NAME

bes, BES – text file format for Boolean Equation Systems

DESCRIPTION

The acronym *BES* stands for *B*oolean *E*quation *S*ystem. It is a mathematical formalism used to solve verification problems, such as model checking and equivalence checking.

In CADP, BES is also a text file format to represent Boolean Equation Systems.

BOOLEAN EQUATION SYSTEMS

A BES is a non-empty set of equation blocks. Each block is a (possibly empty) set of equations. Each equation contains, on its left-hand side, a boolean variable, and, on its right-hand side, a boolean formula built using boolean variables (noted *X*, *X'*, etc.), constants (**`true`**, **`false`**), and binary operators of disjunction (**`or`**) and conjunction (**`and`**).

Without loss of generality, it is assumed that each boolean formula is either purely disjunctive (i.e., contains only **`or`** operators), or purely conjunctive (i.e., contains only **`and`** operators). Formulas containing both operators can be easily eliminated by introducing auxiliary variables. The formula **`false`** (resp. **`true`**) is considered as a disjunctive (resp. conjunctive) formula.

A boolean variable is said to be defined by an equation if this variable occurs on the left-hand side of this equation. A boolean variable is said to be defined in a block if it is defined by an equation of this block.

Each boolean variable occurring on the right-hand side of an equation in a BES must be defined by some equation of this BES (but not necessarily in the same block).

Multiple definitions of a variable are forbidden, i.e., a variable is defined by at most one equation.

Recursive definitions of a variable are allowed (e.g., a variable occurs both on the left- and right-hand side of the same equation).

A boolean variable defined by an equation is disjunctive (resp. conjunctive) if the boolean formula on the right-hand side of this equation is purely disjunctive (resp. conjunctive).

If a variable *X'* occurs on the right-hand side of the equation defining a variable *X*, then *X* (directly) depends on variable *X'*.

If a variable defined in a block *B* depends on a variable defined in a block *B'*, then block *B* (directly) depends on block *B'*.

A BES is said alternation-free if there are no cyclic dependencies between its blocks (but there can be cyclic dependencies between variables inside blocks).

Each equation block has a sign, which is equal to **`mu`** (resp. **`nu`**) if the block denotes the minimal (resp. maximal) fixed point of the functional induced by the equations of the block; see [Mat06] for a mathematical definition of this functional.

BES NOTATION

BES can be either represented in binary form (in internal memory) using the **caesar_solve**(LOCAL) library, or in textual form (in files with **.bes** extension) using the syntax described below.

LEXICAL CONVENTIONS

In a BES file, lexical tokens may be separated by any sequence of spaces, tabulations, carriage returns, newlines, vertical tabulations, form feeds (these characters are those recognized by the POSIX function **`isspace()`**), or comments, which are enclosed between **`(*`** and **`*)`**; these sequences are always skipped and ignored.

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

SYNTAX DEFINITION

The textual syntax of BES files is described by the following context-free grammar, where **`<empty>`** denotes the empty sequence of symbols and **`<nat>`** denotes a non-negative integer:

```
<axiom> ::= <block-list>

 <block-list> ::= <block>
               |  <block> <block-list>

 <block> ::= block <sign> <block-idf> <unique> <mode> is
                  <equation-list>
             end block

 <sign> ::= mu
          |  nu

 <block-idf> ::= B<nat>

 <unique> ::= <empty>
            |  unique

 <mode> ::= <empty>
          |  mode <nat>

 <equation-list> ::= <equation>
                   |  <equation> <equation-list>

 <equation> ::= <local-variable-idf> = <formula>

 <local-variable-idf> ::= X<nat>

 <global-variable-idf> ::= X<nat>_<nat>

 <formula> ::= <atomic-form>
             |  <disjunctive-form>
             |  <conjunctive-form>

 <atomic-form> ::= false
                 |  true
                 |  <local-variable-idf>
                 |  <global-variable-idf>
```

```
<disjunctive-form> ::= <atomic-form> or <atomic-form>
                     | <atomic-form> or <disjunctive-form>

<conjunctive-form> ::= <atomic-form> and <atomic-form>
                     | <atomic-form> and <conjunctive-form>
```

BLOCK IDENTIFIERS

In a BES file, each block has a unique identifier of the form **`B<nat>`**, where **`<nat>`** is called the block index. Blocks may occur in the file in any order w.r.t. their indexes, and these indexes are not necessarily contiguous (i.e., there may be ''holes'' in the block numbering). If for some index **`i`** there is no block **`Bi`** defined in the file, it is considered that the block **`Bi`** is empty.

VARIABLE IDENTIFIERS

In each equation block, each boolean variable defined by an equation (i.e., occurring on the left-hand side of an equation) has an identifier of the form **`X<nat>`**, which is unique in the block, where **`<nat>`** is called the variable index. Variables may be defined in the block in any order w.r.t. their indexes, and these indexes are not necessarily contiguous (i.e., there may be ''holes'' in the variable numbering).

Variable identifiers occurring in formulas have two forms:

- **`X<nat1>_<nat2>`** is a global identifier that denotes the variable **`X<nat1>`** defined in the block **`B<nat2>`**. Global identifiers can be used for any variable occurring on the right-hand side of an equation.

- **`X<nat>`** is a local identifier that denotes the variable **`X<nat>`** defined in the current block.

MODE CLAUSE

One can attach an optional clause **`mode <nat>`** to each equation block. Such a clause specifies that the variables defined in this equation block should be computed using the resolution algorithm named A**`<nat>`** of the **caesar_solve_1**(LOCAL) library. The value **`<nat>`** is called the resolution mode of the block. If the **`mode`** clause is absent, the resolution mode is set to 0 by default, meaning that the resolution algorithm A0 will be used.

UNIQUE CLAUSE

Since the **caesar_solve_1**(LOCAL) library does not operate globally, but locally, to compute a given variable defined in a given equation block, the same equation block may be solved several times, each time for computing a different variable of this block. Of course, caching is used to avoid recomputing variables already calculated.

It is however possible to help the solver by attaching an optional clause **`unique`** to an equation block. Such a clause indicates that the block will be solved only once, meaning that the value of a single variable defined in the block will be computed. This increases the memory performance of certain resolution algorithms. Any attempt at solving twice an equation block having the **`unique`** clause will trigger a run-time error. By default, if the **`unique`** clause it absent, one assumes that several variables of the equation block will need to be computed.

BES FILE EXAMPLE

An example of a boolean equation system containing two equation blocks is shown below:

```
block nu B0 is
      X0 = X1 and X2
      X1 = X0 or X1 or X2
      X2 = X0_1 and X3
      X3 = X1 or X4
      X4 = true
   end block

   block mu B1 is
      X0 = X1 or X2
      X1 = false
      X2 = X2 and X3
      X3 = X0 or X1 or X3
   end block
```

The global variable `X0_1`, which is present on the right-hand side of the equation defining variable `X2` in block `B0`, references the variable `X0` defined in block `B1`.

HOW TO CREATE A BES FILE

One can produce a BES file manually, using a text editor.

BES files can also be generated using option **-diag** of the tool **bes_solve**(LOCAL), or using option **-bes** of the tools **bes_solve**(LOCAL), **bisimulator**(LOCAL), **evaluator3**(LOCAL), and **evaluator4**(LOCAL).

HOW TO READ A BES FILE

BES files can be read and resolved using the **bes_solve**(LOCAL) tool. They can also be loaded in memory in binary internal form using the `CAESAR_READ_SOLVE_1()` procedure of the **caesar_solve**(LOCAL) library.

BIBLIOGRAPHY

[Mat06] R. Mateescu. "CAESAR_SOLVE: A Generic Library for On-the-Fly Resolution of Alternation-Free Boolean Equation Systems." Springer International Journal on Software Tools for Technology Transfer (STTT), v. 8, no. 1, p. 37-56, 2006. Full version available as INRIA Research Report RR-5948. Available from http://cadp.inria.fr/publications/Mateescu-06-a.html

SEE ALSO

bes_solve(LOCAL), **bisimulator**(LOCAL), **caesar_solve_1**(LOCAL), **evaluator**(LOCAL), **evaluator3**(LOCAL), **evaluator4**(LOCAL), **rbc**(LOCAL), **scrutator**(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 any bug to cadp@inria.fr