NAME

caesar_rename_1 – the ‘‘rename_1’’ library of OPEN/CAESAR

PURPOSE

The ‘‘rename_1’’ library provides primitives for processing ‘‘renaming files’’. These files specify which labels of a graph should be renamed, and how to rename them, using a set of regular expressions.

USAGE

The ‘‘rename_1’’ library consists of:

- a predefined header file **caesar_rename_1.h**;
- the precompiled library file **libcaesar.a**, which implements the features described in **caesar_rename_1.h**.

Note: The ‘‘rename_1’’ library is a software layer built above the primitives offered by the ‘‘standard’’ library.

RENAMING FILES

In this section, we define the format of renaming files, as they are used in the *CADP* toolbox. The next sections will explain how the ‘‘renaming_1’’ library of *OPEN/CAESAR* supports (and extends) this format.

A renaming file is a text file containing a set of ‘‘renaming patterns’’. There is no mandatory suffix (i.e., file extension) for renaming files: any file name can be used; however, it is recommended to use one of the following suffixes ‘‘.ren’’ (recommended) or ‘‘.rename’’.

The syntax of renaming files is described by the following context-free grammar:

```
<axiom> ::= <blanks> <header> <blanks> \n <renaming-list>

<header> ::= rename

<renaming-list> ::= <empty>
                  |  <renaming> <renaming-list>

<renaming> ::= <left-pattern> -> <right-pattern> \n

<left-pattern> ::= <blanks> <left-regexp> <blanks>
                 |  <blanks> "<left-regexp>" <blanks>

<right-pattern> ::= <blanks> <right-regexp> <blanks>
                  |  <blanks> "<right-regexp>" <blanks>
```

where:

- **\n** denotes the newline character;
- **<blanks>** is any sequence of spaces, tabulations, carriage returns, newlines, vertical tabulations, or form feeds; these characters are those recognised by the POSIX function **isspace()**; they are always skipped and ignored;
- **<empty>** denotes the empty sequence;
- **<left-regexp>** is a character string specifying a regular expression according to the definition given in the **regexp**(LOCAL) manual page. The **<left-regexp>** may be enclosed between

double quotes, which will be removed and ignored.

- **<right-regexp>** is a character string specifying a ''replacement'' expression according to the definition given in the manual page of the POSIX **regexp(LOCAL)** command. The **<right-regexp>** can be surrounded by double quotes, which will be removed and ignored.

Note: renaming files are case sensitive: upper-case and lower-case letters are considered to be different.

Note: in the **<renaming-list>**, lines that are empty or contain only blanks will be ignored.

Semantically, a renaming file behaves as function that maps a character string S (presumably representing the label of a transition) to another character string (the renamed label).

More precisely, the effect of a renaming file F on a character string S is defined as follows. The **<renaming-list>** of S is scanned from top to bottom in order to determine the first **<left-regexp>** matched by S. If this **<left-regexp>** does not exist, S is kept unchanged. If it exists, S will be renamed into the corresponding **<right-regexp>**. Renaming is performed according to the conventions for text substitution defined in the POSIX command **regexp(LOCAL)**.

Note: If S matches several **<left-regexp>**s, only the first one is taken into account. Renamings do not cumulate (although such behaviour can be explicitly programmed by invoking function **CAESAR_APPLY_RENAME_1()** several times).

For instance, the following file:

```
rename
GET -> PUT
PUT -> GET
```

will swap the labels named **"GET"** and **"PUT"**. Similarly, the following file:

```
rename
"PUT !\([A-Z]*\) !\([A-Z]*\)" -> "PUT !\2 !\1"
```

will swap the offers of **"PUT"** labels, e.g., **"PUT !ABC !XYZ"** will be renamed into **"PUT !XYZ !ABC"**.

GENERALIZED RENAMING FILES

The above format for renaming files is the one used in the *CADP* toolbox for renaming labels selectively. The ''rename_1'' library of *OPEN/CAESAR* supports this format, while providing additional flexibility, in several directions:

- The ''rename_1'' library allows to parameterize the definition of **<header>**, which can therefore be different from **"rename"**. The value of **<header>** is determined by a regular expression passed as parameter to function **CAESAR_CREATE_RENAME_1()** (see below).
- The ''rename_1'' library also allows files without **<header>**. This special case is obtained by giving the **NULL** value to the corresponding parameter in function **CAESAR_CREATE_RENAME_1()**. In such case, the **<axiom>** of the grammar is simply defined as **<renaming-list>**, which does not change the semantics.
- The ''rename_1'' library allows four possibilities for renaming a character string S according to a **<left-regexp>** R: *total matching* (S should match R entirely, otherwise renaming does not

occur), *single partial matching* (renaming is performed for the first sub-string of S that matches R), *multiple partial matching* (renaming is performed for every sub-string of S that matches R), or *gate matching* (renaming is only performed for the first word of S and only if this first word matches R, the remaining part of S being unchanged in any case; the first word of S is the sub-string starting at the beginning of S and ending at the first character **!**, **?**, **(**, space, or tabulation, if any, or at the end of S otherwise; in the case where S is a LOTOS label, the first word of S denotes a LOTOS gate identifier). The choice between these possibilities is determined by the value of an actual parameter passed to function **CAESAR_CREATE_RENAME_1()**. The renaming files used in the *CADP* toolbox follow the total match semantics.

DESCRIPTION

The ''rename_1'' library allows to process one or several renaming files at the same time. Each renaming file is read, parsed and checked; if correct, its contents are stored (under a compiled form) in a data structure called ''renaming object''. Afterwards, the renaming file will only be handled through a pointer to its corresponding renaming object.

FEATURES

.....

CAESAR_TYPE_RENAME_1

```
typedef CAESAR_TYPE_ABSTRACT (...) CAESAR_TYPE_RENAME_1;
```

This type denotes a pointer to the concrete representation of a renaming object, which is supposed to be ''opaque''.

.....

CAESAR_CREATE_RENAME_1

```
void CAESAR_CREATE_RENAME_1 (CAESAR_R, CAESAR_PATHNAME, CAESAR_HEADER,
                             CAESAR_KIND)
   CAESAR_TYPE_RENAME_1 *CAESAR_R;
   CAESAR_TYPE_STRING CAESAR_PATHNAME;
   CAESAR_TYPE_STRING CAESAR_HEADER;
   CAESAR_TYPE_NATURAL CAESAR_KIND;
   { ... }
```

This procedure allocates a renaming object using **CAESAR_CREATE()** and assigns its address to ***CAESAR_R**. If the allocation fails, the **NULL** value is assigned to ***CAESAR_R**.

Note: because **CAESAR_TYPE_RENAME_1** is a pointer type, any variable **CAESAR_R** of type **CAESAR_TYPE_RENAME_1** must be allocated before used, for instance using:

```
CAESAR_CREATE_RENAME_1 (&CAESAR_R, ...);
```

The actual value of the formal parameter **CAESAR_PATHNAME** denotes a character string containing the file name of the renaming file. If the file name has a suffix (see above for a discussion about suffixes for renaming files), this suffix should be part of the character string **CAESAR_PATHNAME** (no suffix will be added implicitly). The renaming file referred to by **CAESAR_PATHNAME** should exist and be readable.

As a special case, if **CAESAR_PATHNAME** is equal to **NULL**, then the renaming file will be read from the standard input.

The actual value of the formal parameter **CAESAR_HEADER** denotes a character string containing a regular expression according to the definition given in the manual page of the POSIX **regexp(LOCAL)** command. This regular expression specifies the **<header>** that must occur at the first line of the renaming file.

As a special case, if **CAESAR_HEADER** is equal to **NULL**, then the renaming file should have no header line.

The actual value of the formal parameter **CAESAR_KIND** should be equal to 0 if total matching is desired, to 1 if multiple partial matching is desired, to 2 if single partial matching is desired, or to 3 if gate matching is desired (see above for a definition of these terms).

The renaming file is parsed: its **<left-regexp>**'s and **<right-regexp>**'s are analyzed and stored (under a compiled form) into the renaming object ∗**CAESAR_R**.

So doing, various error conditions may occur: the renaming file can not be open; it is empty, or the first line does not match the header specified by **CAESAR_HEADER**; **CAESAR_HEADER** is not a valid regular expression; the renaming file has syntax errors; it contains some **<left-regexp>** (resp. some **right-regexp>** that is not a valid regular expression (resp. replacement); etc. In such case, a detailed error message is displayed using the **CAESAR_WARNING()** procedure, and the **NULL** value is assigned to ∗**CAESAR_R**.

.....

CAESAR_HEADER_RENAME_1

```
#define CAESAR_HEADER_RENAME_1 "rename"
```

This macro-definition returns the standard header for the renaming files used in the *CADP* toolbox (see above). In such case, this macro-definition should be used as an actual value for parameter **CAESAR_HEADER** when invoking function **CAESAR_CREATE_RENAME_1**.

.....

CAESAR_DELETE_RENAME_1

```
void CAESAR_DELETE_RENAME_1 (CAESAR_R)
   CAESAR_TYPE_RENAME_1 *CAESAR_R;
   { ... }
```

This procedure frees the memory space corresponding to the renaming object pointed to by ∗**CAESAR_R** using **CAESAR_DELETE()**. Afterwards, the **NULL** value is assigned to ∗**CAESAR_R**.

.....

CAESAR_APPLY_RENAME_1

```
CAESAR_TYPE_STRING CAESAR_APPLY_RENAME_1 (CAESAR_R, CAESAR_S)
   CAESAR_TYPE_RENAME_1 CAESAR_R;
   CAESAR_TYPE_STRING CAESAR_S;
   { ... }
```

This function attempts to rename the character string **CAESAR_S** according to the renaming object pointed to by **CAESAR_R**.

If renaming succeeds, this function returns a character string containing the renamed string. The address of this character string is left unspecified, but it is assumed to be different from **CAESAR_S**.

If renaming fails, this function returns **CAESAR_S**. It is therefore possible to decide whether renaming succeeded or not, by comparing the result to the second parameter passed to **CAESAR_APPLY_RENAME_1**.

Note: in any case, the contents of **CAESAR_S** will not be modified (there is no side effect).

Note: when renaming succeeds, it is not allowed to modify the character string returned by **CAESAR_APPLY_RENAME_1()** nor to free it, for instance using **free(3)**.

Note: when renaming succeeds, the contents of the character string returned by **CAESAR_APPLY_RENAME_1()** may be destroyed by a subsequent call to this function. In particular, it is forbidden to call **CAESAR_APPLY_RENAME_1()** by giving to **CAESAR_S** the value returned by a former call to **CAESAR_APPLY_RENAME_1()** for which renaming succeeded. For instance, the following call is forbidden in the general case:

```
CAESAR_APPLY_RENAME_1 (..., CAESAR_APPLY_RENAME_1 (..., ...));
```

.....

CAESAR_FORMAT_RENAME_1

```
CAESAR_TYPE_FORMAT CAESAR_FORMAT_RENAME_1 (CAESAR_R, CAESAR_FORMAT)
   CAESAR_TYPE_RENAME_1 CAESAR_R;
   CAESAR_TYPE_FORMAT CAESAR_FORMAT;
   { ... }
```

This function allows to control the format under which the renaming object pointed to by **CAESAR_R** will be printed by the procedure **CAESAR_PRINT_RENAME_1()** (see below). Currently, the following formats are available:

- With format 0, information about the renaming object is displayed such as: the pathname of the corresponding renaming file, the header (if any), the number of patterns, the list of left and right patterns, etc.
- (no other format available yet).

By default, the current format of each renaming object is initialized to 0.

When called with **CAESAR_FORMAT** between 0 and 0, this fonction sets the current format of **CAESAR_R** 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_R** 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_R**. If **CAESAR_FORMAT** is equal to the constant **CAESAR_MAXIMAL_FORMAT**, the result is the maximal format value (i.e., 0). In all other cases, the effect of this function is undefined.

.....

CAESAR_MAX_FORMAT_RENAME_1

```
CAESAR_TYPE_FORMAT CAESAR_MAX_FORMAT_RENAME_1 ()
   { ... }
```

Caution! This function is deprecated. It should no longer be used, as it might be removed from future versions of the *OPEN/CAESAR*. Use function **CAESAR_FORMAT_RENAME_1()** instead, called with

argument **CAESAR_MAXIMAL_FORMAT**.

This function returns the maximal format value available for printing renaming objects.

.....

CAESAR_PRINT_RENAME_1

```
void CAESAR_PRINT_RENAME_1 (CAESAR_FILE, CAESAR_R)
   CAESAR_TYPE_FILE CAESAR_FILE;
   CAESAR_TYPE_RENAME_1 CAESAR_R;
   { ... }
```

This procedure prints to file **CAESAR_FILE** a text containing information about the renaming object pointed to by **CAESAR_R**. The nature of the information is determined by the current format of the renaming object pointed to by **CAESAR_R**.

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

.....

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