Modification of ALL – SAT solver to search verification kits in testing

I A Lyapunova\textsuperscript{1} and N A Fomenko\textsuperscript{2}

\textsuperscript{1} Southern Federal University
\textsuperscript{2} Gubkin Russian State University of Oil and Gas
To date, systems are being actively developed that imply automatic proof of software compliance with the so-called specification, i.e. requirements for it. However, in most projects, testing is still the basis of the verification process. Often this is done by people whose task is to develop tests for finding errors made by programmers in the code. Simplified tests represent a set of “input” parameters supplied to the entrance to the program, and upon completion of its work, the “output” parameters are compared with the “expected” ones according to the logic described in the specification. Testing can determine how repeated program execution with the intention of finding errors in it, but this in turn does not prove its correctness. This paper discusses the use of SAT solvers to solve the problem of finding verification kits for software testing and the development of an ALL-SAT solver modification.
The basic methods of verification tests

The main stages of SOFTWARE development and approximate distribution of introduced and detected errors

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Project creation</th>
<th>Implementation</th>
<th>Testing</th>
<th>Exploitation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Unit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

Stages of software development and information about errors

[Graph showing the distribution of errors across different stages of software development]
Automatic SOFTWARE validation presents a significant problem in the absence of strict standards for specification development and code writing. Besides however, these methods have the following significant drawbacks: slow speed, human intervention may be required and in General it is not possible to build a complete system of axioms and rules of withdrawal. Thus, software testing remains the most popular and effective SOFTWARE verification procedure for the majority of large projects where the introduction of formal methods is not possible. The testing process can be divided into 3 stages:

**Testing Process Diagram**
Module testing is a procedure of testing individual subroutines or a certain set of program functions. The implication here is that before you begin testing a program as a whole, you should test the individual small modules that make up the program. This approach allows you to control the combinatorics of testing, since the initial focus is on small modules of the program. Just do not forget about the easier debugging, testing speed and the ability to parallelize testing modules.

Testing, having obvious advantages, also faces a number of problems: for large sequential or parallel programs it is very difficult to sort through all the input data, and for dynamic structures it is impossible; the huge size of the required coverage; full coverage does not guarantee the absence of errors.
Development of ALL – SAT – solver

The search of all performing sets of Boolean formulas is widely used in such areas as checking unlimited character models, QBF-problems, Boolean optimization problems and many others.

To implement the ALL – SAT solver, the most obvious possibility is to take the algorithm of the existing SAT solver as a basis. This way is also one of the most effective.

SAT is NP-complete problem, therefore, it is extremely important for the theory of algorithms, and its solution is able to give an answer to question of equality of classes P and NP.

Therefore, the use of the SAT solver (figures 4, 5) may be useful for the solution common problems such as scheduling in educational institutions or construction rational route's (task salesman's.) For implementations multiplatform it was decided to write a program based on the use of the web-technologies (HTML, CSS, JavaScript). Thus, this PAT solver is a web page with an input field and a button that starts the algorithm for solving the problem.
An example of the script of the SAT – solver when the formula is executable
An example of the script of the SAT-solver when the formula is not feasible
Summary

The resulting solver can be run on most devices that have a browser. Specific interest is the possibility of implementing such algorithms on the GRID system [4]. During debugging and testing of the program it was found that ALL – SAT – solver works correctly and copes with its tasks [4, 5]. It can also be concluded that the resulting solver is significantly inferior in performance implemented C++ MiniSAT [4, 5], but during debugging and testing of the program it was found that the solver works correctly and copes with its tasks. Tests were also conducted on mobile devices and different versions of browsers.