Multiagent Technology for Parallel Implementation of Boolean Constraint Method for Qualitative Analysis of Binary Dynamic Systems

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

Abstract - The high computational complexity of qualitative analysis problems requires the development of software and tools for its solution using parallel computations and service-oriented access to the resources of high-performance computing environments. We offer a new implementation technology of Boolean constraints method for the qualitative research of binary dynamic system functioning on a finite time interval. According to this method, the satisfiability verification of dynamic system properties is reduced to solving the Boolean satisfiability problem or quantified Boolean formula problem. Offered technology is based on a decentralized multi-agent approach to managing the construction and study of Boolean models of dynamic properties in a distributed computational network. Microservices-based software for models constructing and studying are installed on dedicated computational nodes. The agents associated with installation nodes, run these microservices. Local agent’s knowledge base stores information about their neighborhoods. The discrete event-driven agent’s behavior model provides decentralized control based on direct peer-to-peer interactions of agents. Using this approach, we develop a multiagent system automating the constructing a dynamic property model and parallel solving the Boolean satisfiability problem and quantified Boolean formula problem. The results of computational experiments confirmed the effectiveness of the developed software based on the proposed technology.

Keywords - multiagent technology; qualitative analysis, Boolean model; quantified Boolean formula; satisfiability; parallel solver.

I. INTRODUCTION

Qualitative research helps to identify the significant, important from both a theoretical and practical point of view features of a dynamic system. The main objective of qualitative research is to analyze the behavior of the trajectories of a dynamic system to verify whether this behavior complies with the totality of constraints characterizing the property. For qualitative analysis of binary dynamic systems (BDS), we offer an approach based on the Boolean constraints method [1], in which the checking of the required dynamic property is reduced to the satisfiability verification of certain constraints on the trajectories behavior of the BDS. For BDS, whose functioning is considered on a finite time interval, such restrictions are written in the language of Boolean equations or Boolean formulas with quantifiers. The first type of constraints constitutes the Boolean satisfiability (SAT) problem [2]; the second type is associated with quantified Boolean formula (QBF) problem [3]. The following stages of the Boolean constraints method in the qualitative analysis of BDS could be singled out taking into consideration mentioned above:

- Formation of a dynamic property specification in the language of predicate logic using bounded universal and existential quantifiers.
- Conversion of a property formula with bounded quantifiers to one with universal quantifiers. Replacing the bounded quantifiers binding the variables with finite domains by equivalent without quantifiers.
- Construction of a property model as a Boolean constraint (a Boolean equation or a quantified Boolean formula), that satisfies the logical specification of the property and the equations of the system dynamics.
- Verification of satisfiability of the received Boolean constraint by the efficient SAT and QBF solvers.

In the first two stages, theoretical studies are carried out to develop specifications for the dynamic properties of BDS. The next two stages require software development. Applied software modules are implemented as independent microservices. The use of microservices allows operating the modularity properties in new conditions for the organization of distributed computing when inter-module communication is provided only through the message transfer mechanism. Multi-agent decentralized management and self-organization mechanism, namely direct interactions of agents [4], are used to organize the interconnection of microservices. The advantages of this approach are given in [4, 5]. The active group of agents of the multi-agent system (MAS), which are delegated the rights to manage the microservices, is formed for non-procedural problem
formulation using a logical inference on a distributed knowledge base (KB) of a subject area. As this base a computational model $KB = (M, P, N, In, Out, Com)$ is used, where $M, P, N$ are finite sets of, correspondently, modules, parameters and computational field (CF) nodes. Relations $In \subset P \times M$ and $Out \subset M \times P$ determine the relationship of the modules by the input and output parameters, respectively. The relation $Com \subset M \times N$ defines the topology of the CF [6]. A module $M_i \subset M$ calculates the values of the output parameters using the specified values of the input parameters. The KB is distributed the way that each agent has limited knowledge (only $In$ and $Out$ relations) of both the capabilities of other system agents and the topology of the CF as a whole. KB allow formulating a problem by specifying the given set $A^0 \subset P$ and the required set $B^0 \subset P$ parameters values. A couple of sets $T = (A^0, B^0)$ is called the non-procedural formulation of the problem on the KB model. Thus, the computational model determines the rules for the application and combination of modules in the process of automatic $T = (A^0, B^0)$ problem solving. The instrumental support of the offered technology is the HPCSOMAS (High Performance Computing Service-Oriented Multi-Agent System) framework 3.0 [7], which automates the creation and functioning of agents based on computational microservices. Based on HPCSOMAS agent implementation involves the stage of its local KB creation, which contains the information about the agent’s predecessors and successors defined by $In$ and $Out$ relations. Thus, the distributed KB offers only direct pair interactions between agents to form an active agents group ($Ag^*$) required for solving the $T$ problem. The agent behavior is based on a discrete event-driven model FSMwVW (Finite State Machine with Variables and Works) described in detail in [7]. This model supports agent coordination based on decentralized control “on the readiness of input data” (event-driven control). Offered approach without a central control node increases the fault tolerance and scalability of calculations. According to this approach, to implement the Boolean constraints method [1] we used HPCSOMAS for developing the MAS for the micro-services based qualitative analysis of binary dynamic system (MSQABDS).

II. RELATED WORKS

The existing software for qualitative analysis of BDS (in particular, Boolean networks [8-11]) have some systemic disadvantages, namely: these tools are limited by the complexity of the Boolean model and the format of its presentation; they require programming skills from subject specialists, since they are often used only as platform-specific command line tools. In [11], the drawbacks of the simulation, binary decision diagram (BDD)- and SAT-based algorithms are noted, associated respectively with the complexity of finding all attractors, limitation of the Boolean network dimension and the impossibility of parallelizing. The last disadvantage is explained by the fact that the SAT approach is used iteratively inside the attractor search algorithm to solve the auxiliary problem of finding a path of length $k$ [8]. A parallel algorithm represented in [11] is based on the partitioning of the state transition graph. However, this algorithm does not exceed the algorithm [8] efficiency for searching attractors on classic examples of Boolean models of gene regulatory networks (GRN). In general, in existing publications, the focus of researchers is on finding attractors (equilibrium states and cycles of a given length) for autonomous BDS [8–11], while the important for dynamics analysis problem of checking the isolation property of an attractor is not considered. It should be noted that methods and means of qualitative analysis of reachability type properties are needed for a complete image of the dynamics of the functioning of autonomous nonlinear BDS. In [12], reachability property is defined as a classical one studied in a qualitative analysis of the behavior of BDS trajectories; however, verification of this property is performed only for linear systems, and the reachability property verification remains an open question for a nonlinear BDS.

The approach we propose, in contrast to the procedural one, used in [8-11], is declarative and oriented on the study of various dynamic properties. We developed not an algorithm for checking the satisfiability of a property, but a mathematical model in the form of Boolean constraints (SAT or QBF problem) that takes into account both the specification of the property and the equations of a specific object dynamics. The results obtained during the solving of this system using efficient SAT and QBF solvers determine the property satisfiability. This approach provides data parallelism by splitting the source property Boolean model. According to this approach, the aimed MSQABDS was created.

III. ARCHITECTURE OF MSQABDS

The architecture of the MSQABDS, reflecting the components composition and their interconnection by data, is shown in Fig. 1. The MSQABDS consists of two main subsystems: Boolean modeling and solving Boolean

![Figure 1. MSQABDS architecture](image-url)
equations. The HPCSOMAS automatizes Tomcat based microservices creation for all subsystems modules, and agents, which are delegated the rights to manage the microservices. Analysis of the solving of some problems of studying the behavior of trajectories for autonomous BDS [6], non-autonomous significantly nonlinear BDS [7] and BDS with the additive perturbations [13] based on the Boolean constraints method allowed to distinguish the following stages of constructing a property model:

- Constraints creation using the dynamics description (CCDD module);
- Constraints creation using the logical property specification (CCPS module).

These modules are implemented using object-oriented technology in C++ language. Different CCPS modules correspond to different properties.

A. Dynamics Description of BDS

The dynamics description for autonomous synchronous BDS is represented by a vector-matrix equation

$$x' = F(x^{i-1}),$$  \hspace{1cm} (1)

where \( x \) – is a state vector, \( x \in B^n \), \( B = \{0,1\} \), \( n \) – is the dimension of the state vector; \( t \in T = \{1,2,...,k\} \) – is discrete time (the time step number); \( F(x) \) – is a vector function of Boolean algebra which is call transition function. For every state \( x^0 \in B^n \) we determine the trajectory \( x(t, x^0) \) of the system (1) as a finite-length state sequence of \( x^0, x^1, ..., x^k \) from the \( B^n \) set. It is known, that (1) for \( t = k \) is equal to the following Boolean equation

$$\Phi_k(x^0, x^1, ..., x^k) = \bigoplus_i^n \bigoplus_m^n (x'_i \oplus F_i(x^{i-1})) = 0,$$  \hspace{1cm} (2)

where \( x'_i \) – \( i \)-th component of \( x' \) vector; \( \oplus \) – modulo-2 addition; \( F_i \) – \( i \)-th vector \( F \) component, function \( \Phi_k(x^0, x^1, ..., x^k) \) denotes the \( k \)-step transition for \( t = 1, 2, ..., k \). For \( k = 1 \) the equation (2) acquire the form for one-step transition:

$$\Phi_1(x^0, x^1) = \bigoplus_i^n (x'_i \oplus F_i(x^0)) = 0.$$  \hspace{1cm} (3)

Obtained in correspondently with (3) the Boolean one-step transition formula is the basis for constructing a Boolean model of a dynamic property. This formula is used afterward to construct the \( k \)-transition (2).

B. Specification of Dynamical Property

At present, specifications and corresponding Boolean formulas are developed (the mathematical justification for the derivation of which is given in [1]) to identify branching of a state, equilibrium states, closed trajectories (cycles) and for the following dynamic properties of reachability type:

- the main reachability property, the security property, the simultaneous reachability property;
- the reachability property with phase constraints, the attraction property;
- the connectivity property of the goal set;
- the property of total reachability of the goal set from the set of initial states.

The specification of a dynamic property is written in the language of formal logic. The main syntactic elements of a logical formula are [1]: 1) subject variables (components of vectors \( x^0, x^1, ..., x^k \), time \( t \)); 2) bounded existential and universal quantifiers; 3) logical operators \( \lor, \land; \) 4) final formulas, which are the assertions about the belonging of specific states of the set of trajectories \( x(t, x^0) \) (\( x^0 \mathbb{E} X^n \)) to the estimated sets \( X^0 \) and \( X^k \). The elements of these sets are determined by the roots of the Boolean equations \( G^i(x) = 0 \), \( G^j(x) = 0 \), \( G^m(x) = 0 \) or by the characteristic functions of these sets \( \overline{G^o} \), \( \overline{G^m} \), \( \overline{G^m} \).

C. Boolean Modeling Implementation

BDS dynamics description (1) can be formulated using the mathematical language in the Latex, MathML input formats, or represented in the “CNET” format [14] (for compatibility with other software systems). Output formats of the Boolean constraints for the property description are DIMACS [15] or QDIMACS [16]. The Boolean modeling subsystem includes BFMDE (Boolean Function Mathematical Description Editor) service (Fig. 2), which allows entering the system dynamics equations (1) in mathematical language. BFMDE is implemented in Java programming language and includes appropriate converters transfer these equations from the input format to the internal representation, based on which the algorithm for calculating the Boolean function is built (serial or parallel depending on the dimension of the state vector). This algorithm is used for dynamical construction a truth table according to which a Boolean function is generated in Conjunctive (or Disjunctive) Normal form (CNF or DNF). BFMDE uses the shareware FMath Javascript Equation Editor [17] for the formulas input.

D. Example of Boolean Model Creation

Let us consider an illustrative example from [18]. The BDS dynamics description is presented by the following system of equations (\( n = 3 \)):

![Figure 2. Web-interface of mathematical editor](image-url)
\[ x_i' = x_{i-1}^{-1} \land x_{i-1}' = x_{i-1}^{-1}, x_i' = \overline{x_i^{-1}}. \] (4)

Correspondently with the above steps of building a Boolean model, the CCDD module reads (4) from the input format and builds the equations for a one-step transition in the system (4) using (3).

\[
\Phi_1(x^0, x^1) = \overline{x_1} \land x_0 \land x_0 \lor x_1 \lor \overline{x_2} \land x_2 \lor x_2 \lor x_2 \lor x_3 \lor x_3 \lor x_3 = 0. \] (5)

The CCPS module generates according to \( \Phi_1 \) the Boolean equations for \( k \)-step transition (\( k=2 \)):

\[
\Phi_2(x^0, x^1, x^2) = \Phi_1(x^0, x^1) \lor x_2 \lor x_3 \lor x_3 \lor x_3 \lor x_3 \lor x_3 = 0. \] (6)

Let us verify the main reachability property formulated as follows. Any trajectory starting from the set of initial states \( X^0 \), the goal set \( X^* \) [1]. The specification of the reachability property in the language of predicate logic has the form [1]:

\[
(\forall x^0 \in X^0)(\exists x^1 \in T)(x(t, x^0) \in X^*). \]

The next logical equation correspond to this specification and the equations of dynamics (4)

\[
G^0(x^0) \lor \Phi(x^0, x^1, ..., x^k) \lor (\lor \Pi_{i=1}^k G^i(x^i)) = 0. \]

Let \( X^0 = \{x \in X : x_1 = 0\} \), \( X^* = \{x \in X : x_3 = 1\} \). Then \( G^0(x^0) = x_1, G^i(x^i) = x_3, \Phi_2 \) is given in (6). We get the Boolean model of the reachability property:

\[
x_1^0 \lor \Phi_2(x^0, x^1, ..., x^2) \lor x_1^1 \lor x_1^1 = 0. \] (7)

This Boolean model is converted to DIMACS format in which the variable is represented by its index. For \( x_i \), the index is calculated as \( i + 3t \) when \( i=1,2,3 \) and \( t=0,1,2 \). The index is negative if the variable in (7) has the logical negation. The model (7) in DIMACS format for SAT solver takes the form (disjuncts are separated by \( \lor \)):

```
p cnf 9 15
-1 0 4 -2 -3 0 -4 2 3 0 5 -1 0 5 1 0 6 2 0 -6 -2 0 7 -5 -6 0
-7 5 6 0 -8 4 0 8 4 0 -9 -5 0 9 5 0 -6 0 -9 0
```

The SAT solver answers “UNSAT” therefore the reachability property is satisfiable [1], what can be seen from the transition diagram shown in Fig. 3.

The user needs to specify the input \( (A^i) \) and output \( (B^i) \) parameters for the non-procedural formulation of the \( T \) problem on the KB model in the web-interface of the user agent (Fig. 1): \( T=(A^i\{F, f, n, k, X, X\}; B^i\{BM,R,Y,R\}) \) (Fig. 4). After launching the task, the problem solving process is performed automatically. As a result of logical inference, an active group of agents \( Ag^* = \{Ag_1, Ag_2, Ag_5, Ag_6\} \) is formed (highlighted in Fig. 4). Decentralized control of the \( Ag^* \) active group and the problem solving based on direct agents interactions are supported by FSMwVW agents behavior model described in [7]. After the task is completed, the user receives the solving result – a Boolean model and a sign of the reachability property satisfiability (\( Y_R = \) TRUE).

IV. SUBSYSTEM FOR SOLVING BOOLEAN EQUATIONS

The subsystem for solving Boolean equations includes the following components (software modules) developed by the authors: the complete search sequential solver BESS (Boolean Equations System Solver) intended to find all (or \( k \)) solutions, the parallel SAT solver Hpcsat and the QBF solver Hpcqsat. External shareware sequential SAT and QBF solvers can be connected to Hpcsat and Hpcqsat as base solvers. The description and evaluation of the effectiveness of Hpcsat are given in [19].

A. BESS Solver

The solver BESS (Boolean Equations System Solver) is implemented using the C programming language and based on the full search method, combines the constraint propagation technique with intelligent backtracking using a look ahead strategy. The basic idea of intelligent backtracking is to track the last change in the value of the variables and, in failure, return not to the nearest vertex of the search tree, but to the one to which the values leading to inconsistency of Boolean constraints were assigned. The BESS solver was used to compare the proposed declarative approach with the procedural one, based on...
which the solver BNS [8] was developed for attractors searching in the Boolean network. As an example, the Boolean models of gene regulatory networks (GRN) given in [20] were chosen. Based on the method of Boolean constraints Boolean models were constructed for checking the properties of the presence of equilibrium states in the GRN models. The results of BNS speedup compared to BNS are shown in Fig. 5. The declarative approach to solving these problems gives a speed advantage three times more than the procedural approach offered in [8]. The search for all solutions is required for solving some problems. In particular, the BESS solver has used for the finding a basin of an attractor and a domain of attraction. A parallel solution of large-scale problems is carried out using BESS as the base solver in Hpcsat. Here and below, the sequential solver, which is launched in the slave processes of the based on message passing interface (MPI) technology parallel solver is called the base solver.

### B. A Parallel Solver Hpcsat

The satisfiability verification of some dynamic properties of BDS (in particular, the properties of connectivity and total reachability) based on the method of Boolean constraints requires solving the QBF problem. The parallel solver Hpcsat developed by the authors is focused on a QBF, in the form of which a Boolean model of the dynamic property is formed. Based on the master-slave architecture Hpcsat solver uses data parallelism by splitting the original QBF. A sequential QBF solver solves the subproblems as independent subtasks in slave processes.

The parallel solver Hpcsat is based on partitioning search space approach and implemented using MPI technology in C++ language. In the master process, the building of the splitting tree of search space and the control of the slave processes functioning is performed. The master process performs initial splitting the Boolean model and controls the queue of subtasks. The slave process performs the following functions: deeper splitting submodels, their simplification, and launch of base solver finding solution for this submodel. The simplification procedure is based on the constraint propagation method, as a result of which the values of other variables can be obtained, or the solution of this submodel can be found. The splitting procedure regulates (if possible, equalizes) the complexity of the subtasks, splitting first the submodels with the largest count of variables remaining after the simplification. In the first case, after completion of the simplification process, $k$ variables remain unmarked in the submodel; this number $k$ is taken as the subtask complexity. This approach balances slave processes by maintaining the tasks number in the queue according to the free resources number. The computational experiments are conducted using resources of Irkutsk Supercomputer Center of SB RAS [21] to compare Hpcsat with a similar parallel solver HordeQBF [22]. As a base solver, DepQBF [23] is used in both Hpcsat and HordeQBF. The shareware versions of HordeQBF and DepQBF are installed on computational cluster "Akademik V.M. Matrosov" for experiments. Fig. 6 shows the results of the first experiment — the number of instances solved for a given time limit (168 hours). These 76 instances are selected from the test of QBF problems [24]. As a selection criterion, the solution time (no less than 17 seconds and no more than a given time limit) of these instances solved by the sequential solver DepQBF was used. According to these results, only the Hpcsat solved all these 76 instances. Table 1 shows the statistical results data of a second computational experiment on a set of another test of QBF problems [25] selected with the analogical criterion. The average solution time is the quotient of dividing the total time of solving selected instances to their number. The limitation on the solution time for one instance in the second experiment is 4800 seconds. The results of these two experiments show a significant

### TABLE I. COMPARISON OF HPCSAT AND HORDEQBF

<table>
<thead>
<tr>
<th>Processor cores</th>
<th>Characteristics</th>
<th>Hpcsat</th>
<th>HordeQBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Total runtime</td>
<td>2487.37</td>
<td>27798.72</td>
</tr>
<tr>
<td></td>
<td>Average runtime</td>
<td>138.19</td>
<td>1544.37</td>
</tr>
<tr>
<td></td>
<td>Number of unsolved problems</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>256</td>
<td>Total runtime</td>
<td>1049.75</td>
<td>26657.28</td>
</tr>
<tr>
<td></td>
<td>Average runtime</td>
<td>58.32</td>
<td>1480.96</td>
</tr>
<tr>
<td></td>
<td>Number of unsolved problems</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>512</td>
<td>Total runtime</td>
<td>855.86</td>
<td>29615.28</td>
</tr>
<tr>
<td></td>
<td>Average runtime</td>
<td>47.55</td>
<td>1645.29</td>
</tr>
<tr>
<td></td>
<td>Number of unsolved problems</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>
advantage of the developed Hpcqsat solver. The differences between Hpcqsat and HordeQBf are as follows: 1) HordeQBf solver is based on the “portfolio” approach, and Hpcqsat uses the search space partitioning; 2) Hpcqsat does not use learning, and HordeQBf uses the exchange of conflicting disjuncts; 3) Hpcqsat manages the queue of subtasks for the slave processes.

Computational experiments were carried out, confirming the substantial advantage of the developed parallel solver Hpcqsat based on partitioning search space approach in comparison with the analogous solver HordeQBf based on a “portfolio” approach. A parallel software implementation of the logical method for solving problems of the qualitative analysis of BDS was performed. The developed software can be used during conducting computational experiments in various subject areas, wherein the process of research discrete Boolean models arises. It should be noted that the method of Boolean constraints [1] is a quite general method of qualitative analysis of BDS on a finite time interval. This method was also applicable to autonomous BDS: systems with control inputs; systems with a memory depth of more than one; systems of a general form, for which the transition function is implicitly specified.

V. CONCLUSION

Based on the multi-agent technology for implementing the Boolean constraint method, the microservice-oriented MSQABDS system is developed. MSQABDS automates the distributed solution of qualitative analysis problems for the BDS. The integration in one model of both the description of the property and the equations of the dynamics of a specific object is the unique feature of the models created on the Boolean constraints method base. Reduction of the problem of the qualitative analysis to the solving of SAT and TQBF problems allows the data parallelism and high scalability with an increasing dimension of the problem. The developed technology is of great practical significance due to the extensive use of BDS as models in both theoretical and applied research.

ACKNOWLEDGMENT

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

REFERENCES


