NAME

caesar_solve_2 – the “solve_2” library of OPEN/CAESAR

PURPOSE

The “solve_2” library provides primitives for solving linear equation systems, which are provided “on the fly”. This library can be used as a back-end for various on the fly verification tools that formulate their corresponding problems (e.g., probabilistic verification) in terms of linear equation systems.

USAGE

The “solve_2” library consists of:

- a predefined header file **caesar_solve_2.h**;
- the precompiled library file **libcaesar.a**, which implements the features described in **caesar_solve_2.h**.

Note: The “solve_2” library is a software layer built above the primitives offered by the “standard”, “area_1”, and “table_1” libraries, and by the *OPEN/CAESAR* graph module.

LINEAR EQUATION SYSTEMS

A “linear equation system” is a set of (possibly recursive) equations with numerical variables on their left-hand sides and linear algebraic expressions on their right-hand sides.

The expressions on the right-hand sides of equations are built from numerical variables (**X1**, **X2**, etc.) and real numbers (**3.1416**, **−0.5**, etc.) using arithmetic operators (**+**, ∗).

Equations are of the following form:

```
Xi = Ci1 * X1 + ... + Cin * Xn + Bi
```

where the real numbers **Cij** are the coefficients of variables **Xj** and the real numbers **Bi** are constants, for i, j between 1 and n. If a coefficient **Cij** is equal to 0, the term **Cij * Xj** is omitted.

A variable is defined in a linear equation system if it occurs as left-hand side in an equation of the system. Multiple definitions of a variable are forbidden, i.e., a variable can occur as left-hand side in at most one equation of a linear equation system. Each variable occurring on the right-hand side of an equation in a linear equation system must be defined by some equation of the system.

If the formula on the right-hand side of the equation defining a variable **Xi** contains another variable **Xj**, there is a dependency from **Xi** to **Xj**.

An example of a linear equation system containing four equations is shown below.

```
X1 = 0.4 * X2 + 0.6 * X3
X2 = 0.5 * X1 + 0.2
X3 = 0.3
```

This system contains a cyclic dependency between variables **X1** and **X2**.

ON THE FLY RESOLUTION

Given a linear equation system, the on the fly (or local) resolution problem consists in computing the value of a particular variable defined in the system. Contrary to global resolution, which consists in computing the values of all variables defined in the system (and therefore must examine all equations of the system), on the fly resolution computes the value of a variable without necessarily examining all equations of the system. This resolution technique allows to construct the linear equation system in a demand-driven manner, and hence it is useful for building tools for on the fly verification, which explore one or more labelled transition systems incrementally.

The on the fly resolution method for linear equation systems implemented in the ‘‘solve_2’’ library (see [MR18]) proceeds as follows. Each system has an associated resolution routine responsible for computing the values of the variables defined in the system.

The resolution routine associated to a linear equation system is easier to develop using a representation of the system as a dependency graph (also known as Signal Flow Graph), which provides a more intuitive view of the dependencies between variables. Given an equation system, its corresponding dependency graph is defined as follows:

- For each variable **`Xi`** occurring in the system, there is a vertex **`Xi`** in the dependency graph.
- For each dependency from a variable **`Xi`** to a variable **`Xj`** such that the term **`Cij * Xj`** occurs on the right-hand side of the equation defining **`Xi`**, there is an edge ‘‘(**`Xi`**, **`Cij`**, **`Xj`**)’’ in the dependency graph.
- There is a special sink vertex **`X0`** (i.e., without any outgoing transitions), which denotes the constant **`1.0`**.

To construct its dependency graph, the linear equation system shown above must be first transformed into the equivalent form below, by replacing each constant **`Bi`** by a term **`Bi * X0`** and adding the equation **`X0 = 1.0`**:

```
X0 = 1.0
X1 = 0.4 * X2 + 0.6 * X3
X2 = 0.5 * X1 + 0.2 * X0
X3 = 0.3 * X0
```

Its dependency graph is the following:

```
(X1, 0.4, X2)
(X1, 0.6, X3)
(X2, 0.5, X1)
(X2, 0.2, X0)
(X3, 0.3, X0)
```

The resolution routine associated to a linear equation system will explore forward the corresponding dependency graph and will propagate backward the values of variables already stabilized (i.e., the value of which has been determined). When solving a variable, only the part of the dependency graph relevant for computing the value of the variable is explored. For example, solving the variable **`X3`** of the system above requires to explore only the dependency ‘‘(**`X3`**, **`0.3`**, **`X0`**)’’, yielding the value **`0.3`** for **`X3`**. The resolution of variables **`X1`** and **`X2`**, which are present on a cycle of the dependency graph, is carried out by a traditional linear resolution algorithm (e.g., an iterative algorithm) executed on the equations defining **`X1`** and **`X2`**, yielding the values 0.325 and 0.3625, respectively.

To enable on the fly exploration, the dependency graphs associated to linear equation systems are represented in a generic, implicit manner using a scheme similar to the one defined by the *OPEN/CAESAR* graph module for representing labelled transition systems. This representation roughly consists of the following ingredients (see procedure **`CAESAR_CREATE_SOLVE_2()`** below for additional details):

- Variables (vertices of the dependency graph) are represented as pointers to memory areas of fixed size. The precise meaning of the variable contents is defined by the application program and is not relevant for the resolution algorithms.

- A linear equation system is equipped with several functions computing various information about the variables defined in the system: a comparison function, a hashing function, a printing function, and an iterator procedure which enumerates the successors of a variable in the dependency graph.

To speed up the resolution process, a linear equation system has associated an internal table which stores the variables already explored during the previous calls of the resolution routine associated to the system. This avoids recomputations of variables by subsequent calls of the resolution routine, leading (for the sparse linear equation systems encountered in verification problems) to an overall resolution process of time complexity linear in the size of the system (number of variables and operators).

DESCRIPTION

The ‘‘solve_2’’ library allows to create and handle linear equation systems on the fly, providing procedures for resolution, inspection, and writing information to text files.

FEATURES

....

CAESAR_TYPE_SOLVE_2

```
typedef CAESAR_TYPE_ABSTRACT (...) CAESAR_TYPE_SOLVE_2;
```

This type denotes a pointer to the concrete representation of a linear equation system. This representation is supposed to be ‘‘opaque’’.

....

CAESAR_TYPE_REAL

```
typedef double CAESAR_TYPE_REAL;
```

CAESAR_TYPE_REAL is the real number type used in the ‘‘solve_2’’ library.

....

CAESAR_TYPE_ERROR_SOLVE_2

```
typedef enum {
      CAESAR_NONE_SOLVE_2,
      CAESAR_MULTIPLE_RESOLUTION_SOLVE_2,
      CAESAR_MEMORY_SHORTAGE_SOLVE_2,
      CAESAR_SINGULAR_SOLVE_2
}     CAESAR_TYPE_ERROR_SOLVE_2;
```

This enumerated type defines the error codes produced as a side effect by calls to the function **CAESAR_COMPUTE_SOLVE_2()** (see below), which performs the resolution of a variable defined in a linear equation system. The error codes have the following meaning:

- **CAESAR_NONE_SOLVE_2** indicates that the resolution was performed successfully.
- **CAESAR_MULTIPLE_RESOLUTION_SOLVE_2** indicates that another resolution of a variable of the system was already performed, whereas at the creation of the linear equation system (see

procedure **CAESAR_CREATE_SOLVE_2()** below) a single resolution was specified for the system.

- **CAESAR_MEMORY_SHORTAGE_SOLVE_2** indicates that a memory allocation failed during the resolution.
- **CAESAR_SINGULAR_SOLVE_2** indicates that the linear equation system contains singularities, i.e., either it has no solution, or it does not have a unique solution.

Note: The error code produced by a call to **CAESAR_COMPUTE_SOLVE_2()** can be obtained by using the function **CAESAR_STATUS_COMPUTE_SOLVE_2()** (see below).

....

CAESAR_CREATE_SOLVE_2

```
void CAESAR_CREATE_SOLVE_2 (CAESAR_L,
                            CAESAR_UNIQUE_RESOLUTION,
                            CAESAR_SOLVE_MODE,
                            CAESAR_EPSILON,
                            CAESAR_VARIABLE_AREA,
                            CAESAR_LIMIT_SIZE,
                            CAESAR_HASH_SIZE,
                            CAESAR_PRIME,
                            CAESAR_VARIABLE_COMPARE,
                            CAESAR_VARIABLE_HASH,
                            CAESAR_VARIABLE_PRINT,
                            CAESAR_VARIABLE_ITERATE,
                            CAESAR_INFO)
   CAESAR_TYPE_SOLVE_2 *CAESAR_L;
   CAESAR_TYPE_BOOLEAN CAESAR_UNIQUE_RESOLUTION;
   CAESAR_TYPE_NATURAL CAESAR_SOLVE_MODE;
   CAESAR_TYPE_REAL CAESAR_EPSILON;
   CAESAR_TYPE_AREA_1 CAESAR_VARIABLE_AREA;
   CAESAR_TYPE_NATURAL CAESAR_LIMIT_SIZE;
   CAESAR_TYPE_NATURAL CAESAR_HASH_SIZE;
   CAESAR_TYPE_BOOLEAN CAESAR_PRIME;
   CAESAR_TYPE_COMPARE_FUNCTION CAESAR_VARIABLE_COMPARE;
   CAESAR_TYPE_HASH_FUNCTION CAESAR_VARIABLE_HASH;
   CAESAR_TYPE_PRINT_FUNCTION CAESAR_VARIABLE_PRINT;
   void (*CAESAR_VARIABLE_ITERATE) (CAESAR_TYPE_POINTER, CAESAR_TYPE_POINTER,
      void (*) (CAESAR_TYPE_REAL, CAESAR_TYPE_POINTER));
   CAESAR_TYPE_POINTER CAESAR_INFO;
   { ... }
```

This procedure allocates a linear equation system using **CAESAR_CREATE()** and assigns its address to ∗**CAESAR_L**. If the allocation fails, the **NULL** value is assigned to ∗**CAESAR_L**.

Note: Because **CAESAR_TYPE_SOLVE_2** is a pointer type, any variable **CAESAR_L** of type **CAESAR_TYPE_SOLVE_2** must be allocated before used, for instance using:

```
CAESAR_CREATE_SOLVE_2 (&CAESAR_L, ...);
```

The actual value of the formal parameter **CAESAR_UNIQUE_RESOLUTION** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used to assign a boolean indicating whether only one variable (or several variables) of the system will be solved.

The actual value of the formal parameter **CAESAR_SOLVE_MODE** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used to assign the resolution mode, determining which algorithm will be used by the resolution routine associated to the system.

Currently, the following resolution modes are available:

- Mode 0 corresponds to a resolution algorithm based upon a depth-first search of the dependency graph associated to the system, with detection of strongly connected components (SCCs) and iterative resolution of the variables contained in the SCCs [MR18]. This mode works only for stochastic linear equation systems, whose equations **Xi** = **Ci1 ∗ X1** + ... + **Cin ∗ Xn** + **Bi** are such that all coefficients **Cij** and constant **Bi** are positive, and their sum **Ci1** + ... + **Cin** + **Bi** is less than or equal to 1.0.
- (no other resolution mode available yet)

If the resolution mode given by **CAESAR_SOLVE_MODE** is not among the resolution modes above, or the linear equation system does not have the form expected by that resolution mode, the effect is undefined.

The actual value of the formal parameter **CAESAR_EPSILON** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used to assign the precision of comparisons between real numbers during the resolution. If this value is equal to zero, it is replaced by the default value **1E−6**.

The actual value of the formal parameter **CAESAR_VARIABLE_AREA** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used to assign the (constant) size and (constant) alignment factor of the variables defined in the system.

The actual value of the formal parameter **CAESAR_LIMIT_SIZE** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used to assign the maximal size (number of items) of the internal table associated to the system. This value must be less or equal to a predefined value M (see the "table_1" library). If it is equal to zero, it is replaced by the default value M.

The actual value of the formal parameter **CAESAR_HASH_SIZE** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used to assign the size (number of entries) of the hash-table accompanying the internal table associated to the system. If this value is equal to zero, it is replaced by a default value greater than zero (see the "table_1" library).

The actual value of the formal parameter **CAESAR_PRIME** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used to assign a boolean value allowing to adjust the size of the hash-table accompanying the internal table associated to the system. If this value is equal to **CAESAR_TRUE** and if the value of **CAESAR_HASH_SIZE** is not a prime number, this value will be replaced by the nearest smaller prime number (since some hash functions require prime modulus). Otherwise, the value of **CAESAR_HASH_SIZE** will be kept unchanged (see the "table_1" library).

The actual value of the formal parameter **CAESAR_VARIABLE_COMPARE** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used as a comparison function for the variables defined in the system.

Precisely, the actual value of **CAESAR_VARIABLE_COMPARE** should be a pointer to a comparison function **caesar_f** with two parameters **caesar_variable_1** and **caesar_variable_2** that returns **CAESAR_TRUE** (resp. **CAESAR_FALSE**) if the variables pointed to by **caesar_variable_1** and **caesar_variable_2** are equal (resp. different). A pointer to the linear equation system (i.e., the value assigned to ∗**CAESAR_B**) can be obtained within **caesar_f** by calling the function

CAESAR_CURRENT_SYSTEM_SOLVE_1() (see below).

The actual value of the formal parameter **CAESAR_VARIABLE_HASH** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used as a hash-function for the variables defined in the system.

Precisely, the actual value of **CAESAR_VARIABLE_HASH** should be a pointer to a hash function **caesar_f** with two parameters **caesar_variable** and **caesar_modulus** that returns a hash-value computed on the byte string **caesar_variable [0]** up to **caesar_variable [caesar_size - 1]**, where the actual value of **caesar_size** will always be equal to the size of the variable pointed to by **caesar_variable**. This hash-value must belong to the range 0..**caesar_modulus**-1. A pointer to the linear equation system (i.e., the value assigned to ∗**CAESAR_L**) can be obtained within **caesar_f** by calling the function **CAESAR_CURRENT_SYSTEM_SOLVE_2()** (see below).

The actual value of the formal parameter **CAESAR_VARIABLE_PRINT** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used as a printing procedure for the variables defined in the system.

Precisely, the actual value of **CAESAR_VARIABLE_PRINT** should be a pointer to a printing procedure **caesar_p** with two parameters **caesar_file** and **caesar_variable** that prints to file **caesar_file** information about the contents of the variable pointed to by **caesar_variable**. A pointer to the linear equation system (i.e., the value assigned to ∗**CAESAR_L**) can be obtained within **caesar_p** by calling the function **CAESAR_CURRENT_SYSTEM_SOLVE_2()** (see below).

The actual value of the formal parameter **CAESAR_VARIABLE_ITERATE** will be stored and associated to the linear equation system pointed to by ∗**CAESAR_L**. It will be used as an iterator procedure enumerating all successors of the variables defined in the system.

Any user-defined procedure **caesar_p** can be used as an actual value for formal parameter **CAESAR_VARIABLE_ITERATE**, provided that its declaration has the form:

```
void caesar_p (caesar_variable_1, caesar_variable_2, caesar_loop)
   CAESAR_TYPE_POINTER caesar_variable_1;
   CAESAR_TYPE_POINTER caesar_variable_2;
   void (*caesar_loop) (CAESAR_TYPE_REAL, CAESAR_TYPE_POINTER);
   { ... }
```

This procedure **caesar_p** enumerates all successors of the variable pointed to by **caesar_variable_1**. A pointer to the linear equation system (i.e., the value assigned to ∗**CAESAR_L**) can be obtained within **caesar_p** by calling the function **CAESAR_CURRENT_SYSTEM_SOLVE_2()** (see below). At each iteration performed by **caesar_p**, two actions must be carried out:

- First, the variable pointed to by **caesar_variable_2** must be assigned a new value, such that **caesar_variable_1** and **caesar_variable_2** are the source and target vertices of an edge in the dependency graph.
- Second, the procedure pointed to by **caesar_loop** must be called. The actual value of the formal parameter **caesar_loop** is a procedure **caesar_q** whose declaration has the form:

```
void caesar_q (caesar_coefficient_2, caesar_variable_2)
   CAESAR_TYPE_REAL caesar_coefficient_2;
   CAESAR_TYPE_POINTER caesar_variable_2;
   { ... }
```

Therefore, each call to the procedure pointed to by **caesar_loop** must have the following parameters:

```
(*caesar_loop) (caesar_coefficient_2, caesar_variable_2)
```

Parameter **caesar_coefficient_2** represents the coefficient of the variable **caesar_variable_2** on the right-hand side of the equation defining **caesar_variable_1**, i.e., “(**caesar_variable_1**, **caesar_coefficient_2**, **caesar_variable_2**)” represents an edge of the dependency graph.

Note: The memory area pointed to by the parameter **caesar_variable_1** contains a variable and should neither be modified, nor freed by the procedure **caesar_p**.

Note: The memory area pointed to by the parameter **caesar_variable_2** is already allocated and should not be freed by the procedure **caesar_p**.

The value of **CAESAR_INFO** has no effect on the execution of procedure **CAESAR_CREATE_SOLVE_2 ()**. Parameter **CAESAR_INFO** is intended to serve for future extensions of this procedure; when using the current version of the “solve_2” library, it is recommended to set this parameter to **NULL**.

....

CAESAR_CURRENT_SYSTEM_SOLVE_2

```
CAESAR_TYPE_SOLVE_2 CAESAR_CURRENT_SYSTEM_SOLVE_2 ()
   { ... }
```

This function returns a pointer to the linear equation system which is currently under resolution. It should be called only within the functions and procedures given as actual values for the formal parameters **CAESAR_VARIABLE_COMPARE**, **CAESAR_VARIABLE_HASH**, **CAESAR_VARIABLE_PRINT**, and **CAESAR_VARIABLE_ITERATE** of procedure **CAESAR_CREATE_SOLVE_2 ()** (see above); in this case, the result is a pointer to the linear equation system created by the call to **CAESAR_CREATE_SOLVE_2 ()**. If this function is called anywhere else in the application program, the result is undefined.

Note: This function allows to invoke, within the five aforementioned functions and procedures, various primitives of the “solve_2” library on the current linear equation system (e.g., resolution, printing, etc.).

....

CAESAR_DELETE_SOLVE_2

```
void CAESAR_DELETE_SOLVE_2 (CAESAR_L)
   CAESAR_TYPE_SOLVE_2 *CAESAR_L;
   { ... }
```

This procedure frees the memory space corresponding to the linear equation system pointed to by ***CAESAR_L** using **CAESAR_DELETE ()**. The variables stored in the internal table allocated during previous resolutions (if any) of the linear equation system are also freed. Afterwards, the **NULL** value is assigned to ***CAESAR_L**.

....

CAESAR_PURGE_SOLVE_2

```
void CAESAR_PURGE_SOLVE_2 (CAESAR_L)
   CAESAR_TYPE_SOLVE_2 CAESAR_L;
```

```
    { ... }
```

This procedure reinitializes the information associated to the linear equation system pointed to by **CAESAR_L**. The internal table associated to the system is emptied using **CAESAR_PURGE_TABLE_1()**. Afterwards, the linear equation system is exactly in the same state as after its creation using **CAESAR_CREATE_SOLVE_2()**.

....

CAESAR_COMPUTE_SOLVE_2

```
CAESAR_TYPE_REAL CAESAR_COMPUTE_SOLVE_2 (CAESAR_L, CAESAR_V)
    CAESAR_TYPE_SOLVE_2 CAESAR_L;
    CAESAR_TYPE_POINTER CAESAR_V;
    { ... }
```

This function computes the value of the variable pointed to by **CAESAR_V**, which must be defined in the linear equation system pointed to by **CAESAR_L**. It also sets a field of type **CAESAR_TYPE_ERROR_SOLVE_2** associated to the system, indicating whether the resolution was carried out successfully or not; this field can be inspected using the function **CAESAR_STATUS_COMPUTE_SOLVE_2()** (see below).

....

CAESAR_STATUS_COMPUTE_SOLVE_2

```
CAESAR_TYPE_ERROR_SOLVE_2 CAESAR_STATUS_COMPUTE_SOLVE_2 (CAESAR_L)
    CAESAR_TYPE_SOLVE_2 CAESAR_L;
    { ... }
```

This function returns the status of the last resolution performed by a call to the function **CAESAR_COMPUTE_SOLVE_2()** (see above) on a variable defined in the linear equation system pointed to by **CAESAR_L**.

....

CAESAR_FORMAT_SOLVE_2

```
CAESAR_FORMAT CAESAR_FORMAT_SOLVE_2 (CAESAR_L, CAESAR_FORMAT)
    CAESAR_TYPE_SOLVE_2 CAESAR_L;
    CAESAR_TYPE_FORMAT CAESAR_FORMAT;
    { ... }
```

This function allows to control the format under which the linear equation system pointed to by **CAESAR_L** will be printed by the procedure **CAESAR_PRINT_SOLVE_2()** (see below). Currently, the following formats are available:

- With format 0, statistical information concerning the linear equation system is displayed, such as: the size of variables and resolution mode, the number of variables explored during resolution, etc.
- With format 1, statistical information concerning the internal table associated to the system is printed using the procedure **CAESAR_PRINT_TABLE_1()**.
- With format 2, the contents of the internal table associated to the system is printed using the procedure **CAESAR_PRINT_TABLE_1()**.

- (no other format available yet)

By default, the current format of each linear equation system is initialized to 0.

When called with **CAESAR_FORMAT** between 0 and 2, this function sets the current format of **CAESAR_L** to **CAESAR_FORMAT** and returns an undefined result.

When called with another value of **CAESAR_FORMAT**, this function does not modify the current format of **CAESAR_L** but returns a result defined as follows. If **CAESAR_FORMAT** is equal to the constant **CAESAR_CURRENT_FORMAT**, the result is the value of the current format of **CAESAR_L**. If **CAESAR_FORMAT** is equal to the constant **CAESAR_MAXIMAL_FORMAT**, the result is the maximal format value (i.e., 2). In all other cases, the effect of this function is undefined.

....

CAESAR_PRINT_SOLVE_2

```
void CAESAR_PRINT_SOLVE_2 (CAESAR_FILE, CAESAR_L)
   CAESAR_TYPE_FILE CAESAR_FILE;
   CAESAR_TYPE_SOLVE_2 CAESAR_L;
   { ... }
```

This procedure prints on file **CAESAR_FILE** an ASCII text containing various informations about the linear equation system pointed to by **CAESAR_L**. The nature of these informations is determined by the current format of the linear equation system pointed to by **CAESAR_L**.

Before this procedure is called, **CAESAR_FILE** must have been properly opened, for instance using **fopen(3)**.

BIBLIOGRAPHY

[MR18] Radu Mateescu and Jose Ignacio Requeno. On-the-Fly Model Checking for Extended Action-Based Probabilistic Operators. Springer International Journal on Software Tools for Technology Transfer (STTT), 20(5):563--587, October 2018. Available from **http://hal.inria.fr/hal-01862754/en**

....

AUTHOR(S)

Radu Mateescu

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