Responses to Reviews

Manuscript Number:	15-TIE-3480
Manuscript Title:	Formal Modeling and Verification of a Rate-Monotonic Scheduling Implementation with Real-Time Maude
Submitted to:	Transactions on Industrial Electronics
Manuscript Type:	Regular paper

I. General Response

We thank all the reviewers for their careful considerations on our paper. The comments are very insightful and invaluable. Thanks to the comments, we have improved the paper a lot to make it clearer and better presented.

The main changes in the paper are as following, which we have also highlighted in the revised manuscript:

- Section IV about the formal model of the target implementation is revised. Some technical details have been simplified and more literal explanations are used to deliver our ideas.
- Section V.B is expanded. Some discussion on the efficiency of our approach is added.
- The detailed technical proof of Theorem 2 in Section V.C has been included in the Appendix.
- Section VI about related work has been improved.

Detailed responses to all the comments are shown in the rest of this document.

II. Responses to Reviewer 1

A. General Comments

COMMENT:

On the positive side, this paper presents a solid piece of work. Periodic task scheduling is an important (and difficult) problem, and the topic of the paper appears to be well suited within the field of the industrial real-time systems. Overall the paper is well structured, technically accurate, presents the work in a comprehensive and reasonable way, and tackles theoretical as well as practical aspects. There are two principal contributions of the paper:

1) A novel implementation of the Rate-Monotonic Scheduling algorithm, which contains more complex and realistic details instead of the ideal setting.

2) Rate-Monotonic Scheduling is investigated using Real-Time Maude for the first time, to apply formal methods such as model checking and theorem proving to analyze theoretical results, verify desired properties, and evaluate results.

RESPONSE:

Thanks for the pros. We are happy that the work in the paper interests the readers. We hope that it provides a new and formal way to verify a real-time system, which is able to investigate important problems (such as periodic task scheduling) in a more realistic level.

COMMENT:

The weak point is the lack of evidence of practicality, efficiency and scalability. Since there are only small and simplistic scenarios to analyze this model, it is difficult to judge the effectiveness of the proposed implementation. The key scenarios appear conveniently small to allow the implementation fully supports all these examples, returning at the same time appropriate counterexamples to guide and adjust the design. Is this true for larger examples of "real world" applications within the field of industrial real-time systems? How do the authors propose to handle very large scenarios? How does it behave on typical real-time scheduling problems used in industry? I would like to see more complex examples to convince the reader of the practicability of the approach on the assumptions given for the model, and would probably increase the mentioned benefits. The paper does not give any directions to the above problems, and some sentences pointing out the difficulties in the implementation of more generic and realistic scenarios would be most welcome to understand the real contribution of the paper.

RESPONSE:

Thanks for the suggestions. As mentioned in the paper, the system under verification is a real-world RMS implementation, which is used in an avionic control system. The scenarios presented in Section V.B are also real scenarios from the online system. Our industrial partner has very strict timing requirements. They use at most 5 tasks for scheduling to avoid high overhead. They agree that the verification results presented in the paper satisfy their practical requirements.

On the other hand, as presented in the revised Section V.B, we have examined our approach by verifying randomly generated scenarios where the number n of tasks are over 10. In that case, the least common multiple of the periods of all tasks can be as large as 10^5 times the interrupt cycle T. All feasible combinations of execution traces are checked in our approach. Furthermore, the number n of tasks can be 20 and even more if we allow different tasks possess a same period.

COMMENT:

One of my biggest problems with the theoretical part of this work is that the paper is technically solid as far as I can tell,

but there's not much yet in the way of theorems and results about formal properties in Maude. The complete and detailed proof of the Theorem 2 should be provided somewhere. If the proof of the main results of schedulability and correctness of the model is straightforward (as is suggested) it seems not necessary to present a complex formalism on Real-Time Maude and all the theorems provided are natural consequences of Theorem 1. I agree that even non-surprising results need to be investigated in the proof of Theorem 2, some sentences pointing out the difficulties in the theoretical formalization in Real-Time Maude would be most welcome. Please, include (e.g., in an appendix or even in a technical report) a detailed proof.

RESPONSE:

We are happy that the readers are interested in the detailed technical proof, and we have now included it in the Appendix. It is true that readers may be interested in the detailed proof of Theorem 2 to feel convinced of the completeness of our approach. However, the proof needs a bit more theoretical background about rewriting logic and more technical details of our model, which we have simplified in the paper in order to ease the understanding of our approach. We would prefer to refer the detailed proof to a non-anonymous technical report, if possible, in the final version.

COMMENT:

Another important point which needs improvement is the description of the implementation. I guess other readers might have less difficulties with the motivations if they know Maude and the Real-Time Maude extension, but if you aim for a more general audience, it might not work. Many pages are spent for the description of the formal modeling of the implementation and the section is hard to read. I highly recommend the authors to thoroughly revise this part of the paper to make it easy-to-read. It would be better preferable to have a much shorter description of the principal insights for the implementation followed by more convincing examples and benchmarks with respect to related work to convince the reader of the practicability of the approach. You should try to summarize this section and to extend the original contribution. From my point of view, there is a trade off in this paper between the contributions and the presentation of the implementation.

RESPONSE:

Thank you for the invaluable suggestions. Section IV has been revised. Some unnecessary detailed definitions presented as code are removed, and replaced by more literal explanations. We hope that the current presentation would make Section IV more easy-to-read, even for an audience who did not know Real-Time Maude before. On the other hand, Section VI (Related Work) is also improved. We compare our work with the existing theoretical work and the existing verification work in a more complete way.

B. Detailed Comments

COMMENT:

Page 7, Related Work. More comparisons with other approaches based on model checking would help to evaluate pros and cons of an implementation with Real-Time Maude with respect to different languages and tools. The authors should review the comparison with [24] and [25] more thoroughly. You should benchmark your approach against others.

RESPONSE:

Thanks for the pointing this out. The related work section is now improved. More comparisons with the theoretical approaches and with [24,25] are added. The differences with [24,25] in the models and in the way of modeling are mainly discussed. On the other hand, it is a bit difficult to compare the efficiency and scalability between [24,25] and our approach. One reason is that we have a different setting with [24] and [25], since the objectives are different: we aim at verifying an RMS implementation from a real-world real-time system, while [24] and [25] targeted the RMS algorithm itself. Another reason is that [24] and [25] presented no discussion on the efficiency and scalability. Furthermore, they used `TMSVL` and an extension of SPIN, respectively, as verification tools, which are not open-source, making us unable to re-implement their work.

COMMENT:

Page 8, Conclusions. Where in the paper do you show that the details of your "realistic" implementation are "sufficient" for the behaviors of all real systems used in the current industry? This should be briefly discussed.

RESPONSE:

Sorry for the confusion. The statement has been corrected. The work presented in the paper aims at modeling and verifying a realistic RMS implementation in an industrial avionic system. It is a piece of real work in the industry. In this sense, "sufficient" means that the details described in our model meet the requirements and expectations from the users and the engineers. On the other hand, we believe that our approach could be applied to other similar systems as well.

III. Responses to Reviewer 2

A. General Comments

COMMENT:

One of the problems with this paper is that it is not completely self-contained. Some knowledge about Real-Time Maude is necessary in order to fully understand it. I found the short introduction about Real-Time Maude (section II.B) not enough for understanding the formalism used in the rest of the paper (see details below).

RESPONSE:

Thank you very much for pointing this out. The Sections II.B and IV are improved to be more self-contained now. The idea is that, Section II.B presents mainly an overview of Real-Time Maude, and then Section IV describes the model with introducing necessary syntax on-the-fly.

COMMENT:

Another concern is that the significance of the paper could be much better if the paper addressed the problem of modelling RMS in a more general way. Instead, the paper takes the

form of a case study, where the formal modelling of one specific RMS implementation is presented, rather than a general method for modelling and analyzing RMS implementations.

RESPONSE:

Thank you for the suggestion. Yes, we agree that we have two choices to do the work presented in the paper: one is to develop a general way to model RMS algorithms or implementations, and then instantiate the general model to get a concrete one for our target system; the other is to shape an accurate model of the target system directly.

We chose the latter according to the following considerations. By experience, *the development engineers (who write the code) and verification engineers (who verify the code or the system) are usually not the same persons.* When a verification engineer models a system, he is actually abstracting the system or the code. It is difficult to ensure that the model behaves the same as the system or the code. And it is more difficult to make other engineers and users believe that the model behaves the same as the system or the code. Furthermore, the developers and users hope that the model behaves not only the same as what they had in mind, but also the same as what is written in the code. A way out is to make the model not only *behave like* the code, but also *"look" like* the code. This requires the model to be as accurate as possible, based on a fact that the expressiveness of the modeling language is powerful enough. It makes the model specific, but improves the engineers' confidence in the model and the verification results. That is also why we emphasize the corresponding relationship between the functions in the model and the lines in the pseudocode, when we introduce our model in Section IV.

On the other hand, we are also interested in the first direction. This will be the future work and we would try to get a better balance between them.

COMMENT:

Also, the discussion about soundness and completeness of the analysis (section V.C) is not fully developed and, once again, it refers to the specific model rather than being a general result.

RESPONSE:

Thanks for this comment. The detailed proof of Theorem 2 is now included in the Appendix. On the other hand, since we chose the accurate way to achieve the model as discussed above, yes, the proof is a specific one.

B. Detailed Comments

COMMENT:

[Section II.B] In the definition of IR, it is not clear what s and s' are. I guess they are states, but the concept of state was not introduced.

RESPONSE:

Yes, they can be states. But more generally, they are terms. This part has been modified to specify s and s' (in fact, t and t' now) clearly.

COMMENT:

[Section II.B] In the definition of a class, the authors should specify that a class includes a collection of rules, otherwise this fact may be missed.

RESPONSE:

Thanks very much for reminding. This is added now.

COMMENT:

[Section III] Page 2 right column, line -24: The implementation is shown as schedule() in Figure 1 $\rightarrow$ The pseudocode of schedule() is shown in Figure 1

RESPONSE:

Thanks for the careful reading. It is done now.

COMMENT:

[Section IV.A] I don't understand the meaning of `[ctor]` in the definition of some operations (maybe because I have no specific background on Maude). I could not find the explanation of this writing in the introduction to Maude.

RESPONSE:

Sorry for missing the explanation of `[ctor]`. It is now added when `[ctor]` is used for the first time.

COMMENT:

[Section IV.A] For the stack sort, the authors mention operations push, pop and peek, but then you don't write the definitions of these operations. Instead, they write the definitions of bottom and #, which are not explained in the text. This part should be made clearer.

RESPONSE:

Sorry for the misleading presentation in the text. In fact, `bottom` and `#` are constructors of sort `Stack`. This part is modified now. Some detailed definitions in code are removed, and are replaced by more explanations. We hope that the current presentation would make Section IV more easy-to-read.

COMMENT:

[Section IV.A] Also, the meaning of NzNat is not explained.

RESPONSE:

Sorry for this incompleteness. `NzNat` meant non-zero natural numbers. `NzNat` is now replaced by `Nat` to reduce unnecessary complexity for the readers.

COMMENT:

[Section V.C] In the statement of Theorem 1, the definition of time-robust real-time rewrite theory is missing. Also, the meaning of tick-stabilizing atomic propositions is not defined. This makes the section not self-contained.

RESPONSE:

Sorry for the inconvenience for reading. In fact, the definitions of time-robustness and tick-stabilization (which is now replaced with tick-invariance) require more knowledge about rewriting logic. We avoid introducing the accurate definitions of them. Instead, now we give short descriptions of them before Theorem 1. More detailed introduction can be found in the Appendix.

COMMENT:

[Section V.C] The proof of theorem 2 is missing. The authors just indicate how the proof could be developed, but they do not develop it. A possibility would be to point to a document where this proof has been developed.

RESPONSE:

We are very happy that the readers are interested in the detailed and technical proof, which is now included in the Appendix. The detailed proof requires a bit more theoretical background about rewriting logic and more details of our model, which we tried to simplify in the paper in order to ease the understanding of our approach. We would prefer to refer the detailed proof to a non-anonymous technical report, if possible, in the final version.

IV. Responses to Reviewer 3

COMMENT:

RMS has many different realization methods, no comparison is given to justify the strong point of real-time maude when applied to RMS.

RESPONSE:

The comparison could be found in Section VI (and we have expanded according to the reviewers' comments :-)). We would like to clarify that we are *not implementing* RMS, we are actually *verifying* it. We choose Real-Time Maude since it verifies the model using model checking technique. If verification passes, no deadline will be missed in the implementation as long as the model assumptions are met.

COMMENT:

Rate-monotonic scheduling is a very simple scheduling and has been sufficiently investigated in the field of embedded system, the authors claim the innovation of this paper as the implementation using real-time maude, however, I cannot see any special point (or advantage) when using this real-time maude in modelling RMS.

RESPONSE:

Thank you for the insightful comment. We agree that RMS has been intensively studied because of its importance. However, most of the results are based on the assumption that *task switching does not take time*. That assumption is not realistic. Models with and without the assumption have totally different behaviors. For instance, as shown in Figure 2(b), in the RMS implementation we verified, the switching from time 9 to 11 blocks the interrupt handling, hence delaying the initiation time of τ_1. This kind of phenomenon, to our knowledge, would not happen in the theoretical analysis models. Although there is very few results that take into account the switching overhead (as discussed in Section VI), none of them is generic and our system under verification is not in their scope.

Despite the above reason, the most important motivation of this paper is to verify the system not only for the schedulability, but also for the correctness. That is, we want to verify that the target system does schedule the tasks under the RMS algorithm. The work presented should be seen as a verification problem, instead of a schedulability test. From verification point of view, this paper models a real-world RMS implementation with sufficient technical details. Two important properties–schedulability and correctness–are verified by model checking technique, and the soundness and completeness of the results are demonstrated.

COMMENT:

The assumptions given in IV is too rigid, which makes the scheduling problem to be investigated by very simple, then what is the difficulty?

RESPONSE:

Thank you for the question. When model checking technique is used to verify a given system, the tool (model checker) explores the whole system state space to check whether the given property holds. All feasible combinations of execution traces are checked. As a result, the state space explosion is the most common problem for a model checker. That means the state space of the system/model is too huge for the tool to explore with acceptable temporal and spatial cost. The scale of the state space depends on the complexity of the model of the system.

As an industrial verification application, we are not trying to make the problem as difficult as possible. Instead, we are trying to simplify the model so that the existing tools are able to solve, while the target system satisfies the assumptions on the model so that the verification results are trustworthy.

On the other hand, the assumptions are actually not that rigid. For example, the mask mechanism considered makes the job initiation times of task τ_i possibly not equal kT_i, implying that the deadlines of the jobs may be not equal to $(k+1)T_i$ (see assumption A1'). This is also discussed in the beginning of Section IV, pointing out the blocking at time 10 in Figure 2(b). Another example is that non-determinism exists in our model, as discussed in Section VI.

COMMENT:

The description in section IV (A-E) is trivial, it can only be viewed as a simple application of the rewriting logic.

RESPONSE:

Thanks for the careful reading. We agree that this is an application of rewriting logic. In this paper, we do apply the existing theories and techniques of rewriting logic to solve a real industrial problem, which has not yet been investigated by this sort of techniques, achieving an industrial contribution.

COMMENT:

The model checking part is not sufficiently discussed, the authors only list the commands but not illustrate the mechanism behind these commands.

RESPONSE:

Thank you for the suggestion. Our work does focus on how to model a realistic industrial system and how to apply formal techniques (such as model checking) to verify important properties of the system. The mechanism behind the model checking commands (i.e. of a model checker) is another issue.

The field of model checking techniques and the implementation of model checkers have been intensively studied for decades. If the reviewer is interested in the mechanism of

the model checker provided by Real-Time Maude, we would recommend the reference [A] to the reviewer:

[A] S. Eker, J. Meseguer, and A. Sridharanarayanan, “The Maude LTL Model Checker”, *Electr. Notes Theor. Comput. Sci.*, vol. 71, pp. 162-187, 2002.