UmSat-Documentation

Daniel Andrén, Lars Hellström, Klas Markström

1. Usage and options

The stand-alone solver of the UmSat suite is called UmSat. It take number of basic options and reads one or several constraint plain text files containing constraints. Several files can contain constraints of the same type.

Usage

`UmSat` [*options...*] *filename* [*filename2 ...*]

Options

`-t` *seconds*	UmSat runs atmost the given time in seconds and then stops
`-a`	UmSat will find all solutions of the problem
`-c`	UmSat will print the number of solutions of the problem
`-s`	Run in sieve-mode. Output as follows: “SAT” : The problem is satisfiable “UnSAT” : The problem is not satisfiable “Time” : Printed if -t is used and the solver runs out of time before the problem has been determined.

2. Data Format

Yet to come

3. The UmSat Tcl extension

3.1. Creating and destroying solvers. The extension defines only one command: [command]solver—solver—, which creates a SAT-solver object and a command for interacting with that solver. The syntax is

—solver— command name

where command name is the name to give the command for the new SAT-solver. This solver is destroyed when its command is deleted (—rename —command name— ””—). The return value from —solver— is, as for most Tk commands, the name of the created command.

3.2. Using solvers. A solver command (i.e., a command *created by* —solver—, not —solver— iself) has the general syntax

solver subcommand arg

where the subcommand specifies the actual operation. Many operations take a list of literals as an argument. A *literal* is a nonzero integer whose absolute value is the "name" of a variable and whose sign tell whether the variable is to be negated. (The variable names should be chosen as small as possible, since they will become indices into a non-sparse vector.)

3.2.1. *Constraints.* The first group of subcommands add constraints to the solver database. They do not return anything particular.

solver —clause— litlist

This adds a CNF clause constraint; at least one of the literals must be true.

solver —atmost— limit litlist

This adds an 'atmost' constraint; at most limit (an integer) of the literals may be true simultaneously.

solver —jmf— upper litlist lower litlist

This adds a comparison pair constraint. The upper litlist must be lexicographically greater than or equal to the lower litlist, i.e., if the value of the ith literal in upper litlist is equal to the value of the ith literal in lower litlist for all $i < n$ then it may not be the case that the nth literal in upper litlist is false and the nth literal in lower litlist is true.

solver —xor— litlist

This adds a XOR clause constraint; an even number of the literals must be true.

solver —imply— literal litlist

This adds a multiimplication constraint; if any one of the literals in the litlist are true then the literal must be true as well.

3.2.2. *Solving.* The second group of subcommands exercise the actual SAT-solving capabilities of the solver object.

solver —solve— seconds

This tells the solver to seek a *model* (assignment of values to variables) that satisfies all given constraints. It returns —1— if there is such an assignment (problem is satisfiable) and —0— if there isn't (problem is unsatisfiable). The seconds argument can be used to specify a time limit after which the command returns even if it has not determined whether the problem is solvable. In that case the return code of the command is —3— (break).

solver —upl— litlist

This performs a Unit Propagation Lookahead of the given literals in the constraint database. It returns —1— if this was successful (no contradiction was detected) and —0— if it was not (a contradiction was found).

solver —model—

This returns the current "model" of the solution, i.e., the list of those literals which are set to true. The return value is not defined after the —solve— or —upl— subcommands have returned —0—.

3.2.3. *Introspection.* The third group of subcommands query more superficial parts of the solver state.

solver —cget— option

This command returns the value associated with the given option.

solver —cget -nlearnts—
solver —cget -nconstraints—
solver —cget -nassigns—
solver —cget -nvariables—

These return the number of learnt clauses, non-learnt constraints, assigned variables, and variables overall respectively.

Department of Mathematics, UmeåUniversity , SE-901 87 Umeå, Sweden