主 题： IEEE Transactions on Reliability - Decision on Manuscript ID  
TR-2022-452.R1

15-Mar-2023  
  
Dear Prof. Jiang:  
  
Thank you for submitting the following manuscript  
  
TR-2022-452.R1 entitled "Simulink Compiler Testing via Configuration Diversification with Reinforcement Learning"  
  
to the IEEE Transactions on Reliability. It has been  
reviewed independently by experts in this area.  
Based on their comments, the paper requires a minor revision.  
  
Please prepare a revised version in response to these  
comments and a separate list detailing how each comment  
is addressed in your revision. The paper will be  
re-evaluated by the same reviewers. Please note that it is TRel's policy that a submission including appendix and supplementary materials cannot exceed the 15-page limit.  A manuscript over 15 pages will be rejected without further review. This is a mandatory requirement in accordance with TRel policy.  
  
To upload your manuscript material, log into  
https://mc.manuscriptcentral.com/tr-ieee and enter your  
Author Center, where you will find your manuscript title  
listed under "Manuscripts with Decisions."  
Under "Actions," click on "Create a Revision."  
Your manuscript number has been appended to denote a revision.  
  
You may also click the below link to start the revision process  
(or continue the process if you have already started your  
revision) for your manuscript. If you use the below link  
you will not be required to login to ScholarOne Manuscripts.  
  
\*\*\* PLEASE NOTE: This is a two-step process. After clicking on the link, you will be directed to a webpage to confirm. \*\*\*  
  
https://mc.manuscriptcentral.com/tr-ieee?URL\_MASK=389db91ca32a4bfbb6b3d1eb72c224d0  
  
Because we are trying to facilitate timely publication of  
manuscripts submitted to the IEEE Transactions on Reliability,  
your revised manuscript should be submitted by 14-Apr-2023.  
Otherwise, we may have to consider your paper withdrawn,  
moved to a later queue, or begin the peer review process anew.  
  
We look forward to your re-submission.  
  
Sincerely,  
  
Prof. Sudipto Ghosh  
Associate Editor  
IEEE Transactions on Reliability  
  
============================================  
Associate Editor's Summary  
  
Two reviewers are happy with the changes but one reviewer would like to see further clarifications. Please address the points made regarding efficiency, contribution by DDQN and the importance of bugs in your next revision. Thank you!  
  
============================================  
Reviewers' Comments to Author:  
  
Reviewer: 1  
  
Comments to the Author  
Thank the authors for carefully addressing my concerns. And I believe this paper is ready for publication.  
  
Reviewer: 2  
  
Comments to the Author  
The paper addresses an important problem of stress testing CPS compilers which are used in many embedded systems. The paper has solid motivation and impact.  
  
The paper also found an important lever to further optimize the SOTA fuzzer, which is also a solid contribution.  
  
I do not have doubts about the impact/contribution of the research. However, I think some relevant information must be added to help understand the methodology and completeness.  
  
0) [Please consider having this discussion] Efficiency. The paper does not discuss much about efficiency. It seems like the time needed to complete the test is indeed pretty long, so would be great to understand in more details. I see some response from the authors to previous questions around this topic, but I do not see data in the paper (I see that the tools authors are citing e.g. the fuzzer or SLEMI go into some details by breaking down time needed for components). What is the bottleneck? Will a simpler technique than DDQN speed things up by order of magnitudes? Will parallelization help (e.g., looks like the SLEMI paper discussed/incorporated parallel evaluation; would it also help here to use a cluster of machines instead of two computers to find more bugs?)

**Response:**

Sincerely thank you for your positive comment.

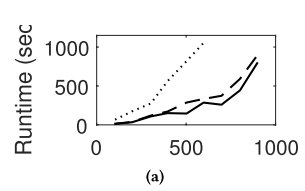
**2.1 Response to the comments by Reviewer 2:**

效率问题：首先，我们实验时间耗费主要由三个部分组成，分别是有效模型生成时间(从选择块到修复错误， SLforge中完成)、模型仿真时间、以及从模型获取特征向量并计算的时间。此外，模型包含的模块越多，生成器生成模型所需的时间也越多，具体如SLforge所展示的生成有效模型时间随模块个数的变化图，其中实线表示SLforge的变化曲线。在我们的实验初期，我们使用包含五层模型并且模块数量超过两千的模型来进行测试，因此耗费了大量的时间，在随后的实验中，我们逐步缩小了生成模型的规模，我们生成1-3个层级的模型，并且顶层模块的个数为[30-100]。其次，在对比实验中，对于每个对比方法，都需要利用生成器生成模型并根据对比算法改变生成器配置。具体不同配置下生成模型所占用的时间如下图所示，除此之外，还包含模型仿真时间，以及Bug模型约简所耗费的时间，综合上述所有因素，导致了我们测试周期较长。

其次，影响效率的瓶颈主要在于，当模型生成器在生成模型的时候，需要考虑CPS模型的半形式化规范，简单来说，在生成模型前，生成器会根据半形式化规范，检索可用的模型和连接，并配置每个模块的参数和端口类型。当模型生成失败时，生成器需要通过多次迭代试图去修复错误，并且在每次生成模型之后会进行编译仿真验证模型是否有效，而对模型进行编译和仿真都需要一定的时间步来进行。因此模型的编译和仿真会占用大量的时间，这取决于CPS语言自身的特点。

除此之外，从CPS模型中提取模型特征也需要一定的时间，我们通过Simulink自身的API来获取和计算CPS模型中的部分特征，这部分特征的计算取决于方法执行的效率以及模型的复杂程度。因此在一定程度上也会影响到测试的效率。在未来的工作中，我们将会对此进行改进。

对于并行计算方面，SLEMI在生产模式下，对模型的变异(和缓存)是并行进行的，也就是说在获取源数据后，SLEMI可以同时生成多个源模型的变体，而不需要串行生成。然而对于我们的强化学习组件配置方法来说，我们需要先获取当前配置下环境的状态，然后才能利用学习到的知识去更改配置，因此，在配置学习一过程我们可能无法实现并行学习，但是对于模型生成阶段，可以通过并行计算来加快模型的生成过程。当然，在以后的工作中，我们可能会在模型特征提取阶段实现并行计算，从而加快从CPS模型中提取特征向量的过程。



1) [Highly recommended] Contribution by DDQN: It's interesting that authors are using RL and DDQN, but a discussion of other RL (e.g., the non-model based ones) would be good to have. Have the authors already tried other approaches? Why do we think DDQN is a good fit (from experiment or intuition). What are the limitations (could be discussed in the threats section)

**2.2. Response to the comments by Reviewer 2:**

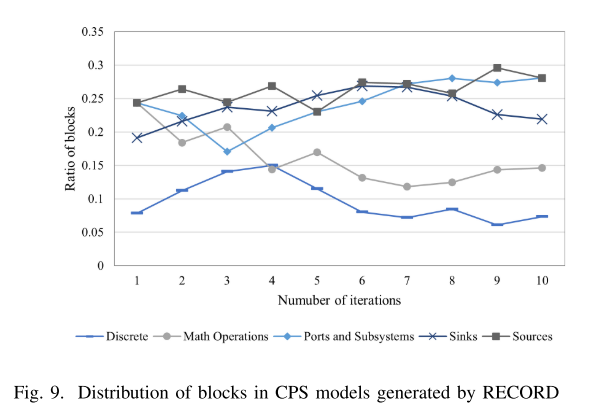
DQN作为深度强化学习其中的一种算法，结合了深度学习和强化学习的优点，直接从高维原始数据学习控制策略。将神经网络和Q-Learning结合起来，输入原始数据(作为状态State)，输出则是每个动作Action对应的价值评估Value Function(Q值)。DQN解决了深度学习需要大量标签进行监督学习的问题，其次，DQN通过构建experience replay（经验池）的方法来解决数据相关性和非静态分布的问题。最后，DQN使用两个网络，一定程度降低了当前Q值和目标Q值的相关性，提高了算法稳定性。而DDQN作为DQN的一种改进，解决了DQN中可能出现的过度估计的问题（也就是估计值比现实值要大，导致算法会将次优当成最优）。综合上述DDQN的特点，我们采用了DDQN作为我们强化学习组件使用的算法，并且在实验三(RQ3)中验证了DDQN对我们测试方法的有效性。

在这个RQ中，我们分析了RECORD在每次迭代中生成的不同库中块的分布，以研究强化学习组件的影响。

我们运行RECORD来为一个episode生成具有不同配置的CPS模型。在每次迭代中，我们收集一组由RECORD生成的CPS模型，并计算这些CPS模型中每个库中的块的比例。由于SLforge只支持CPS模型库[1]的一个子集，我们主要分析了该RQ中与离散库、数学运算库、端口和子系统库、汇点库和源库相关的块的比例。通过观察模型中相关块的分布变化，来探究DDQN对实验的影响。

如图9所示(论文里好像写的如图1 ,不知道是不是写错了)，RECORD尝试生成不同块分布的CPS模型。例如，最初的sink块的比例是0.19。在配置学习过程中，RECORD将sink块的生成概率提高到0.27，以探索更多的sink块的输入空间。然而，经过几次迭代后，RECORD发现添加更多sink块的奖励正在减少。RECORD然后配置生成器以减少生成sink块的概率。因此，sink块的比例下降到0.22左右。我们可以观察到其他库中块的类似增加或减少(或相反)趋势。例如，在本实验中，RECORD倾向于首先生成端口和子系统块较少的CPS模型。由于这可能导致CPS模型简单，多样性较低，RECORD增加了Ports和子系统块的比例，增加了CPS模型的复杂性。这样，RECORD可能有更高的概率生成复杂的CPS模型，从而触发更多的Simulink编译器错误。

根据以上观察，RECORD通过动态调整不同块的分布来探索Simulink编译器的输入空间。通过这种方式，RECORD可以有效地探索输入空间，而不会生成太多“无用的”CPS模型，例如只有大量sink块的CPS模型。



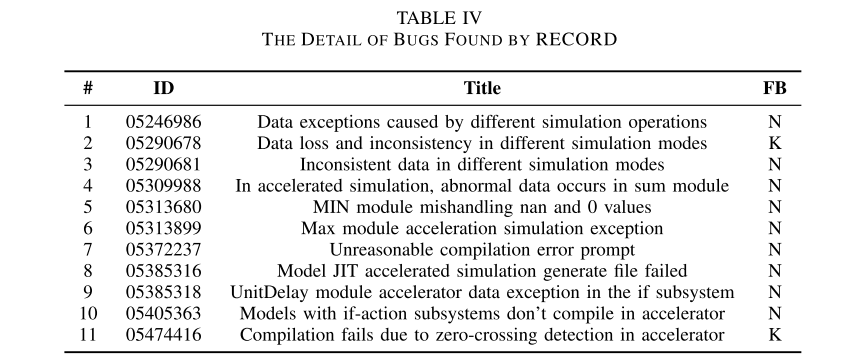
(2) Importance of the bugs. It's often discussed in the fuzzing literature how important these found bugs are. It would be good to have something around that. Another thing unclear was how many of these reported bugs have been confirmed by the developers and were fixed.  
Thank you again for the important work. I hope authors would consider having these discussions.

**2.3. Response to the comments by Reviewer 2:**

bug的重要性。在模糊的文献中经常讨论这些发现的bug有多重要。如果周围有东西就好了。另一件不清楚的事情是，这些报告的错误中有多少已经被开发人员确认并修复了。  
在Matlab官方Bug报告中，将发现的Bug分为安全问题、不正确的代码生成、辅助功能问题以及其他问题(包括Matlab崩溃等)四种类型,大部分的问题可能会导致模型仿真得到的结果与实际不一致，或者导致生成的嵌入式代码发生严重错误，如果这些是被用于医药以及航天等安全领域，会造成无法挽回的损失。



其次，Matlab会在每个最新版本中修复部分已发现的错误，并在Bug报告中描述错误说明和解决方法。可以通过三种方法来确定bug是否被修复，一种是由开发人员邮件通知。另一种更为有效的方法是在最新版本的Maltab中运行Bug模型，观察是否能复现bug。其次，还可以在官方Bug报告中查看是否有相关Bug描述信息。我们在实验中发现的Bug均是提交给开发人员并被确认的Bug。具体如下图所示，FB 表示是否是新的Bug。其中是否为新的Bug由开发人员反馈得知。



如下图所示，开发人员邮件反馈通知05474416已经在Matlab R2022b版本中解决了加速模式下过零检测异常的问题。

  
  
Reviewer: 3  
  
Comments to the Author  
I read through the paper and feel all my concerns were addressed, and the response nicely locates each change made in response.  
  
============================================