Case Driven TLC Model Checker Analysis in Energy Scenario

Vadym Shkarupylo [a,b], Ihor Blinov [c], Valentyna Dusheba [b], Jamil Abedalrahim Jamil Alsayaydeh [d]

[a] *National University of Life and Environmental Sciences of Ukraine, 15, Heroiv Oborony, Str., Kyiv, 03041, Ukraine*
[b] *G.E. Pukhov Institute for Modelling in Energy Engineering of the NAS of Ukraine, 15, General Naumov, Str., Kyiv, 03164, Ukraine*
[c] *Institute of Electrodynamics of the NAS of Ukraine, 56, Peremohy, Av., Kyiv, 03057, Ukraine*
[d] *Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal, Melaka, 76100, Malaysia*

Abstract

Today, model checking techniques and corresponding tools are widely applied in diverse case driven scenarios, the safety critical ones in particular. Addressing current situation in Ukraine, an energy domain is among the topical spheres, where safety critical business processes take place. To foster the functional safety of corresponding program-algorithmic solutions, the model checking techniques and related tools are applied to the formal specifications of named solutions. Doing so is not a trivial task: it depends on a particular use case scenario determining the architecture (structure and couplings) of the resulting design artifact. Moreover, the outcomes of formal techniques and tools application directly depend on specification atomicity level chosen – as a tradeoff between the complexity of program-algorithmic constituent addressed to be represented in formal specification and available computational and spatial resources of the computing platform with model checking technique implementation – because of an exponential growth of transition system state space. To this end, to foster the effectiveness of model checking technique application, with respect to a particular case driven scenario, the analysis of broadly applied TLC model checker has been conducted on the basis of a role model from energy domain. Experimentation has been conducted by addressing two alternative implementations of the TLC method. Both – computational and spatial properties – have been covered. To estimate also the domain related spatial expenses on verification, with respect to the number of software threads utilized, the approximation task has been resolved.

Keywords 1

Artifact, formal specification, model checking, safety critical scenario, TLA, TLC, verification

1. Introduction

Nowadays, the complexity level of modern program-algorithmic solutions addressing the scenarios taking place in diverse safety-critical domains, e.g., energy scenarios (Finnish nuclear industry [1]), avionics (satellite operational mode management scenarios [2]), safety critical software as a part of railway control systems [3], etc., typically exceeds the limitations of computational and spatial resources required to successfully apply time-proven formal verification techniques and corresponding instruments in one-to-one manner – due to the exponential growth of transition system state space to be traversed through during an automated formal verification by way of model checking [4]. For instance, considering the avionics scenarios, an exponential growth of both spatial [5] and computational [6] expenses has been faced. To diminish named effect, different approaches have been

The Sixth International Workshop on Computer Modeling and Intelligent Systems (CMIS-2023), May 3, 2023, Zaporizhzhia, Ukraine
EMAIL: shkarupylo.vadym@nubip.edu.ua (V. Shkarupylo); blinovihor@gmail.com (I. Blinov); vdusheba@gmail.com (V. Dusheba); jamil@utem.edu.my (J. A. J. Alsayaydeh)
ORCID: 0000-0002-0523-8910 (V. Shkarupylo); 0000-0001-8010-5301 (I. Blinov); 0000-0002-8929-3625 (V. Dusheba); 0000-0002-9768-4925 (J. A. J. Alsayaydeh)

proposed previously. Among those, there is an attempt to vary the atomicity level to be applied in formal specification by coupling the specified properties into the groups, i.e., the "hyper-properties", expressing the relations between the constituents; to this end, the broadly applied Temporal Logic of Actions (TLA; by Leslie Lamport) and the corresponding TLA Checker (TLC) have been brought to the use [7]. Alternatively, the decomposition based approach can be applied: formal specification is decomposed into sub-specifications to be subsequently verified with the time proven Event-B method [8]. Yet another promising direction to proceed is to decrease the model checking related computational expenses via a mu-calculus based abstraction reduction technique implementation [2]. In addition, the bounded model checking approach can be applied, e.g., by bringing the mathematical apparatus of the discrete time Markov chains (DTMC), paired with the probabilistic computation tree logic (PCTL) and probabilistic symbolic model checker (PRISM) [9]. Moreover, to diminish the effect of transition system state space exponential growth, the Reduced Ordered Binary Decision Diagrams (ROBDDs) are shown to be the proven instrument – more than 10^{20} states can be encompassed [10].

Thus, it can be seen that the problem of an exponential growth of the transition system state space is multi-dimensional, and can be approached diversely. At the same time, it can be noted that implementation related aspects taking place during the model checking are still lacking the attention of the research community – in terms of the influence of a particular implementation on the related computational and spatial expenses on certain case driven model checking task resolving.

In our work, the TLC model checker has been chosen to be the instrument to be applied during the case study – because of its wide usage in diverse safety critical scenarios: the grounding is provided below (in the Section 3).

Rest of the paper is organized as follows: in the Section 2, the problem statement is made; Section 3 is devoted to the related work analysis; in the Section 4, the case study addressing the scenario from the energy domain is described; Section 5 contains the conclusion and thoughts on the future work; the acknowledgments are given in the Section 6; in the Section 7, the references are provided.

2. Problem statement

Taking into consideration the aforesaid, paper addresses the problem of an exponential growth of the transition system state space – in terms of the corresponding computational and spatial expenses. An attempt to estimate named expenses, with respect to a case driven scenario from the energy domain has been made. To this end, the broadly used TLC model checker (method) has been utilized as an instrument. Two alternative implementations of the TLC have been investigated in terms of the related computational and spatial expenses: the BFS (Breadth-first Search) one – implemented by the default; the alternative DFS (Depth-first Search) implementation. As it can be seen from the naming, these implementations differ by the method applied to traverse through the states of a transition system.

Let the model checking task is formalized as follows [4]:

$$M, b \models \psi, \tag{1}$$

where M – the transition system – Kripke structure over a set of the atomic prepositions AP; b – "behavior" – as a sequence of states to be traversed through during the model checking with the TLC method; ψ – temporal formula, in TLA+ syntax, prescribed in the formal specification. To positively state regarding specification consistency, ψ has to be "true" for each element of b.

In given paper, the model checking task has been resolved with respect to a case driven scenario taking place in energy domain, and specified with a role model of European identification codes registry update process.

The idea is to estimate and compare different implementations of the TLC method, by grounding on a particular case driven scenario – with respect to corresponding computational and spatial expenses that take place during the model checking. This approach is an attempt to discover and

assess the factors – e.g., the architecture (structure and couplings) specified in the formal model, number of state variables, etc. – affecting the computational and spatial costs of formal verification with a particular implementation of certain model checker (the TLC in our case). Moreover, an attempt to estimate the outcome of bringing the multithreading to the implementations of the TLC method has also been made.

3. Related work

When addressing specification atomicity concept, it first needs to be noted that, prior to being represented formally, the specified properties are commonly represented in certain textual or graphical form, e.g., with BPMN-notation (Business Process Model and Notation). To verify related spatial properties, the sBPMN Verification Framework has been proposed [11]: it is based on the TLC model checker application to formal specifications written in the TLA+ formalism of the TLA temporal logic [12]. The distinctive features of named formalism are the modularity and mathematical strictness. It can be addressed as a propositional logic coupled with the temporal operators: X (Next) and G (Globally). Because of the fact that model checkers (the implementations of model checking techniques) are the instruments to be applied to formal specifications, but not to the initial artifacts directly, there is also a necessity to control the adequacy of specifications. It can be accomplished, for instance, by comparing the properties of the transition system retrieved from the initial artifact – e.g., block-diagram, UML (Unified Modeling Language) activity diagram, etc. – and from the resulting formal specification [5]. In addition, to formally check the abstractions applied in specification, a deductive verification based technique can be utilized prior to the model checking [13]. Moreover, similarly to [11], from the architectural viewpoint, the TLC and corresponding tools have successfully been applied to discover the inconsistencies that may occur during the SDN (Software-defined Networking) topology and related policies changes [14].

As a scenario representing the necessity of chosen specification abstraction level adoption, i.e. atomicity level varying, an industrial distributed Taurus database formal specification synthesis and verification process can be considered: named TLC model checker has been successfully applied to consistency checking task resolving [15]. The obtained results have once again shown the effectiveness of model checking technique to be utilized for design flaws discovery at the design stage of engineering process, prior to the validation, e.g., testing in particular. In addition, in case there are no design flaws that have been discovered while model checking, verification outcome can be approached differently: either be treated as the design solution consistency approval, or as an indicator that specification atomicity level applied needs to be shifted. Moreover, dealing with the processes taking place in the large-scale distributed software systems, the task of eventual consistency over the data replicas maintenance arises: formal instruments (TLC, TLA, TLA+) have been utilized within the MET (Model Checking-driven Explorative Testing) framework – in an attempt to construct the "bridge" between the model checking being applied at the design stage of engineering process and the validation by way of testing: to address a tradeoff between the exhaustive nature of model checking facing the exponential growth of transition system state space and case-driven testing [16]. To diminish the effect of named exponential growth, the compositional model checking techniques can be applied, e.g., the Interaction-Preserving Abstraction (IPA) framework addressing the specifications written in TLA+ [17]. Moreover, novel TLA+ Debugger utility makes it possible not only debug the temporal formulas evaluations, but also inspect states and transitions of transition system [18].

4. Case study

As a demonstrative problem domain, where diverse safety-critical scenarios take place, modern electricity market of Ukraine [19], adjusted in accordance with the harmonized European electricity market model, has been considered [20]. European identification codes registry update process has been addressed as a case study (Figure 1).

Considering the UML (Unified Modeling Language) diagram, depicted in the Figure 1, the TLA+ specification atomicity level has been chosen to be applied in “one-to-one” manner. Similar step has been made in the previous case study, where the avionics scenario has been approached [5].

Among the distinctive features of current case study, there are, in particular, the different type of the initial artifact (UML activity diagram instead of block diagram), and also the different problem domain.

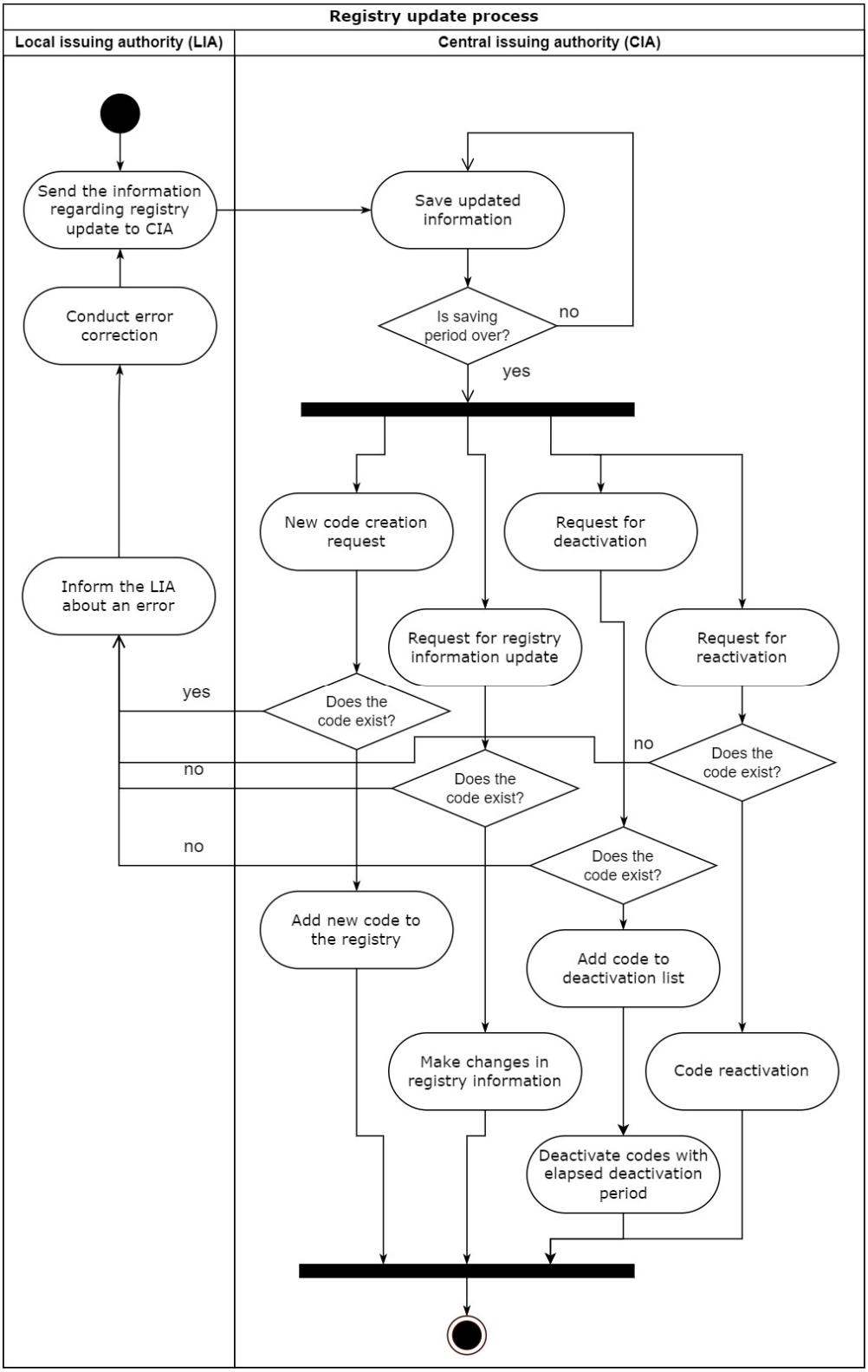

Figure 1: Fragment of a role model addressing the European identification codes registry update process

In the Figure 1, a fragment of the complete role model is depicted as the UML activity diagram. A complete role model encompasses 23 activities, plus 6 conditional operators. Named model is approached as an artifact – an entity with the architecture (structure and couplings) and the content [21]. These constituents are represented in the formal TLA+ specification with respect to the approach described below.

4.1. Formalization and experimentation

To synthesize the formal specification suitable for verification in an automated manner, the following two staged approach has been applied:

1. To create the architectural prototype (encompassing both algorithmic structure and the couplings; in a pseudo code-like manner) of yet to be synthesized TLA+ specification, the PlusCal algorithmic language has been used [22]. An outcome of this stage is an input data for the following one. The PlusCal specification is treated as the preliminary artifact.
2. To generate the resulting artifact – the TLA+ specification – from the PlusCal pseudo code, the TLA+ Toolbox IDE (Integrated Development Environment) has been utilized [12].

The TLA+ specification is addressed to be the input data for the TLC model checker intended to be applied in an automated manner.

4.1.1. Approach to the PlusCal specification synthesis

To obtain the preliminary artifact, the following approach has been applied:

1. Each of the aforementioned activities (Figure 1) has been represented in the PlusCal specification with a state variable. Named variables have been grouped within corresponding set V': $|V'| = 23$.
2. Moreover, additional three state variables have been created to represent conditional operators: initial conditional operator, prior the fork construct; group of 4 conditional operators after the fork construct (Figure 1); final conditional operator which is out of the scope of diagram fragment depicted. As an outcome, the complementary state variables have been introduced with the following set: V'': $|V''| = 3$.
3. Yet another state variable has been used to initiate/terminate the computational process formalized in the PlusCal specification.

Thus, with respect to the approach applied, the resulting state variables set has been obtained as follows: $V = V' \cup V'' \cup \{v^*\}$, where $v^* \in V$ is the initiate/terminate state variable.

4.1.2. TLA+ specification synthesis and checking

It has taken 348 rows of pseudo code to represent the role model in PlusCal. After that, the instruments of TLA+ Toolbox IDE have been applied to generate the resulting TLA+ specification from the PlusCal representation. As an outcome, the TLA+ specification has been synthesized containing 508 rows of code. Thus, it can be seen that the initial PlusCal artifact is significantly more concise comparing to the resulting TLA+ specification, while preserving the same architectural constituent.

The consistency of synthesized TLA+ specification has been proven in an automated manner – with the TLC applied. To state about the consistency of the initial role model (Figure 1), the aforementioned adequacy checking technique has been used [5].

Spatial properties of a transition system traversed through during an automated TLC checking can be represented with the elements of Kripke structure M (1) [4], over a set of atomic prepositions

$AP = V \times D$, where, in our case, $D = \{0,1\}$ – set of allowable state variable values, i.e., set of Boolean values: $M = \langle S_0, S, R, L \rangle$, where $S_0 \subset S$ – set of initial states; S – finite set of states; $R \subseteq S^2$ – total set of transitions: $\forall s \in S \; \exists s' \in S : (s, s') \in R$; $L : S \to 2^{AP}$ – states labeling function.

4.1.3. Transition system analysis

In our case, $S_0 = \{s_0, s_1\}$, where $s : V \to D$: $L(s_0) \Delta L(s_1) = \{(v'', 0), (v'', 1)\}$, where $v'' \in V'' \subset V$ state variable represents the group of four identical conditions after the fork construct in the Figure 1. Depending on whether the identification code exists initially, we can have either $s_0 \in S_0$ or $s_1 \in S_0$ initial state.

To form the AP set, a dichotomy principle has been applied: $AP = AP' \cup AP''$, where $AP' = V \times \{0\}$, $AP'' = V \times \{1\}$. Elements of these subsets have been approached as follows: $(v_i, 0) \in AP'$ – i-th activity has not been accomplished yet ($i = 1,2,...,27$); $(v_i, 1) \in AP''$ – i-th activity has already been accomplished.

Transition system (TS) spatial properties discovered during an automated formal verification with the TLC model checker are as follows: "TS depth" – 28 – number of sequential transitions from certain initial state ($s_0 \in S_0$ or $s_1 \in S_0$) to the final one; total number of distinct states found – $|S| = 290$. These properties have been treated as the indexes of model checking task complexity.

4.1.4. Experimentation and obtained results

Experimentation has been conducted on the following platform: CPU – AMD Ryzen R5 2400g; RAM – 16 GB; Java Runtime Environment, version "1.8.0_241"; TLC version: 2.14 of 10 July 2019.

During this, two alternative implementations of the TLC model checker have been encompassed: the BFS- and the DFS-based ones. It needs to be noted, though, that the DFS-implementation of the TLC is coupled with a significant drawback in terms of the automation – "TS depth" has to be discovered first, and then be specified manually. On the contrary, default BFS-implementation does not need to be accompanied with the "TS depth" parameter.

Results of previous experimentations have shown, that, depending on the architectural plane of specification, there is a bound, in terms of the number of state variables, when certain TLC implementation is more efficient in terms of the corresponding computational expenses [6, 23].

Adequacy of the resulting TLA+ specification has been proven with the aforementioned technique [5]. Two alternative implementations (BFS and DFS) of the TLC model checker have been applied, in both single and multithreaded manner.

Let $tn = 2^0, 2^1, ..., 2^3$ be the number of software threads utilized for the TLC model checking. Let $f(tn)$ be the related computational expenses. Let the speedup from multithreading implementation is calculated as follows: $\alpha(tn) = (f(1) / f(tn))$. Obtained experimental results are provided in the Table 1.

Table 1

Computational expenses on the automated formal verification with the TLC model checker

tn	$f_{BFS}(tn)$, ms	$\alpha_{BFS}(tn)$	$\lvert S^*_{BFS} \rvert$	$f_{DFS}(tn)$, ms	$\alpha_{DFS}(tn)$	$\lvert \overline{S^*_{DFS}} \rvert$	ξ_{DFS}
2^0	1214.000	1.000	336	638.400	1.000	4194.000	1.000
2^1	1157.800	1.049	336	632.900	1.009	5382.890	1.283
2^2	1149.100	1.056	336	665.300	0.960	7267.050	1.733
2^3	1143.800	1.061	336	682.700	0.935	8209.850	1.958

In the Table 1, each numerical value is an arithmetical average of 10^2 measures; $f_{BFS}(tn)$ – computational expenses related with the BFS implementation of the TLC method; $f_{DFS}(tn)$ – computational cost of the alternative DFS implementation; $\alpha_{BFS}(tn)$ – speedup from bringing multithreading to the BFS implementation; $\alpha_{DFS}(tn)$ – speedup taking place for the DFS implementation of the TLC; $\left|S^*_{BFS}\right| = const$ – total number of states generated while the BFS model checking to construct the resulting transition system with $|S| = 290$ states; $\left|\overline{S^*_{DFS}}\right|$ – average total number of states generated with the alternative DFS implementation (an average value has been calculated because of the non-deterministic nature of the DFS implementation when the multithreading is applied); ξ_{DFS} – model checking task spatial complexity relative estimation, depending on the number of threads utilized; is calculated conceptually similarly to $\alpha(tn)$: $\xi_{DFS} = \left(\overline{S^*_{DFS_1}} \Big/ \overline{S^*_{DFS_{tn}}}\right)$.

In the Table 1, it can be seen that the DFS implementation of the TLC method has appeared to be significantly more effective in terms of the related computational expenses, when comparing to the default BFS alternative: from about 1.902 times (for $tn = 2^0$) to about 1.675 times (for $tn = 2^3$). It does correspond to the results obtained previously [6, 23], and it can be considered as a significantly viable argument in favor of the DFS implementation when the iterative approach to verification takes place [24]. On the contrary, when addressing the spatial properties of the model checking task resolved with a particular TLC implementation, the picture changes drastically – in terms of the ratio between the numbers of transition system states generated ($\left|S^*_{BFS}\right|$ and $\left|\overline{S^*_{DFS}}\right|$ values): $\left(\left|\overline{S^*_{DFS}}\right| \Big/ \left|S^*_{BFS}\right|\right) \in [12.482; 24.434]$. The BFS implementation becomes to be significantly more preferable.

With respect to the multithreading, the BFS implementation once again looks to be having a significant advantage in terms of the spatial aspect – because of the constant value of $\left|S^*_{BFS}\right|$ index, regarding the number of software threads utilized. At the same time, when dealing with the alternative DFS implementation, the value of $\left|\overline{S^*_{DFS}}\right|$ index rises rapidly with the increase of the number of concurrently acting threads.

In an attempt to estimate the growth of the DFS-related spatial expenses (from the number of software threads utilized), an approximation task has been resolved on the basis of tn and ξ_{DFS} indexes (Figure 2). As an outcome, the following estimation function $\xi'_{DFS}(tn)$ has been obtained:

$$\xi'_{DFS}(tn)^{-1} = (a + b/tn), \tag{2}$$

where $a = 0.435$, $b = 0.603$; determination coefficient $R^2 = 0.986$.

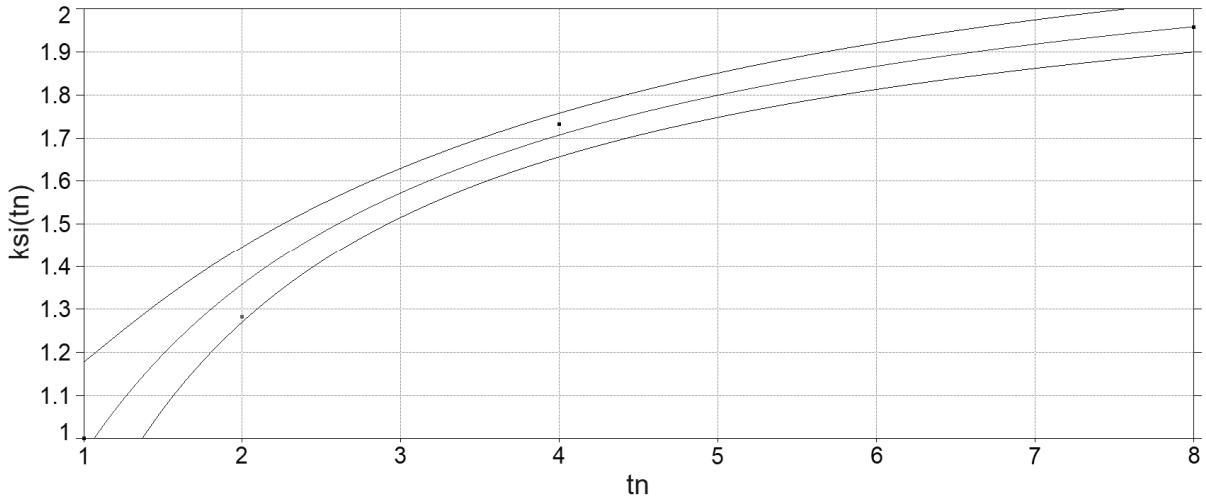

Figure 2: DFS related spatial expenses growth estimation, with respect the number of threads utilized

In the Figure 2, confidence intervals have been built for the confidence probability 0.95. Expression (2) can be used as an estimation of spatial expenses growth, with the increase of the number of concurrently acting threads.

In terms of the computational expenses, it can be seen, in the Table 1, that bringing multithreading to the default BFS implementation does not lead to a significant speedup. Moreover, the major "leap ahead" has been faced while shifting from $tn = 2^0$ to $tn = 2^1$ – about 5% improvement. These results and concluding remarks conform to the ones obtained previously, when an avionics safety critical scenario has been addressed [25]. On the contrary, in accordance with the data from the Table 1, when dealing with multithreaded DFS implementation, the outcome is contradictory: in case of $tn = 2^1$, just a minor speedup (about 1%) has been faced; at the same time, by further increasing the tn value, the results have appeared to be even worse, an opposite trend has been revealed. On the other hand, previously obtained results from the avionics domain have shown a significant positive trend (more than two times speedup for the case of $tn = 2^2$), with a slight decrease for the case of $tn = 2^3$ [25]. Such contradictory outcomes for different case scenarios prompt an assumption that DFS implementation is vastly case sensitive, depending on the number of state variables and the architectural plane of formal specification.

To summarize the distinctive features of both implementations of the TLC method, with respect to the energy domain scenario considered, the following conclusions can be formulated:

1. Default BFS implementation of the TLC model checker can be considered to be the more preferable solution in terms of the following aspects: no need to specify the depth of search space – a definitive argument in terms of the automation, when comparing to the DFS alternative, where named parameter has to be specified manually; constant spatial expenses on the model checking tasks resolving, in terms of the multithreaded implementation.
2. Under proper circumstances, the alternative DFS implementation of the TLC method provides a significant verification related time costs reduction. By encompassing also the results of previous experimentations (both synthetic and domain related – avionics) [5, 6, 23, 25], these circumstances (factors) have been discovered to be the number of state variables and the architectural plane of formal specification.

These concluding thoughts provide the background to further increase the set of case driven scenarios with the TLC model checker implementations utilized, in an attempt to work out and generalize the rules (recommendations) for a particular TLC implementation practical usage, in terms of the related computational and/or spatial expenses decrease.

5. Conclusion

Thus, broadly used TLC model checker has been investigated in given paper on the basis of the case driven scenario taking place in modern energy domain: a role model describing the European identification codes registry update process has been addressed as a design artifact. The consistency of corresponding program-algorithmic constituent has been proven with the named method applied. To prove the credibility of the results obtained, the adequacy of the resulting formal specification has been checked with respect to previously introduced technique.

Both alternative implementations of the TLC method have been considered, including the multithreading aspect: the BFS and the DFS implementations. It has been found out that while the BFS-related outcomes conform to the results and assumptions that have been made previously (on the basis of synthetic and domain related – avionics – scenarios), the DFS-related ones, on contrary, have appeared to be demonstrating just a minor speedup (about 5%) in the case of two threads applied; with further increase of the number of threads utilized, even the negative speedup factor has been faced. These results have led us to a conclusion that the DFS implementation of the TLC method significantly depends on both the number of state variables taking place in formal specification and the architectural plane of the latter. At the same time, in general, on the basis of the case study conducted, the DFS implementation has appeared to be about 1.675 – 1.902 times more efficient in terms of the related time costs, when comparing to the default BFS implementation of the TLC. On the contrary, when addressing the spatial aspects (the number of transition system states generated), the DFS implementation has appeared to be significantly worse comparing to the BFS alternative: from about 12.482 to about 24.434 times.

With an intention to assess the character of the DFS-related spatial expenses growth, with respect to the number of concurrently acting software threads, the approximation task has been resolved, and corresponding estimating function has been obtained.

Future work is focused on an attempt to formulate the recommendations to both TLC implementations (the BFS and the DFS) practical usage, depending on the number of state variables taking place in the formal specification and on the peculiarities of the architectural plane of the latter.

6. Acknowledgements

Paper has been prepared on the basis of the results obtained in accordance with the tasks of the following research works: 0120U102683 “Development of specialized computer technologies for modeling and processing of operational information in energy problems”; 0121U110615 “Development of methods and means for safety-critical systems designing process artifacts verification”, carried out by the Department of Mathematical and Computer Modeling of the G.E. Pukhov Institute for Modeling in Energy Engineering of the National Academy of Sciences of Ukraine, with a contribution of the Institute of Electrodynamics of the National Academy of Sciences of Ukraine. Paper also addresses the tasks of W911NF-22-2-0153 research work carried out by the G.E. Pukhov Institute for Modeling in Energy Engineering of the National Academy of Sciences of Ukraine and funded by the US Army Engineer Research and Development Center (ERDC).

7. References

[1] A. Pakonen, T. Tahvonen, M. Hartikainen, M. Pihlanko, Practical applications of model checking in the Finnish nuclear industry, in: Proc. 10th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies, San Francisco, CA, USA, June 11–15, 2017, pp. 1342–1352.

[2] V. Nardone, A. Santone, M. Tipaldi, D. Liuzza, L. Glielmo, Model checking techniques applied to satellite operational mode management, IEEE Systems Journal, vol. 13(1) (2019) 1018–1029. doi: https://doi.org/10.1109/JSYST.2018.2793665.

[3] S. Resch, M. Paulitsch, Using TLA+ in the development of a safety-critical fault-tolerant middleware, in: 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Toulouse, France, 23-26 October 2017, pp. 146–152. doi: https://doi.org/10.1109/ISSREW.2017.43.

[4] E. M. Clarke, O. Grumberg, D. Kroening, D. Peled, H. Veith, Model checking: 2nd ed. Massachusetts: The MIT Press, 2018.
[5] V. Shkarupylo, J.A.J. Alsayaydeh, I. Tomičić, A. Chemeris, V. Dusheba, A technique for checking the adequacy of formal model, ARPN Journal of Engineering and Applied Sciences, 16 (2021) 1707–1719. URL: http://www.arpnjournals.org/jeas/research_papers/rp_2021/jeas_0821_8670.pdf.
[6] V. Shkarupylo, I. Blinov, A. Chemeris, V. Dusheba, J. Alsayaydeh, A. Oliinyk, Iterative approach to TLC model checker application, in: Proc. 2021 IEEE KhPI Week on Advanced Technology, Kharkiv, Ukraine, September 13–17, 2021, pp. 283–287. doi: https://doi.org/10.1109/KhPIWeek53812.2021.9570055.
[7] L. Lamport, F. B. Schneider, Verifying hyperproperties with TLA, IEEE Computer Security Foundations Symposium, CSF 2021, IEEE (2021). doi: https://doi.org/10.1109/CSF51468.2021.00012.
[8] K. Kraibi, R.B. Ayed, J. Rehm, S. Collart-Dutilleul, P. Bon, D. Petit, Event-B Decomposition analysis for systems behavior modeling, in: Proc. 14th International Conference on Software Technologies, ICSOFT 2019, Prague, Czech Republic, July 26–28, 2019, vol. 1, pp. 278–286. doi: https://doi.org/10.5220/0007929602780286.
[9] H. Gao, W. Huang, X. Yang, Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data, Intelligent Automation and Soft Computing, vol. 25(3) (2019) 547–559.
[10] A. Biere, A. Cimatti, E.M. Clarke, O. Strichman, Y. Zhu, Bounded model checking, Advances in computers, vol. 58(11) (2003) 117–148.
[11] R. Saddem-Yagoubi, P. Poizat, S. Houhou, Business processes meet spatial concerns: the sBPMN verification framework, in: M. Huisman, C. Păsăreanu, N. Zhan (Ed.), Formal Methods, FM 2021, Lecture Notes in Computer Science, vol. 13047, Springer, Cham, 2021, pp. 218–234. doi: https://doi.org/10.1007/978-3-030-90870-6_12.
[12] M. A. Kuppe, L. Lamport, D. Ricketts, The TLA+ toolbox, in: 5th Workshop on Formal Integrated Development Environment, F-IDE 2019, Porto, Portugal, October 7, 2019, EPTCS 310, 2019, pp. 50–62. doi: http://doi.org/10.4204/EPTCS.310.6.
[13] J. Amilon, C. Lidström, D. Gurov, Deductive verification based abstraction for software model checking, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Verification Principles, ISoLA 2022, Lecture Notes in Computer Science, vol. 13701, Springer, Cham, 2022. doi: https://doi.org/10.1007/978-3-031-19849-6_2.
[14] Y.-M. Kim, M. Kang, Formal verification of SDN-based firewalls by using TLA+, IEEE Access, 8 (2020) 52100–52112. doi: https://doi.org/10.1109/ACCESS.2020.2979894.
[15] S. Gao et al., Formal verification of consensus in the Taurus distributed database, in: M. Huisman, C. Păsăreanu, N. Zhan (Eds.), Formal Methods, FM 2021, Lecture Notes in Computer Science, vol. 13047, Springer, Cham, 2021. doi: https://doi.org/10.1007/978-3-030-90870-6_42.
[16] Y. Zhang, Y. Huang, H. Wei, X. Ma, MET: Model checking-driven explorative testing of CRDT Designs and Implementations: report, 2022. doi: https://doi.org/10.48550/arXiv.2204.14129.
[17] X. Gu et al., Compositional model checking of consensus protocols via interaction-preserving abstraction, in: 2022 41st International Symposium on Reliable Distributed Systems (SRDS), Vienna, Austria, September 19–22, 2022. doi: https://doi.org/10.1109/SRDS55811.2022.00018.
[18] M. A. Kuppe, The TLA+ debugger, in: P. Masci, C. Bernardeschi, P. Graziani, M. Koddenbrock, M. Palmieri (Eds.), Software Engineering and Formal Methods, SEFM 2022 Collocated Workshops, SEFM 2022, Lecture Notes in Computer Science, vol. 13765, Springer, Cham, 2023. doi: https://doi.org/10.1007/978-3-031-26236-4_15.
[19] I.V. Blinov, Ye.V. Parus, H.A. Ivanov, Imitation modeling of the balancing electricity market functioning taking into account system constraints on the parameters of the IPS of Ukraine mode, Tekhnichna elektrodynamika, 6 (2017) 72–79. doi: https://doi.org/10.15407/techned2017.06.072.
[20] I. Blinov, S. Tankevych, The harmonized role model of electricity market in Ukraine, in: 2016 2nd International Conference on Intelligent Energy and Power Systems, IEPS 2016 Conference Proceeding, Kyiv, Ukraine, 07–11 June 2016. doi: https://doi.org/10.1109/IEPS.2016.7521861.
[21] M. Broy, A logical approach to systems engineering artifacts and traceability: from requirements to functional and architectural views, in: M. Broy, D. Peled, G. Kalus (Eds.), vol. 34:

Engineering Dependable Software Systems, NATO Science for Peace and Security Series – D: Information and Communication Security, IOS Press, 2013, pp. 1–48. doi: https://doi.org/10.3233/978-1-61499-207-3-1.
[22] H. Alkayed, H. Cirstea, S. Merz, An extension of PlusCal for modeling distributed algorithms, TLA+ Community Event 2020, Oct. 2020, Freiburg (online), Germany. URL: https://hal.inria.fr/hal-03143502/.
[23] V.V. Shkarupylo, I. Tomičić, K.M. Kasian, The investigation of TLC model checker properties, Journal of Information and Organizational Sciences, 1 (2016) 145–152. doi: https://doi.org/10.31341/jios.40.1.7.
[24] E. Mercer, K. Slind, I. Amundson et al., Synthesizing verified components for cyber assured systems engineering, Software and Systems Modeling, 2023. doi: https://doi.org/10.1007/s10270-023-01096-3.
[25] V.V. Shkarupylo, I.V. Blinov, A.A. Chemeris et al., On applicability of model checking technique in power systems and electric power industry, in: A. Zaporozhets (Ed.), Systems, Decision and Control in Energy III. Studies in Systems, Decision and Control, vol. 399. Springer, Cham, 2022. doi: https://doi.org/10.1007/978-3-030-87675-3_1.