**HPC-based parallel software for solving applied Boolean satisfiability problems**

V.G. Bogdanova, S.A. Gorsky, and A.A. Pashinin  
Matrosov Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia  
bvg@icc.ru, gorsky@icc.ru, apcrrol@gmail.com

**Abstract** - Boolean satisfiability is one of the fundamental problems in mathematical logic and the theory of computation. Many practical problems can be formulated as problems of Boolean satisfiability (solving a system of Boolean equations). These include the tasks of cryptography, qualitative research of binary dynamic systems, logical programming, robot action planning, and many others. NP-completeness of the Boolean satisfiability problem actualizes the development of parallel software for solving it in a high-performance computing environment. We focus on the development of specialized parallel solvers for Boolean satisfiability problems oriented to the application in the qualitative study of binary dynamic systems based on the Boolean constraint method. The Boolean constraint method uses a formal definition of the dynamical property, presented in the language of predicate logic with limited existential and universal quantifiers. Based on this definition, a model of dynamical property is formed in the form of Boolean constraints by a series of sequential formal transformations. The developed parallel solvers provide a conclusion on the feasibility of Boolean constraints, interpret this conclusion accordingly the tested property, and find, unlike similar software tools, all sets of values of Boolean variables that lead to this conclusion. An example of applying the developed solver to constructing cycles of a given length of stream ciphers based on shift registers is considered. The results confirming the effectiveness and scalability of the developed software are presented.

**Keywords** - Boolean satisfiability problem; Boolean constraints method; parallel solver; binary dynamic systems

I. INTRODUCTION

Mathematical models described in the language of Boolean equations (constraints) are widely used in various subject areas, such as pattern recognition, coding theory, cryptography, and qualitative analysis of binary dynamic systems (BDS). Reducing exhaustive problems from these areas to a Boolean satisfiability problem requires the development of a technology for constructing Boolean formulas. The computational complexity of the satisfiability problem for Boolean constraints actualizes the development of parallel software for solving it in a high-performance computational environment. We use microservice-oriented technology HPCATAMP [1] for a qualitative study of BDS based on the Boolean constraint method (BCM, [2]). The constructing Boolean formulas technology for encoding the problems in BDS qualitative analysis to the Boolean satisfiability was considered in [3]. Using BCM, a model of the dynamical property of BDS is built. This model includes constraints, and their feasibility indicates the feasibility of this property. These constraints are Boolean while studying the dynamics of BDS in a finite time interval. The Boolean model of the BDS property takes into account both the dynamic equations of the BDS and the structural characteristics of the property expressed in the language of propositional logic or predicate logic with bounded quantifiers. Thus, checking the property satisfiability is reduced to solving an NP-complexity SAT problem (or PSPACE-complex 2QBF problem, depending on the property) with subsequent solving based on data parallelism in case of the large dimension BDS. Let us call this approach the BCM-approach.

In this study, we propose a specialized parallel solver for Boolean satisfiability problems oriented for applying the BCM-approach in the qualitative analysis of BDS. Data parallelization is performed by splitting the original Boolean model. The subtasks for verifying the satisfiability of the obtained submodels are performed as independent ones on a high-performance computational cluster. The developed parallel SAT solver provides a conclusion on the feasibility of Boolean constraints, and find unlike similar software tools, all or a given number of satisfying assignments of Boolean variables. This solver is intended for high-performance computational SMP cluster. In a hybrid heterogeneous high-performance environment, a service-oriented approach is used for solving BDS qualitative analysis problems. The parallel solver is implemented as a microservice installed on dedicated multi-core computing resources. The problem statement agent with the web interface accepts the user’s request, performs splitting of the initial Boolean model at the upper level, sends out tasks to microservice-solvers, monitors their implementation, and combines the results of the work. Each microservice simultaneously determines the feasibility of a submodel. Microservices for building a Boolean model and verifying its feasibility are designed both for stand-alone and for joint work in a distributed solver [4] under the control of an active group of agents that self-organizes for the non-procedural formulation of the problem. The research described in [4] develops in the direction both for creating packages of applied microservices [1] and for implementing a composite service based on such a package. The problem statement made earlier determines the composition of this service.

The search for cycles of limited length is one of the main tasks of qualitative analysis, the computational complexity of which increases exponentially with an
increase in the dimension of the BDS state vector. This study aims to apply parallel and distributed technologies to solve the problem of finding cycles of a given length in high-dimensional BDS. The searching cycles of a given length \( k \) requires to find all solutions. The authors are aware of only sequential SAT solvers able to find all solutions. Therefore developed new parallel solver HpcAllsat is used for solving the problem of searching cycles. The testing of HpcAllsat efficiency is carried out on the example of searching the short length cycles of stream ciphers based on shift registers. The study of solving this problem based on the procedural SAT approach is given in [5].

II. RELATED WORK

In the past two decades, different approaches [6-12] was developed to simplify solving this problem. However, this problem is not fully resolved for large-scale BDS. In [7], the following areas of development of sequential attractor search algorithms were identified:

1. The method of enumeration and simulation [8]. The dimension of the BDS significantly limits the application of this method. In addition, this method is incomplete. It cannot guarantee the presence of all attractors of a given length [5].

2. The method of encoding Boolean function using BDD (Binary Decision Diagram) [9], the disadvantage of which consists in exponential growing the runtime with increasing the dimension of the BDS.

3. Algorithms using the SAT approach, in which transitions in BDS for a bounded number of steps is represented as a propositional formula [10]. The path corresponding to these transitions is found using the SAT solver. Then it is checked whether there is a closed trajectory in this path. The process is iteratively repeated for an increased number of steps. The efficiency of the algorithm mainly depends on the number of transition steps and the BDS dimension.

4. Aggregation approach [6], based on the decomposition of the structural graph of BDS into strongly coupled components (SCC). The aggregation approach uses the decomposition of the structural graph of BDS into SCC, the effectiveness of this approach substantially depends on the structure of the BDS. The SCC decomposition approach is difficult when the graph has a large amount of SCC, which usually occurs in biological networks [11]. In some cases, the divided subnets are still too large. However, for small enough subnet, the resulted attractor states of it may differ from the attractor states of the original network [12].

Parallel versions are also developed for some of the approaches listed above (for example, [6]). The aggregation approach is parallelized. However, the number of SCCs limits parallelization depth. The SAT-based algorithm [10] cannot be parallelized due to its inherent algorithm design [6]. Thus, despite active research in the field of a qualitative analysis of the BDS dynamics, the relevance of developing new and modifying existing methods for studying the behavior of high-dimensional BDS trajectories remains.

III. BCM APPROACH

Let \( X = B^n \ (B = \{0, 1\}) \) be the set of binary vectors of dimension \( n \) (BDS state space). By \( t \in T = \{1, \ldots, k\} \) we denote the discrete time (number of time steps).

For each initial state \( x^0 \in X \), let us define the trajectory \( x(t, x^0) \) as a finite sequence of states \( x^0, x^1, \ldots, x^k \) from the set \( X \). Next, we consider a synchronous autonomous nonlinear BDS in which each pair of adjacent states \( x^i, x^{i+1} \ (t \in T \) of the trajectory is connected by the relation

\[
x^i = F(x^{i-1})
\]

Here \( F : X \to X \) is a vector function of logic algebra, called the transition function. Thus for any \( x^0 \in X \), the system of Boolean equations (1) represents a model of the dynamics of the BDS trajectories behavior in the state space \( X \) on a finite time interval \( T = \{1, 2, \ldots, k\} \). Hereafter, the value of \( k \) in the definition of the set \( T \) is assumed to be a predetermined constant.

The application of the Boolean constraint method for checking the dynamical property reduces to the next scheme:

- Constructing, using the equations of dynamics (1), a function of a one-step transition \( L \),
- \( L(x^{i-1}, x^i) = \bigvee_{\alpha=1}^{n} (x^i_{\alpha} \oplus F_{\alpha}(x^{i-1})) \) [1];
- Propositional encoding of this function in the DIMACS format [13];
- The building the \( k \)-step transition function
- \( \Phi(x^0, x^1, \ldots, x^k) = \bigwedge_{i=1}^{k} L(x^{i-1}, x^i) \) [2] based on this propositional formula;
- Constructing a Boolean model of property by supplementing the function \( \Phi \) with constraints corresponding to the specification of the property;
- Determining the feasibility of these Boolean constraints using, in depending on the property, the SAT or 2QBF solver (to obtain, if needed, the constructive solutions, and post-processing results in this case).

For example (Fig. 1), the above-described scheme is used for searching cycles of a given length \( k = 3 \) based on the BCM approach for an autonomous synchronous BDS [14]. the dynamics equations of which are of the form

\[
x^i_1 = x^{i-1}_3, x^i_2 = x^{i-1}_1, x^i_3 = x^{i-1}_4, x^i_4 = x^{i-1}_1.
\]
At the first stage of BCM-approach, the first service produces the function $L$ of a one-step transition from state $x^0$ to state $x^1$ for the system (2), convert $L$ to CNF, performs Boolean encoding of the function $L$ in DIMACS format. At the second stage, the second service constructs the Boolean model by repeating function $L$ with $n$-increasing indexes of variables for $t$ from 2 to $k$ and adding cycle existence condition $R$ (Fig. 1). The indices $1, 2, \ldots, 16$ correspond to state variables, and indices $17, 18, 19, 20$ – to helper variables $y^t$ of the condition $R$ (Fig. 1) for $q = 1$. The automation technology for constructing a Boolean model of dynamic properties is described in detail in [3].

At the third stage, SAT-solver based service is used for verifying Boolean model satisfiability. Using the SAT solver, we make sure that the constraints are satisfiable, and the cyclic sequence of length $k = 3$ is found (Fig. 1).

A traditional SAT solver produces only one solution, a satisfying assignment of Boolean variables (constructive solution). All solutions SAT (AllSAT) is a variant of the propositional satisfiability problem [15]. In [15], advanced methods for solving the AllSAT-problem are considered, and sequential AllSAT solvers are described. Freely distributed versions of these solvers are available [16], in particular nbc_minisat_all-1.0.2 and bc_minisat_all-1.1.2. We use the sequential AllSAT solver nbc_minisat_all-1.0.2 if it is necessary to find all solutions when searching for all cycles of a given length $k$. However, the sequential solver can not solve this problem in an acceptable time for high-dimension BDS functioning in a long time interval $k$.  

Figure 1. Search for cycles of length $k=3$ in BDS (2) based on the BCM approach

Figure 2. The algorithm of master process

IV. PARALLEL ALLSAT SOLVER

The parallel search of all cycles of a given length is performed to solve problems with a high-dimension state vector of BDS. To organize such a search, the HpcAllsat parallel solver is developed, which is a modified version of the Hpcsat author solver [17]. The solver HpcAllsat is implemented as an MPI application in C++ and is designed to find all solutions to the SAT of high dimension on computing clusters with SMP nodes running Linux. HpcAllsat is based on the search space partitioning approach between several sequential AllSAT solvers launched in slave processes. The algorithms of the master and slave processes are presented in Fig. 2 and Fig. 3.
As test problems for comparing the two approaches while searching for cycles of a given length $k$, we used BDS, which describes the dynamics of the Trivium and Bivium shift register-based stream ciphers with a high dimension of the state vector ($n = 288$ and $n = 177$, respectively). An analysis of these ciphers is given in [5]. The cycle search time for a given length $k$ in the experiment was limited to twelve hours. For the Trivium cipher, the description of the dynamics has the form [5]:

\[ x'_i = x_{287}^{i-1} \oplus x_{286}^{i-1} \oplus x_{243}^{i-1} \oplus x_{69}^{i-1}, \]
\[ x'_i = x_{93}^{i-1} \oplus x_{92}^{i-1} \oplus x_{91}^{i-1} \oplus x_{171}^{i-1} \oplus x_{66}^{i-1}, \]
\[ x'_i = x_{177}^{i-1} \oplus x_{176}^{i-1} \oplus x_{175}^{i-1} \oplus x_{264}^{i-1} \oplus x_{62}^{i-1}, \]
\[ i \in \{2, 3, \ldots, 93, 95, \ldots, 177, 179, \ldots, 288\}, t \in \{1, \ldots, k\}. \]

Bivium, a simplified version of the Trivium cipher, has a smaller dimension of the state vector. The dynamics equations of Bivium [5] have the following form:

\[ x'_i = x_{177}^{i-1} \oplus x_{176}^{i-1} \oplus x_{175}^{i-1} \oplus x_{69}^{i-1} \oplus x_{162}^{i-1}, \]
\[ x'_i = x_{93}^{i-1} \oplus x_{92}^{i-1} \oplus x_{91}^{i-1} \oplus x_{171}^{i-1} \oplus x_{66}^{i-1}, \]
\[ x'_i = x_{i-1}^{i-1}, \]
\[ i \in \{2, 3, \ldots, 93, 95, \ldots, 177\}, t \in \{1, \ldots, k\}. \]

The same sets of cycles of length $k$ are found using both the BCM approach and the BNS solver. There are the following solutions:

- One cycle of length $k = \{1, 10, 11, 15\}$, two cycles of length $k = 12$, and 21 cycles of length $k = 3$ for Trivium;
- One cycle of length $k = 1$ and five cycles of length $k = 3$ for Bivium.

Fig. 4 shows the results (increasing total solution time) obtained for sequentially searching for cycles of length $k$ based on the BCM approach, compared with the algorithmic one. Based on the algorithmic approach, it is not possible to exceed $k = 42$ for Bivium and $k = 29$ for Trivium within the time allotted for solving all the problem (12 hours). Based on the BCM approach, we reached $k = 44$ for Bivium and $k = 30$ Trivium. New cycles for these values of $k$ for both chippers are not found. Repeated cycles (cycles with the same set of states) are excluded during post-processing of the results of the solution in time $O (k + K)$, where $K$ is the number of found cycles of length $k$. In comparison with runtime for solving problems, this time is insignificant. A further increase in the value of $k$ leads to an excess of the time limit for sequential solving the problem.

The second experiment was to evaluate the effectiveness of the developed by the authors of the parallel ALLSAT-solver HpcAllsat. The search for cycles is carried out for Grain80, Bivium and Trivium on the first segment of the computational cluster “Akademik V.M.

V. COMPUTATIONAL EXPERIMENT

The experiments are carried out on the high-performance computational cluster of the Irkutsk Supercomputer Center of the SB RAS [18]. The first experiment is carried out to compare the declarative BCM approach with the algorithmic approach for the sequential solving of the problem of searching for cycles of length $k$ in large-dimensional BDS. The Boolean model of the BDS property includes both the BDS dynamics equations and the specifications characterizing the property. The same solver can be used to verify the different BDS properties. Based on the algorithmic approach, each program is written to test only one. As a solver based on the algorithmic approach, we used the sequential BNS solver (Boolean Networks with Synchronous, freeware version 1.3 from February 2017 is available on the site [19]) based on the SAT approach [5, 10]. The search for cycles based on the BNS solver algorithm will be called the BNS approach in computational experiments. The structure of the solver algorithm, including an iterative cycle in which solver Minisat is launched, makes it difficult to parallelize [6]. Therefore, for an adequate comparison of declarative and algorithmic approaches, the computations in both cases were carried out sequentially. In the first experiment, the sequential solver nbc_minisat_all-1.0.2 is used as the ALLSAT solver in the BCM approach. The first experiment is performed on a PC with Intel Core i7-2670QM CPU 2.2 GHz×8.
Matrosov”, the node of which has two 16-core processors AMD Opteron 6276 “Bulldozer” /“Interlagos” with a frequency of 2.3 GHz. Description of the dynamics Grain80 has the following form [4]:

\[
x_{80} = x_{64}^{t-1} \oplus x_{61}^{t-1} \oplus x_{53}^{t-1} \oplus x_{46}^{t-1} \oplus x_{34}^{t-1} \oplus x_{29}^{t-1} \oplus \ldots \\
+ x_{22}^{t-1} \oplus x_{16}^{t-1} \oplus x_{10}^{t-1} \oplus x_{6}^{t-1} \oplus x_{1}^{t-1} \oplus x_{61}^{t} \oplus x_{53}^{t} \oplus x_{46}^{t} \oplus x_{34}^{t} \oplus x_{29}^{t} \oplus x_{22}^{t} \\
+ x_{64}^{t-1} \oplus x_{61}^{t-1} \oplus x_{10}^{t-1} \oplus x_{6}^{t-1} \oplus x_{1}^{t-1} \oplus x_{61}^{t} \oplus x_{53}^{t} \oplus x_{46}^{t} \oplus x_{34}^{t} \oplus x_{29}^{t} \oplus x_{22}^{t} \\
+ x_{64}^{t-1} \oplus x_{61}^{t-1} \oplus x_{22}^{t-1} \oplus x_{16}^{t-1} \oplus x_{64}^{t} \oplus x_{61}^{t} \oplus x_{53}^{t} \oplus x_{46}^{t} \oplus x_{34}^{t} \oplus x_{29}^{t} \oplus x_{22}^{t} \\
+ x_{64}^{t-1} \oplus x_{61}^{t-1} \oplus x_{22}^{t-1} \oplus x_{16}^{t-1} \oplus x_{64}^{t} \oplus x_{61}^{t} \oplus x_{53}^{t} \oplus x_{46}^{t} \oplus x_{34}^{t} \oplus x_{29}^{t} \oplus x_{22}^{t} \\
+ x_{64}^{t-1} \oplus x_{61}^{t-1} \oplus x_{22}^{t-1} \oplus x_{16}^{t-1} \oplus x_{64}^{t} \oplus x_{61}^{t} \oplus x_{53}^{t} \oplus x_{46}^{t} \oplus x_{34}^{t} \oplus x_{29}^{t} \oplus x_{22}^{t} \\
+ x_{64}^{t-1} \oplus x_{61}^{t-1} \oplus x_{22}^{t-1} \oplus x_{16}^{t-1} \oplus x_{64}^{t} \oplus x_{61}^{t} \oplus x_{53}^{t} \oplus x_{46}^{t} \oplus x_{34}^{t} \oplus x_{29}^{t} \oplus x_{22}^{t} \\
+x_{i}^{t} = x_{i+1}^{t}, i \in \{1, 2, \ldots, 79\}.
\]

The runtime (T), speedup (S) and efficiency (E) are calculated using the formulas:

\[
S(p) = \frac{T_{32}}{T_{p}}, E(p) = \frac{S(p)}{p} = \frac{1}{N} \sum_{i=1}^{N} T_i(p),
\]

\[
S^{av}(p) = \frac{1}{N} \sum_{i=1}^{N} S_i(p), E^{av}(p) = \frac{1}{N} \sum_{i=1}^{N} E_i(p)
\]

where N is a number of test tasks, T_{32} and T_p are the time to solve the problem on, respectively, the 32 and p processor cores; T^{av}, S^{av} and E^{av} are the average time and speedup to solve one problem, and the average efficiency of a load of processor cores, accordingly. The 32 processor cores are taken as the basic unit since this value is the minimum of cores number, on which the most difficult test tasks could be solved within the limited time interval. A similar approach to the calculation of these characteristics is described in [20]. The results of evaluating the runtime with an increase in the number of processor cores are shown in Fig. 5. The decreasing runtime is close to linear for all test problems. Results of the average speedup and efficiency for solver HpcAllsat are shown in Fig. 6. Overall, the results of testing a parallel solver showed speedup close to linear, and efficiency lies within acceptable limits.

In the third experiment, a distributed solution is tested using a composite service to search for cycles of a given length k. The nodes of three computing clusters and the cloud resource [21] on which the service is installed are used as a hybrid environment (Fig. 7). In the experiment, the value “4” is set in the Split field for preliminary splitting into 16 submodels. Though the heterogeneous nature of the computational environment affects some decrease in efficiency (as a rule, accompanying a significant increase in nodes) (Fig. 8), the runtime is significant decreases with increasing the number of processor cores (Fig. 9). Experiments show that in parallel solving the problem on the SMP cluster, the k cycle length can be further increased by increasing the number of processor cores. The presented approach ensures that all cycles of a given length k are found in the state space of stream ciphers based on shift registers. As noted in [5], taking into account the fact that the state spaces in the problems being solved have dimensions 2^{268} and 2^{177}, the problem of finding all cycles of a given length is not
trivial. The proposed declarative approach is also applicable to testing other dynamic properties of BDS [2] including BDS with control actions [22].

VI. CONCLUSION

Based on the BCM approach, a parallel solver is developed to verify the feasibility of the Boolean model of the investigated property of the binary dynamic system with the construction of all or a given number of constructive solutions. The parallel solver is designed to solve the problems of the qualitative analysis of binary dynamic system based on the Boolean constraint method and is focused on application in a high-performance cluster environment. This solver is implemented as a microservice. A composite service is developed for organizing two-level parallelization in solving problems of qualitative analysis in a hybrid environment. Computational experiments are carried out on the example of studying the properties of the presence of a cycle of a given length in a high-dimension binary dynamic system and confirmed the operability and effectiveness of the developed software tools. These tools can significantly increase the dimension of the state vector of the binary dynamic system and the time interval of its functioning when solving these problems in a high-performance computational environment.

ACKNOWLEDGMENT

The present investigation was supported by the Russian Foundation of Basic Research, projects no. 18-07-00596.

REFERENCES


Figure 9. The runtime of Trivium distributed solving...