected, it is ‘‘recorded’’ using the **CAESAR_RECORD_DIAGNOSTIC_1()** function defined below. When the depth-first exploration is completed, a ‘‘summary request’’ is performed using the **CAESAR_SUMMARIZE_DIAGNOSTIC_1()** function defined below. The effect of these two functions depends on the chosen strategy:

- With **CAESAR_NONE_DIAGNOSTIC_1**, only the result of the exploration is printed (i.e., some diagnostic has been found or not). Diagnostics themselves are not printed.
- With **CAESAR_ALL_DIAGNOSTIC_1**, all diagnostics are printed when they are recorded.
- With **CAESAR_FIRST_DIAGNOSTIC_1**, only the first diagnostic is printed when it is recorded; the next ones will not be printed.
- With **CAESAR_DECREASING_DIAGNOSTIC_1**, the first diagnostic is printed when it is recorded; the next ones will be printed iff their size is strictly smaller than all previously recorded diagnostics.
- With **CAESAR_SHORTEST_DIAGNOSTIC_1**, diagnostics are not printed when they are recorded, but the diagnostic with the smallest size is stored in a diagnostic structure. This shortest diagnostic (if any) will be printed when a summary request is emitted.

....

CAESAR_CREATE_DIAGNOSTIC_1

```
void CAESAR_CREATE_DIAGNOSTIC_1 (CAESAR_D, CAESAR_STRATEGY,
                                 CAESAR_DEPTH, CAESAR_FILE,
                                 CAESAR_PROGRESS, CAESAR_HEADER,
                                 CAESAR_FOOTER, CAESAR_SEPARATOR,
                                 CAESAR_REPORT)
   CAESAR_TYPE_DIAGNOSTIC_1 *CAESAR_D;
   CAESAR_TYPE_STRATEGY_DIAGNOSTIC_1 CAESAR_STRATEGY;
   CAESAR_TYPE_NATURAL CAESAR_DEPTH;
   CAESAR_TYPE_FILE CAESAR_FILE;
   CAESAR_TYPE_STRING CAESAR_PROGRESS;
   CAESAR_TYPE_STRING CAESAR_HEADER;
   CAESAR_TYPE_STRING CAESAR_FOOTER;
   CAESAR_TYPE_STRING CAESAR_SEPARATOR;
   CAESAR_TYPE_STRING CAESAR_REPORT;
   { ... }
```

This procedure allocates a diagnostic structure using **CAESAR_CREATE()** and assigns its address to ∗**CAESAR_D**. If the allocation fails, the **NULL** value is assigned to ∗**CAESAR_D**.

Note: because **CAESAR_TYPE_DIAGNOSTIC_1** is a pointer type, any variable **CAESAR_D** of type **CAESAR_TYPE_DIAGNOSTIC_1** must be allocated before used, for instance using:

```
CAESAR_CREATE_DIAGNOSTIC_1 (&CAESAR_D, ...);
```

Note: Before calling this procedure, the **CAESAR_INIT_STACK_1()** procedure of the ‘‘stack_1’’ library should have been invoked (because the “diagnostic_1” library uses the “stack_1” library). Also, the restrictions concerning the **CAESAR_INIT_STACK_1()** procedure should be observed.

The actual values of the remaining formal parameters will be stored and associated to the diagnostic structure pointed to by ***CAESAR_D**.

The value of **CAESAR_STRATEGY** determines the strategy used for this diagnostic structure.

The value of **CAESAR_DEPTH** determines the “maximal depth” used for this diagnostic structure, i.e., an upper bound on the sizes of the diagnostics. If **CAESAR_DEPTH** is equal to zero, there is no upper bound.

The value of **CAESAR_FILE** determines the output file to which the diagnostics will be printed. Before any diagnostic is printed, **CAESAR_FILE** must have been properly opened, for instance using **fopen(3)**.

The values of **CAESAR_PROGRESS**, **CAESAR_HEADER**, **CAESAR_FOOTER**, **CAESAR_SEPARATOR**, and **CAESAR_REPORT** are character strings to be used for printing the diagnostics. The exact semantics of these parameters will not be defined here. For interoperability with the other tools of the CADP toolset (see the **seq** manual page for a definition of the SEQ format), it is advised to use the following actual values:

```
CAESAR_PROGRESS  = "*** sequence(s) found at depth "
CAESAR_HEADER    = "*** sequence found at depth %u\n\n\001<initial state>\002\n"
CAESAR_FOOTER    = "\001<goal state>\002\n\n"
CAESAR_SEPARATOR = "[]\n\n"
CAESAR_REPORT    = "*** no sequence found\n\n"
```

or, depending on the context:

```
CAESAR_PROGRESS  = "*** deadlock(s) found at depth "
CAESAR_HEADER    = "*** deadlock found at depth %u\n\n\001<initial state>\002\n"
CAESAR_FOOTER    = "<deadlock>\n\n"
CAESAR_SEPARATOR = "[]\n\n"
CAESAR_REPORT    = "*** no deadlock found\n\n"
```

....

CAESAR_DELETE_DIAGNOSTIC_1

```
void CAESAR_DELETE_DIAGNOSTIC_1 (CAESAR_D)
   CAESAR_TYPE_DIAGNOSTIC_1 *CAESAR_D;
   { ... }
```

This procedure frees the memory space corresponding to the diagnostic structure pointed to by ***CAESAR_D** using **CAESAR_DELETE()**. Afterwards, the **NULL** value is assigned to ***CAESAR_D**.

....

CAESAR_RECORD_DIAGNOSTIC_1

```
void CAESAR_RECORD_DIAGNOSTIC_1 (CAESAR_D, CAESAR_K)
   CAESAR_TYPE_DIAGNOSTIC_1 CAESAR_D;
   CAESAR_TYPE_STACK_1 CAESAR_K;
   { ... }
```

This procedure records into the diagnostic structure pointed to by ∗**CAESAR_D** the diagnostic contained in the stack pointed to by ∗**CAESAR_K**. The stack pointed to by ∗**CAESAR_K** should contain at least one item (the initial state of the sequence); if it is empty, the result is undefined. The effects of this procedure depend on the strategy and maximal depth associated with the diagnostic structure:

- With **CAESAR_NONE_DIAGNOSTIC_1**, the diagnostic is not printed but stored into the diagnostic structure.
- With **CAESAR_ALL_DIAGNOSTIC_1**, the diagnostic is immediately printed.
- With **CAESAR_FIRST_DIAGNOSTIC_1**, the diagnostic is immediately printed if it is the first one to be recorded; otherwise nothing is done.
- With **CAESAR_DECREASING_DIAGNOSTIC_1**, the diagnostic is immediately printed if its size is less or equal than the maximal depth (if the maximal depth is not equal to zero) and strictly less than the sizes of all previously recorded diagnostics.
- With **CAESAR_SHORTEST_DIAGNOSTIC_1**, the diagnostic is not printed but stored into the diagnostic structure if its size os less or equal than the maximal depth (if the maximal depth is not equal to zero) and strictly less than the sizes of all previously recorded diagnostics. If the character string **CAESAR_PROGRESS** specified for ∗**CAESAR_D** is not the **NULL** pointer, it is printed, together with the size of the recorded diagnostic.

....

CAESAR_SUMMARIZE_DIAGNOSTIC_1

```
void CAESAR_SUMMARIZE_DIAGNOSTIC_1 (CAESAR_D)
   CAESAR_TYPE_DIAGNOSTIC_1 CAESAR_D;
   { ... }
```

This procedure summarizes the current status of the diagnostic structure pointed to by ∗**CAESAR_D**. The effects of this procedure depends on the strategy associated with the diagnostic structure:

- With **CAESAR_NONE_DIAGNOSTIC_1**, the result of the exploration is displayed: if diagnostics have been recorded, the size of the most recently recorded one is printed; otherwise, the character string **CAESAR_REPORT** is printed if no diagnostic has been recorded into the diagnostic structure.
- With **CAESAR_ALL_DIAGNOSTIC_1**, the character string **CAESAR_REPORT** is printed if no diagnostic has been previously recorded into the diagnostic structure.
- With **CAESAR_FIRST_DIAGNOSTIC_1**, same as for **CAESAR_ALL_DIAGNOSTIC_1**.
- With **CAESAR_DECREASING_DIAGNOSTIC_1**, same as for **CAESAR_ALL_DIAGNOSTIC_1**.
- With **CAESAR_SHORTEST_DIAGNOSTIC_1**, the shortest diagnostic recorded is printed, or the character string **CAESAR_REPORT** is printed if no diagnostic has been recorded into the diagnostic structure.

....

CAESAR_EMPTY_DIAGNOSTIC_1

```
CAESAR_TYPE_BOOLEAN CAESAR_EMPTY_DIAGNOSTIC_1 (CAESAR_D)
   CAESAR_TYPE_DIAGNOSTIC_1 CAESAR_D;
   { ... }
```

This function returns the boolean value **CAESAR_TRUE** iff at least one diagnostic has been previously recorded (using the **CAESAR_RECORD_DIAGNOSTIC_1()** function) into the diagnostic structure pointed to by ***CAESAR_D**.

....

CAESAR_BACKTRACK_DIAGNOSTIC_1

```
CAESAR_TYPE_BOOLEAN CAESAR_BACKTRACK_DIAGNOSTIC_1 (CAESAR_D, CAESAR_DEPTH)
   CAESAR_TYPE_DIAGNOSTIC_1 CAESAR_D;
   CAESAR_TYPE_NATURAL CAESAR_DEPTH;
   { ... }
```

This function returns the boolean value **CAESAR_TRUE** iff the exploration should stop and backtrack when reaching depth **CAESAR_DEPTH**. For instance, in a depth-first search, the actual value of **CAESAR_DEPTH** should be equal to the current depth of the stack. The result depends on the strategy and maximal depth associated with the diagnostic structure pointed to by ***CAESAR_D**:

- With **CAESAR_NONE_DIAGNOSTIC_1**, the result is **CAESAR_TRUE** iff **CAESAR_DEPTH** is strictly greater than the maximal depth (if the maximal depth is not equal to zero).
- With **CAESAR_ALL_DIAGNOSTIC_1**, same as for **CAESAR_NONE_DIAGNOSTIC_1**.
- With **CAESAR_FIRST_DIAGNOSTIC_1**, the result is **CAESAR_TRUE** iff **CAESAR_DEPTH** is strictly greater than the maximal depth (if the maximal depth is not equal to zero) or if some diagnostic has been previously recorded into the diagnostic structure.
- With **CAESAR_DECREASING_DIAGNOSTIC_1**, the result is **CAESAR_TRUE** iff **CAESAR_DEPTH** is strictly greater than the maximal depth (if the maximal depth is not equal to zero) or if it is greater or equal to the size of some previously recorded diagnostic.
- With **CAESAR_SHORTEST_DIAGNOSTIC_1**, same as for **CAESAR_DECREASING_DIAGNOSTIC_1**.

....

AUTHOR(S)

Hubert Garavel

FILES

$CADP/incl/caesar_graph.h	interface of the graph module
$CADP/incl/caesar_*.h	interfaces of the storage module
$CADP/bin.‘arch‘/libcaesar.a	object code of the storage module
$CADP/src/open_caesar/*.c	source code of various exploration modules
$CADP/com/lotos.open	shell script to run OPEN/CAESAR

SEE ALSO

Reference Manuals of OPEN/CAESAR, CAESAR, and CAESAR.ADT, **lotos.open**(LOCAL), **caesar**(LOCAL), **caesar.adt**(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

Known bugs are described in the Reference Manual of OPEN/CAESAR. Please report new bugs to cadp@inria.fr