Review of: "Formal Verification of a Change Control Process in Project Management"

Heerko Groefsema[1]

1 University of Groningen

Potential competing interests: No potential competing interests to declare.

This paper describes a method to formally verify control-flow requirements of a process through model checking with CTL and NuSMV.

However, there are a number of fundamental and significant concerns when reading this work.

First and foremost, the authors claim that this "to the best of their knowledge has not been done before". It is not true that this has not been done before. In fact, there is a whole research area devoted to solve exactly this. For example, the following approach has been published in 2015 and 2016 [1, 2], presenting an efficient approach to convert processes modelled in Coloured Petri Nets to Kripke Structures and subsequently verify their correctness on a set of CTL rules using NuSMV. In addition, [1] and [2] present an efficient way of handling concurrency and local next operators, and avoid combinatorial explosion of the resulting state space. As such, the manuscript reviewed here does not add any novelty to the state of the art and does not build on any earlier work in the space (see e.g. [3 - 10]). The authors here do, however, refer to a work of one of the authors in [1, 2], so claiming they're unaware of any such work is, at the very least, peculiar.

It may be true that this has not been done for Integrated Change Processes, but it has been done for processes in general and, therefore, also for Integrated Change Processes.

With respect to the translation from the model to the Kripke structure,I do not understand why [Bjørner, 70] is used instead of the more modern workflow/BP translations. Furthermore, one could argue that the labels on the transitions offer much more detailed information than those on the states. Why not use those? Moreover, Figures 5 and onwards are NOT Kripke structures, as they show labels on the relations. Note that CTL is interpreted over labels representing propositions on states, not over similar labels on transitions.

The entire formal framework (i.e. model, Kripke structure, CTL semantics and translation) is not defined formally, which would have helped to see the discrepancy between actual Kripke structures and the figures depicted herein.

It is then even more surprising to see a translation from a Kripke structure to a computation tree (with labels on the transitions again...). Although CTL is defined on Computation Trees, this step is entirely unnecessary as it can be interpreted DIRECTLY on Kripke structures. That is, the entire evaluation is then up to the model checker.

The formulation of CTL specifications are flawed as well.
- First, auxiliary variables to refer to declarations concerning past states are most often not required. For example:

"Whenever a change is assigned (state #8), previously it has been registered and analyzed (state # 2)", can be formulated in CTL as !E [! state_2 U state_8] (using state_# as shorthand for the labels used).
- Second, CTL specifications are interpreted to hold in the set of initial states. As a result, the specification listed in section 4.1 works because state_1 is the initial state. However, the same pattern would not work for non-initial states. Therefore, the specification used should instead be "AG (State = State_1_IdentifiedNeedForChange -> EF (State = State_20_ClosedChangeReleasedWithProcessImprovements))".
- Third, the specification in section 4.2 does not require the HasGoneThrough variables. See the first comment on auxiliary variables. However, in this case, the specification can be even as simple as "A [(State_1 | State_2 | State_6 | State_7) U State_8]".
- Fourth, the specification in section 4.3 should be preceded by the CTL AG operator. Currently, the antecedent of the implication is only considered at the initial state, where it is false. As a result, the implication is true, and the consequent is never considered, returning a false positive.
- Finally, the specification in section 4.4 has the same issues and highlights the painful misunderstanding of how CTL works, by combinatorially specifying ALL allowed options as separate auxiliary variables in the rules. That is, the correct process paths are hard-coded in the rules, which would be interesting to see in case of large concurrent processes.

How does the approach deal with possible infinite loops?

p13: "Even if some paths in the execution tree are infinite, model checking is guaranteed to terminate." A tree cannot have loops, so it cannot be infinite, unless there exists an infinite number of states on a branch. In this case, model checking does not terminate.
Section 4.5: "it should be true for all situations where a test fails, that a correcting cycle leads to new tests and eventually to approval." Has the use of fairness in NuSMV2 been considered? This could make the verification much stronger (and solves the termination problem for loops).

How does the approach cater for concurrency? Concurrent paths lead to a combinatorial amount of states caused by interleaving. Unless there is a specific method to deal with this state explosion, the approach is intractable very quickly.

p16: "Of course, a finite number of runs does not give absolute surety of the correctness of the change control process". During formal verification we are not interested in a limited number of runs, but ALL possible runs. Therefore, this discussion not helpful to the goals of this paper.

However, it is interesting to see the performance of the presented approach for models of increasing size and complexity such as large concurrent models, particularly in comparison with existing approaches solving the same problem.

I strongly recommend to have the paper reviewed by a native English speaker prior to (re)submission, see e.g. p17: "there was not transition from state #6 ... state #12".

To summarise, the manuscript has a significant number of serious flaws with respect to the novelty, technical depth and correctness of the approach.

References used in this review:

[1] Groefsema, H., and Van Beest, N.R.T.P. "Design-time compliance of service compositions in dynamic service environments." 2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA). IEEE, 2015.

[2] Groefsema, H, Van Beest, N.R.T.P., and Aiello M. "A formal model for compliance verification of service compositions." IEEE Transactions on Services Computing 11.3 (2016): 466-479.

[3] G. Governatori, Z. Milosevic, and S. Sadiq, “Compliance checking between business processes and business contracts,” in Enterprise Distributed Object Computing Conference, 2006. EDOC’06. 10th IEEE International. IEEE, 2006, pp. 221–232. 3]

[4] P. Bulanov, A. Lazovik, and M. Aiello, “Business process customization using process merging techniques,” in Service-Oriented Computing and Applications (SOCA), 2011 IEEE Int. Conf. on. IEEE, 2011, pp. 1–4. 4]

[5] S. Nakajima, “Verification of Web service flows with model-checking techniques,” in Proc. Int. Symp. on Cyber Worlds, 2002, pp. 378–385.

[6] W. Janssen, R. Mateescu, S. Mauw, and J. Springintveld, “Verifying business processes using SPIN,” in Proc. of the 4th Int. SPIN Workshop, 1998, pp. 21–36.

[7] B. Anderson, J. V. Hansen, P. Lowry, and S. Summers, “Model checking for E-business control and assurance,” Systems, Man, and Cybernetics, IEEE Trans. on, vol. 35, no. 3, pp. 445–450, 2005.

[8] D. Bianculli, C. Ghezzi, and P. Spoletini, “A model checking approach to verify bpel4ws workflows,” in Service-Oriented Computing and Applications, 2007. SOCA’07. IEEE International Conference on, 2007, pp. 13–20.

[9] O. Kherbouche, A. Ahmad, and H. Basson, “Using model checking to control the structural errors in bpmn models,” in Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Con- ference on, 2013, pp. 1–12.

[10] A. Kheldoun, K. Barkaoui, and M. Ioualalen, “Specification and verification of complex business processes - a high-level petri net-based approach,” in Business Process Management, ser. LNCS. Springer International Publishing, 2015, vol. 9253, pp. 55–71.