The Application of SMT Solvers in the Mission Critical Software Verification

Andrey Tyugashev, Dmitrii Zheleznov
Samara State Transport University, 2V Svobody Street
Samara 443066
Russia
{a.tyugashev@samgups.ru.}
{rektorat@samgups.ru Blvd}

ABSTRACT: *We have discussed the application of SMT solvers in the Mission Critical Software verification. We have developed the Rules of verification based on Real-Time Control Algorithm's Logic. Required specification can be feasible or non-feasible on defined basis of functional control processes. In this model, the feasibility of the specification is being checked by SMT solver Z3. We have used a particular Java application through API for the SMT solver.*

Keywords: Control Logic, Functional Process, Logical Vector, Real-time Control Algorithm, Logic Programming, SMT Solver

Received: 24 February 2021, Revised 30 May 2021, Accepted 10 June 2021

DOI: 10.6025/jet/2021/12/3/86-92

1. Introduction

The modern technical object such as airplane, submarine, spacecraft or nuclear power station can be reviewed as 'system of the systems' including a lot of subsystems, actuators, sensors, other devices. Like an orchestra playing symphony, all of these devices should co-function in harmonic manner to produce a useful outcome. Each instrument must to start play at a right time. In orchestra, the conductor performs control functions. In modern complex technical complexes, the control system should provide the same functionality. The human could be involved in the process in case of automated control, or not be involved in case of automatic system. Discussing complexity level of control system we can note that in according to Ashby's Law of Requisite Variety [1], "Variety absorbs variety", so the complexity of control system should be adequate to complexity of controlled object. Control system realizes corresponding control algorithms. The 'input data' for control algorithms is so-named 'control logic'. In fact, this logic is representation of coordinated functioning of all units needed to achieve the goal of our system. The 'coordinated' word means here both semantic coordination related to physical restrictions and logic of actions, and coordination in time. The time characteristics of control logic should be adequate to speed of ongoing physical processes associated with the controlled technical complex [2-5].

The very important problem for control logic of complex technical object is evaluation of its parameters and checking if these values are correspond to existing physical and technological constraints. This issue is actual both at design stage when

the key question is feasibility of requirements, and during operation of existing technical object when we need, for example, to analyze performance. This paper is focused on timing (synchronization) parameters, and degree of use of accessible resources (level of workload/overload). The problem has an additional importance due to its straight connection to dependability/safety issues.

Today, as a rule, the control logic's evaluation is being performed by human. Unfortunately, the number of parameters which must be analyzed, for example, for modern spacecraft, can be very big and exceeds the human opportunities. The purpose of the work is to provide automation to this process. We utilize two approaches for evaluation of the control logic – use of SMT solvers, and logical programming.

Herewith, we can review potentially useful approach connected with apply of existing SMT automation tools to provide assistance to specialists responsible for control logic's evaluation [6-7]. The very popular and promising technology today is Satisfiability Modulo Theories (SMT) approach. SMT supported by a lot of commercial and free solvers such as ABSolver, Alt-Ergo, Barcelogic, MathSAT, CVC, OpenSMT, Simplify, STeP, Yices, Z3, etc. We can specify the existing constraints using smt-lib formal language, and then get the answer if the system satisfies (sat) the constraints, or not (unsat). The system even can calculate the values of the variables which provide satisfiability.

In this paper we want also to remind about power and opportunities provided by logic programming. In fact, internal logical inference machine provide us with opportunities comparable with features of modern SMT solvers. Moreover, the logic programming systems are very close by their nature to specificity of Real-Time Control Algorithm's Logic applied to checking of properties of control algorithms. So, we present the corresponding example of application of logic programming in our domain.

2. Method

2.1. Real-Time Control Logic

In previous papers [3, 6, 7] we had proposed the semantic model for real-time control algorithm. The model represents control actions by the set of following tuples:

$$RTCL = \{< f_i, t_i, \tau_i, l_i >\}, i = 1..N \quad (1)$$

f_i represents an identifier of functional process to be executed, and: t_i – time of f_i begin (non-negative integer), τ_i –its duration (non-negative integer). l_i is a 'logical vector' defining whether process should be executed. The logical vector consists of logical variables within checked values: ($\alpha_1 = 0$, $\alpha_2 = 1$, $\alpha_3 = 0$, $\alpha_4 = H$, $\alpha_5 = H$). Herewith, 1 and 0 corresponds to True and False, and '*H*' value means that execution of the process is not depends on value of this logical variable. The presence of logical variables in the model allows specifying a set of options of implementation of the algorithm (including normal and abnormal situations).

Some parameters can be specified by a known constants, some be initially unknown and stated as variables.

The constraints and requirements for real-time control logic can be specified using language of CA formal theory (calculus of real-time control algorithms) proposed by A.A. Kalentyev 3-4]. The extended version of this theory developed by author [6] – Real-Time Control Algorithm's Logic allows its usage for real control logic specification and verification.

We focus on synchronization of functional processes to be executed. The synchronization of two processes can be expressed by following operators: coincidence by begin (named CH from Russian abbreviation), coincidence by end (named CK), direct following ($\rightarrow$), time uncrossing (<>), precedence (<), strict precedence (<<), the overlap with the specified shift (H), parameterized following with the specification of the delay (3A). Table 1 unites short reference descriptions of them.

The sense of operators becomes quite clear after looking at Figure 1-6.

The operators: <. << and <> expressed 'soft' bindings where times of processes' begins and ends may vary in some intervals.

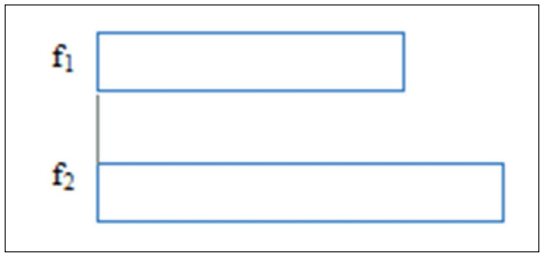

Figure 1. Coincidence ‘begin-begin’ CH

Figure 2. Coincidence ‘end-end’ CK

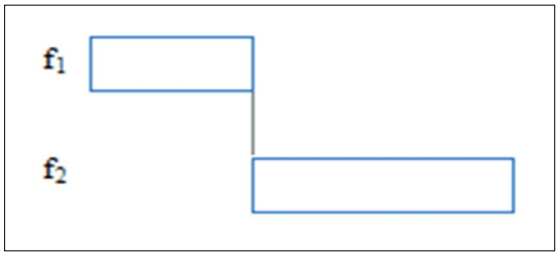

Figure 3. Direct following

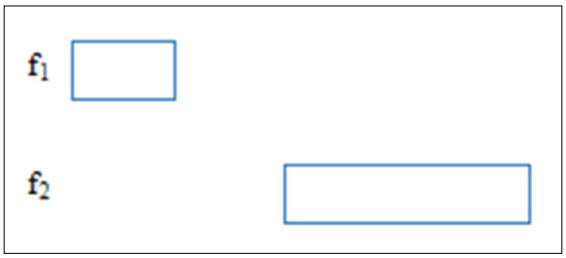

Figure 4. Strict precedence

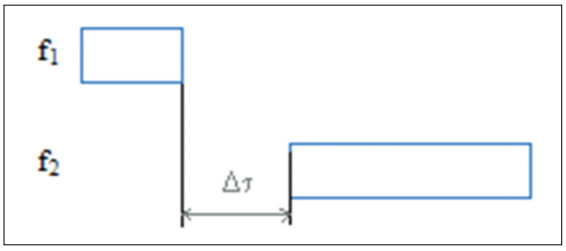

Figure 5. Parameterized following

Special operator <*I*> means logical incompatibility of actions, i.e. the processes cannot be found in the same case of execution. This is means that the same logical variable has value 1 in one vector, and 0 in another.

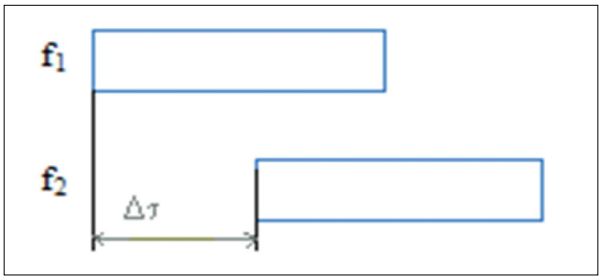

Figure 6. Parameterized overlap

These formal calculi are strong associated with algebraic models or real-time control algorithms [6].

Name	Mean	Signature
CH	'begin-begin'	$(UA_1, UA_2) \rightarrow UA$
CK	'end-end'	$(UA_1, UA_2) \rightarrow UA$
$\rightarrow$	direct following	$(UA_1, UA_2) \rightarrow UA$
H	parameterized overlay	$(UA_1, UA_2, int) \rightarrow UA$
3A	parameterized following	$(UA_1, UA_2, int) \rightarrow UA$
@	absolute time binding	$(UA, integer) \rightarrow UA$
$\Rightarrow$	qualification by logical condition	$(condition, UA) \rightarrow UA$

Table 1. Operators of RTCL

In some cases (due to values of involved variables, and specification to be checked) specification can be feasible with certain parameters of functional processes, but unfeasible with other parameters. The reader can found more detailed description in [7].

Example 1. For the following synchronization requirements: f_1 *CH* f_2; $f_1 \rightarrow f_3$; f_4 *CK* f_5; $f_3 \rightarrow f_4$; $f_2 \rightarrow f_5$, and parameters' values $\tau_1 = 20$, $\tau_2 = 100$, $\tau_3 = 200$, $\tau_4 = 10$, $\tau_5 = 50$, the specification is not feasible due to violation of $f_2 \rightarrow f_5$ requirement (this fact is obvious when we look at Figure 7).

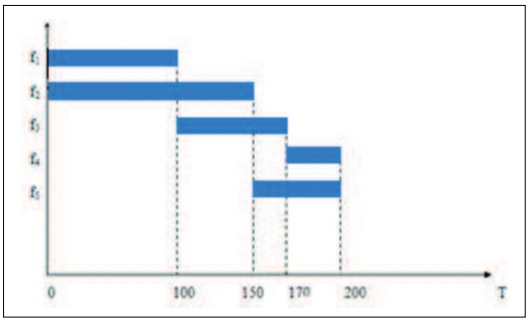

Figure 5. Example of feasibility checking

But if we have the another parameters, for example, $\tau_1 = 100$, $\tau_2 = 150$, $\tau_3 = 70$, $\tau_4 = 10$, $\tau_5 = 50$, specification becomes feasible (see Figure 8).

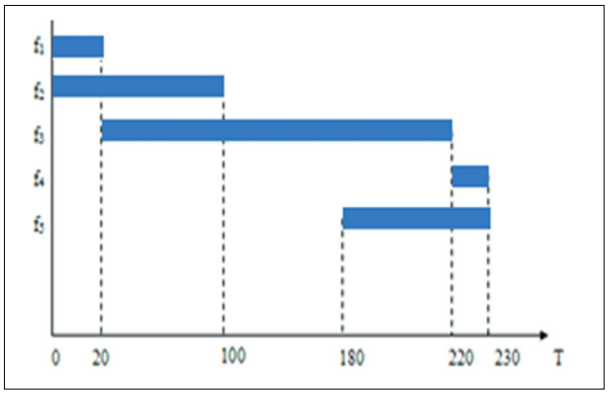

Figure 6. Example of feasibility checking

The very important point is that this model can be applied not only for real-time spacecraft's flight control software (domain where it was initially developed), but for representation of any sort of activity/processes performed by human, robots, various mechanisms, etc. In other words, the presented model is invariant to nature of performer. But at the same time, the model has enough expressive power for adequate representing of Real-Time control logic's complex features in 'time space' and 'logical space'.

2.2 Ways of utilization of SMT solvers functionality

It is not a wonder that the fundamental mathematical objects such as integers, rational and real numbers, vectors, and matrix are supported by existing SMT solvers by default. Consequently, if we will know how we can transform requirements applicable to control logic into requirements applicable to mentioned objects, then we have possibility to utilize functionality of available SMT solvers.

To do this, we use the following transition from relations between functional processes described as formulas of RTCL, to equations and inequalities on numbers.

RTCL formulae	Requires	Comment
$f_i\, CH f_j$	$t_i = t_j$	equation of numbers
$f_i\, CK f_j$	$t_i + \tau_i = t_j + \tau_j$	equation of numbers
$fi \rightarrow f_j$	$t_i + \tau_i = t_j$	equation of numbers
$3A(fi, fj, \delta)$	$t_i + \tau_i + \delta = t_j$	equation of numbers
$H(fi, fj, \delta)$	$t_i + \delta = t_j$	equation of numbers
$fi < fj$	$t_i < t_j$	inequality of numbers
$fi << f$	$t_i + \tau_i < t_j$	inequality of numbers
$f_i <> f_j$	$t_i + \tau_i < t_j \vee t_j + \tau_j < t_i$	disjunction of inequalities
$f_i <l> f_j$	set of boolean equations	logical incompatibilities of FPs (see above)

Table 2. Associations between the control logic requirements and inequalities and equations with numbers

2.3 SMT based Software Tool Prototype

Some of free SMT solvers provide API for calling them from user software. Some of them, for instance, Z3, accessible through Internet, the user can online specify required or unwanted properties using smt-lib language. Using this opportunity, we tried to apply functionality provided by SMT solver, for control logic checking. For this purpose, the software tool prototype was developed. Using the prototype, we have successfully validated prospectiveness of this approach.

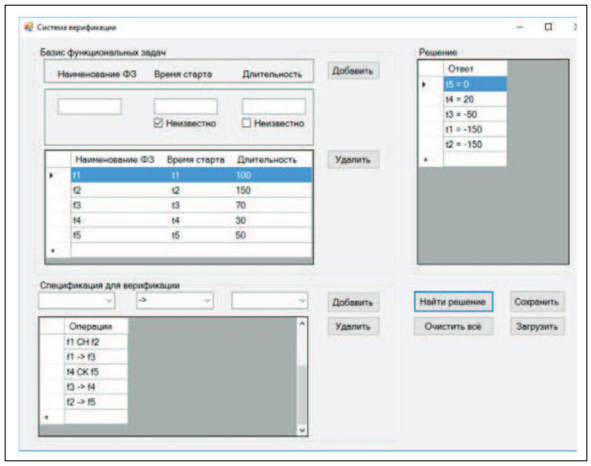

Figure 7. Screenshot of developed software tool prototype

Example 2. The software prototype coded in Java 8, it has intuitively understandable ease user interface. The screenshot is presented in Figure 3 (interface uses Russian).

First, user sets values of model variables in corresponding input fields. Then he needs step-by-step input specification to be verified. For this purpose, graphical user interface elements allow choosing operations of RTCAL logic. Transformation of specification represented in this from, into SMT solver smtlib language, is being performed automatically. There are also buttons for trying to evaluate of feasibility, saving and loading of given specification, etc. The result from Z3 in form 'sat' or 'unsat' is decoded, and if the specification is feasible, parameters' values are shown in special window.

2.4 Utilization of Logic Programming

Follow the monograph [4], let us try to analyze possibility of application of logic programming system's power for our purposes. It is well known that, for example, Prolog logic programming system is equipped with internal logical inference machine. Moreover during looking for answers to user specified questions, Prolog automatically finds values making answer positive (if inference machine system cannot found appropriate values, it returns answer 'No').

This feature provides us with a chance to use Real-Time Control Logic formalism in couple with logic programming system similarly we did it with SMT solvers. The operators of RTCL formulas have to be transformed into Prolog language predicates. The semantics of RTCL can be simply introduced in Prolog terms due to ease using of lists which are main data structure in Prolog language. It is convenient for us due the semantics of RTCAL is formed by tuples which can be reviewed as some equivalent to lists. The logical vector, due to its nature, can be simply represented by list as well. So, we can integrate the specially developed Prolog pre-built program module with specification to be verified, also presented in Prolog language. Then we specify the parameters of basic functional processes of control algorithm, using Prolog language. Doing this, we use variables for unknown parameters. After that, we formulate question (goal) for Prolog system, and get the answer, including values of unknown parameters allowing specification to be feasible.

Example 3.

with Prolog input

CK (*f*5, *f*3).

CK (*f*4, *f*3).

begin_time (*f*5, 10).

duration(*f*3, 50).

duration(*f*5, 90).

and goal ? *CK*(f3,X), user gets the answer 'Yes' and values $X = f3$, $X = f5$, $X = f4$, with the goal ?begin_time(*f*3, *X*). user gets 'Yes, $X = 50$'.

3. Conclusion

We have shown how the algebraic and logical based models of real-time control logic can be applied for feasibility checking using Satisfiability Modulo Theories solvers. The Real Time Control Logic presented in the paper, is a product of evolution of ideas formulated by A.A. Kalentyev in his early formal calculus of control algorithms, and algebra of real-time control algorithms. The proposed approach uses transformation of formal specification represented in terms of RTCL formulas, into equations and inequalities with integers. Then we convert them in SMT solver compatible smt-lib language. The paper presents prototype of software tool supporting the approach and based on calling Z3 SMT solver through its application programming interface.

Acknowledgement

Author wants to gratefully acknowledge Professor Anatoly Kalentyev who introduced him into world of science and particular area of space technologies and who found basis of Real Time Control Algorithm's Logic.

References

[1] Ashby, W. (1956). *Introduction to Cybernetics*, – New York, Chapman & Hall.

[2] Akhmetov, R. N., Makarov, V. P., Sollogub, A. V. (2012). Principles of the Earth Observation Satellites Control in Contingencies, *Information and Control Systems*, vol. 1, p. 16-22.

[3] Tyugashev, A. A. (2006). Integrated environment for designing realtime control algorithms, *Journal of Computer and Systems Sciences International*, 2 (45) 287-300.

[4] Kalentyev, A. A., Tiugashev, A. A. (2006). *Application of CALS technologies in Lifecycle of Complex Control Software*, Samara, Samara Centre of RAS Publishing, (in Russian).

[5] Tyugashev, A., Ilyin, I., Ermakov, I. (2012). Ways to improve quality and reliability of software in aerospace industry, Large-Scale Systems Control, p. 288–299, vol. 39, (in Russian).

[6] Tyugashev, A. (2017). Use of graph-based and algebraic models in lifecycle of real-time flight control software, *In*: Proceedings of Mathematical Modeling Session at the International Conference Information Technology and Nanotechnology (MM-ITNT 2017) p. 306-311, Samara, Russia, 19.11.2017.

[7] Tiugashev, A. (2017). Build and evaluation of real-time control algorithms in case of incomplete information about functional processes' parameters, *In*: Proceedings of 2017 XX IEEE International Conference on Soft Computing and Measurements (SCM'2017), p. 179-185, Saint Petersburg, Russia.