NAME

caesar_hide_1 – the ‘‘hide_1’’ library of OPEN/CAESAR

PURPOSE

The ‘‘hide_1’’ library provides primitives for processing ‘‘hiding files’’. These files specify which labels of a graph should be hidden (i.e., renamed into the internal action ‘‘i’’, or tau) using a set of regular expressions.

USAGE

The ‘‘hide_1’’ library consists of:

- a predefined header file **caesar_hide_1.h**;
- the precompiled library file **libcaesar.a**, which implements the features described in **caesar_hide_1.h**.

Note: The ‘‘hide_1’’ library is a software layer built above the primitives offered by the ‘‘standard’’ library.

HIDING FILES

In this section, we define the format of hiding files, as they are used in the *CADP* toolbox. The next sections will explain how the ‘‘hide_1’’ library of *OPEN/CAESAR* supports (and extends) this format.

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

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

```
<axiom> ::= <blanks> <positive-header> <blanks> \n <pattern-list>
          | <blanks> <negative-header> <blanks> \n <pattern-list>

<positive-header> ::= hide

<negative-header> ::= hide all but

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

<pattern> ::= <blanks> <regexp> <blanks> \n
            | <blanks> "<regexp>" <blanks> \n
```

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 recognized by the POSIX function **isspace()**; they are always skipped and ignored;
- **<empty>** denotes the empty sequence;
- **<regexp>** is a character string specifying a regular expression according to the definition given in the **regexp**(LOCAL) manual page. The **<regexp>** may be enclosed between double quotes, which will be removed and ignored.

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

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

Semantically, a hiding file behaves as a predicate that decides whether a character string (presumably representing the label of a transition) belongs to a set of patterns. Depending on the first line of the hiding file (positive or negative header), a label will be hidden (i.e., renamed into the internal action ''i'', often referred to as ''tau'') if recognized (or vice-versa). More precisely, the effect of a hiding file F is defined as follows:

- If the first line of F is equal to **"hide"**, then any character string S that matches one (or more) **<regexp>**(s) contained in F will be hidden; if S matches no **<regexp>**, it will be kept unchanged.
- If the first line of F is equal to **"hide all but"**, then any character string S that matches no **<regexp>** contained in F will be hidden; if S matches one (or more) **<regexp>**(s), it will be kept unchanged.

For instance, the following file:

```
hide
GET
"ACK !false .*"
```

will hide all character strings equal to **"GET"** or prefixed with **"ACK !false "**. Similarly, the following file:

```
hide all but
"PUT .*"
```

will hide all character strings except those prefixed with **"PUT "**.

GENERALIZED HIDING FILES

The above format for hiding files is the one used in the *CADP* toolbox for hiding labels selectively. The ''hide_1'' library of *OPEN/CAESAR* supports this format, while providing additional flexibility, in several directions:

- The ''hide_1'' library does not hide labels itself: it simply implements the predicate function that determines whether a character string (representing a label) matches or not some of the **<regexp>**'s contained in a file F. On the basis of this information, the *OPEN/CAESAR* programmer is free to hide the label, or perform any other action (for instance, distinguish between input and output labels).
- The ''hide_1'' library allows to parameterize the definition of **<positive-header>** and **<negative-header>**. These symbols can be different from **"hide"** and **"hide all but"**; for instance, they could be replaced by **"input"** and **"output"**. The values of **<positive-header>** and **<negative-header>** are determined by two regular expressions passed as parameters to function **CAESAR_CREATE_HIDE_1()** (see below).
- The ''hide_1'' library also allows files without **<positive-header>** nor **<negative-header>**. This special case is obtained by giving the **NULL** value to the corresponding parameters in function **CAESAR_CREATE_HIDE_1()**. In such case, the **<axiom>** of the grammar is simply defined as **<pattern-list>**. A character string S will be recognized if it matches one of the **<regexp>**'s contained in the file.

- The ‘‘hide_1’’ library allows three possibilities for deciding whether a character string S matches a **<regexp>** R: *total matching* (S should match R entirely), *partial matching* (S should contain a sub-string that matches R), or *gate matching* (the first word of S, should match R entirely, the remaining part of S being ignored; 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_HIDE_1()**. The hiding files used in the *CADP* toolbox follow the total match semantics.

DESCRIPTION

The ‘‘hide_1’’ library allows to process one or several hiding files at the same time. Each hiding file is read, parsed and checked; if correct, its contents are stored (under a compiled form) in a data structure called ‘‘hiding object’’. Afterwards, the hiding file will only be handled through a pointer to its corresponding hiding object.

FEATURES

.....

CAESAR_TYPE_HIDE_1

```
typedef CAESAR_TYPE_ABSTRACT (...) CAESAR_TYPE_HIDE_1;
```

This type denotes a pointer to the concrete representation of a hiding object, which is supposed to be ‘‘opaque’’.

.....

CAESAR_CREATE_HIDE_1

```
void CAESAR_CREATE_HIDE_1 (CAESAR_H, CAESAR_PATHNAME, CAESAR_POSITIVE_HEADER,
                           CAESAR_NEGATIVE_HEADER, CAESAR_KIND)
   CAESAR_TYPE_HIDE_1 *CAESAR_H;
   CAESAR_TYPE_STRING CAESAR_PATHNAME;
   CAESAR_TYPE_STRING CAESAR_POSITIVE_HEADER;
   CAESAR_TYPE_STRING CAESAR_NEGATIVE_HEADER;
   CAESAR_TYPE_NATURAL CAESAR_KIND;
   { ... }
```

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

Note: because **CAESAR_TYPE_HIDE_1** is a pointer type, any variable **CAESAR_H** of type **CAESAR_TYPE_HIDE_1** must be allocated before used, for instance using:

```
CAESAR_CREATE_HIDE_1 (&CAESAR_H, ...);
```

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

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

The actual value of the formal parameter **CAESAR_POSITIVE_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 **<positive-header>** that may occur at the first line of the hiding file.

The actual value of the formal parameter **CAESAR_NEGATIVE_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 **<negative-header>** that may occur at the first line of the hiding file.

As a special case, if both **CAESAR_POSITIVE_HEADER** and **CAESAR_NEGATIVE_HEADER** are equal to **NULL**, then the hiding file should have no header line. Otherwise, **CAESAR_POSITIVE_HEADER** and **CAESAR_NEGATIVE_HEADER** should both be different from **NULL**.

Note: if the regular expressions **CAESAR_POSITIVE_HEADER** and **CAESAR_NEGATIVE_HEADER** are not exclusive (i.e., there exists at least one character string that matches both regular expressions), then the effect is undefined.

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

The hiding file is parsed: its **<regexp>**'s are analyzed and stored (under a compiled form) into the hiding object ∗**CAESAR_H**.

So doing, various error conditions may occur: the hiding file can not be open; it is empty, or the first line matches neither the header specified by **CAESAR_POSITIVE_HEADER** nor the one specified by **CAESAR_NEGATIVE_HEADER**; **CAESAR_POSITIVE_HEADER** (resp. **CAESAR_NEGATIVE_HEADER**) is not a valid regular expression; the hiding file has syntax errors; it contains some **<regexp>** that is not a valid regular expression; etc. In such case, a detailed error message is displayed using the **CAESAR_WARNING()** procedure, and the **NULL** value is assigned to ∗**CAESAR_H**.

.....

CAESAR_POSITIVE_HEADER_HIDE_1

```
#define CAESAR_POSITIVE_HEADER_HIDE_1 "hide"
```

This macro-definition returns the standard positive header for the hiding files used in the *CADP* toolbox (see above). In such case, this macro-definition should be used as an actual value for parameter **CAESAR_POSITIVE_HEADER** when invoking function **CAESAR_CREATE_HIDE_1**.

.....

CAESAR_NEGATIVE_HEADER_HIDE_1

```
#define CAESAR_NEGATIVE_HEADER_HIDE_1 "hide[ \t][ \t]*all[ \t][ \t]*but"
```

This macro-definition returns the standard negative header for the hiding files used in the *CADP* toolbox (see above). In such case, this macro-definition should be used as an actual value for parameter **CAESAR_NEGATIVE_HEADER** when invoking function **CAESAR_CREATE_HIDE_1**.

.....

CAESAR_DELETE_HIDE_1

```
void CAESAR_DELETE_HIDE_1 (CAESAR_H)
```

```
CAESAR_TYPE_HIDE_1 *CAESAR_H;
{ ... }
```

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

.....

CAESAR_TEST_HIDE_1

```
CAESAR_TYPE_BOOLEAN CAESAR_TEST_HIDE_1 (CAESAR_H, CAESAR_S)
   CAESAR_TYPE_HIDE_1 CAESAR_H;
   CAESAR_TYPE_STRING CAESAR_S;
   { ... }
```

This function returns **CAESAR_TRUE** if the character string **CAESAR_S** is recognized by the hiding object pointed to by **CAESAR_H** (according to the semantics defined above), or **CAESAR_FALSE** otherwise.

.....

CAESAR_FORMAT_HIDE_1

```
CAESAR_TYPE_FORMAT CAESAR_FORMAT_HIDE_1 (CAESAR_H, CAESAR_FORMAT)
   CAESAR_TYPE_HIDE_1 CAESAR_H;
   CAESAR_TYPE_FORMAT CAESAR_FORMAT;
   { ... }
```

This function allows to control the format under which the hiding object pointed to by **CAESAR_H** will be printed by the procedure **CAESAR_PRINT_HIDE_1()** (see below). Currently, the following format is available:

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

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

When called with **CAESAR_FORMAT** between 0 and 0, this fonction sets the current format of **CAESAR_H** 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_H** 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_H**. 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_HIDE_1

```
CAESAR_TYPE_FORMAT CAESAR_MAX_FORMAT_HIDE_1 ()
   { ... }
```

Caution! This function is deprecated. It should no longer be used, as itmight be removed from future versions of the *OPEN/CAESAR*. Use function **CAESAR_FORMAT_HIDE_1()** instead, called with argument **CAESAR_MAXIMAL_FORMAT**.

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

.....

CAESAR_PRINT_HIDE_1

```
void CAESAR_PRINT_HIDE_1 (CAESAR_FILE, CAESAR_H)
   CAESAR_TYPE_FILE CAESAR_FILE;
   CAESAR_TYPE_HIDE_1 CAESAR_H;
   { ... }
```

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

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