Reply letter to #CPE-16-0056

“SMT-Based Context-Bounded Model Checking for CUDA Programs”

april 12, 2016

Dear Editor,

We really appreciate the comments of the JBCS reviewers, which were highly insightful and enabled us to greatly improve the quality of our manuscript. The following pages list our responses to each of the comments.

Attached to this letter, please find a revised version of our manuscript, in which revisions in the text are shown using green color.

We hope that the revisions in the manuscript and our responses will be sufficient to make our manuscript in suitable for publication the journal of Brazilian Computer Society.

Best Regards,

Renato Abreu, Mikhail Ramalho, Lucas Cordeiro, Eddie de Lima Filho, and Waldir da Silva Jr.

Federal University of Amazonas (UFAM), Brazil,

E-mail: [renato.abreu@indt.org.br](mailto:renato.abreu@indt.org.br), [mikhail.ramalho@gmail.com](mailto:mikhail.ramalho@gmail.com), [lucascordeiro@ufam.edu.br](mailto:lucascordeiro@ufam.edu.br), [eddie@ctpim.org.br](mailto:eddie@ctpim.org.br), and [waldirsabino@gmail.com](mailto:waldirsabino@gmail.com).

Referee(s)' Comments to Author:

Dear authors,

Reviewers have now commented on your paper. You will see that they are advising that you revise your manuscript. If you are prepared to undertake the work required, we would be pleased to reconsider our decision.

Your revision should be submitted before June 15th, please notice that as this is a special issue there will not be an additional reviewing round.

Sincerely,

Alfredo, Edward and Luciana

Reviewer: 1

Comments to the Author

This paper extends the ESBMC bounded model checker to support CUDA constructs and simulate the execution of CUDA programs in the CPU. Specifically, it models CUDA APIs using usual CPU functions that can be processed by the existing ESBMC checker, and turns GPU threads into CPU threads that are examined and optimized by the ESBMC checker.

This paper's main contributions include: (1) a set of CUDA API models based on CUDA's semantics, (2) the application of a partial order reduction algorithm to curb the thread interleaving explosion issue. In addition, the authors performed a quite comprehensive comparison of their tool ESBMC-GPU with other existing GPU verifiers and checkers.

It is interesting to see how to extend a Pthread-based bounded model checker to handing a different parallel programming model like CUDA. In general, the extension needs to address two main challenges:

(1) model the data structures and the APIs in the new programming model;

(2) resolve the difference between PThread and CUDA (i.e. handle the inherent characteristics of how threads are executed and synchronized).

It seems to me that the authors did a good job in (1), e.g. the model of dim3, cudaMalloc, cudaMemcpy, and cudaFree. The model is written to cater for the need of the ESBMC checker. With the model, ESBMC can check CUDA programs like Pthread programs.

However, the second challenge is handled less swiftly. In the proposed method, each CUDA thread is mapped to a Pthread (in ESBMC). This is fine since the official CUDA simulator uses a similar approach. The main problem is on thread synchronization. CUDA uses lock-step execution within warps (as mentioned in the paper on Page 12), and barrier synchronization within blocks. In contrast, Pthread uses schemes such as mutexes, wait-notification, etc. Bridging the gap between these two different synchronization schemes is tricky. Unfortunately I didn't see how the authors address this gap carefully. They should give some formal descriptions and examples to illustrate that ESBMC-GPU can preserve the CUDA's synchronization scheme precisely.

The proposed Monotonic Partial Order Reduction (MPOR) method deserves more investigation and details. First of all, an existing GPU checker GKLEE checks whether the threads access the same memory location and uses this information to execute only one thread interleaving called canonical schedule. It won't incur any unnecessary thread interleavings, while the proposed MPOR may still explore these interleavings. In other words, GKLEE has implemented, with respect to the CUDA model, what MPOR try to achieve. So do other GPU verifiers like GPUVerify and PUG (which explore one thread interleaving of two symbolic threads).

Second, it is unclear how new the MPOR is. It seems to me that the authors directly reuse the MPOR algorithm already included in ESBMC [23].

Third, while GKLEE and other existing tools explores only one thread interleaving, it is unclear how many interleavings are explored by ESBMC+MPOR although the authors show that the performance is boosted significantly with the MPOR optimization.

The experimental result part is perhaps the most valuable one in this paper. Experimental results show that ESBMC-GPU is able to detect more properties violations than other existing GPU verifiers, while keeping lower rates of false results. This indicates that the proposed approach merits being published. However this part can be improved as well. Particularly, the authors of this paper should contact the authors of other tools to make sure (1) the right configurations are used; (2) the assumptions for these tools to run properly are satisfied; (3) confirm that the findings (e.g. the false positives and false negatives) are real. For example, Page 15 mentions "GKLEE has 24 benchmarks that were not supported, which are due to the use of the memset function, which is a specific function of the C/C++ libraries".

In fact GKLEE supports most of the C/C++ library functions; but the option "-libc" may need to be set properly so as to link the definitions of the C/C++ library functions including memset. With this, GKLEE supports all the benchmarks while the proposed ESBMC-GPU doesn't support 23.

Finally, the presentation of the paper can be improved. Despite including the new materials, it is still in a workshop format. Specifically, more space should be given to the CUDA synchronization modeling and the detailed description of MPOR.

--------strength------------

\* Reuse the existing ESBMC to perform bounded model checking on mutli-thread programs

\* Provide a set of CUDA API models for ESBMC

\* Evaluate extensively the tool and compare it with other existing tools

\* Achieve good evaluation results

--------weakness------------

\* Do not describe how many CUDA APIs are modeled and how other important APIs are modeled

\* Do not address the thread synchronization scheme in CUDA

\* Do not elaborate the MPOR mechanism

\* Lack of awareness of existing tools' one-schedule mechanism (that beats MPOR)

\* Inconclusive experimental comparisons with existing tools

Reviewer: 2

Comments to the Author

In this paper the authors present ESBMC-GPU, which is an extension to the Efficient SMT-Based Context-Bounded Model Checker (ESBMC) tool aimed at verifying SMT programs written for the CUDA framework. This paper is an extension of a previous paper published in a conference, and the authors claim that the version of ESBMC-GPU presented in this paper has been optimized and extended when compared to previous versions. All in all, the paper is well written and the results presented are consistent and sound. Yet, the detailed description of the tool and the writing of the paper can be improved to better reflect the optimizations presented in this work. Some comments and suggestions are below:

- The abstract can be considerably improved. The contributions of this paper should be highlighted at the abstract.

- The authors should mention, in the Introduction, that the ESBMC-GPU tool is available for download. The authors mention this possibility in a footnote at section 4.1, but it is important to highlight this possibility at the Introduction.

- At subsection 2.1, the description of ESBMC should be augmented and improved. Only to illustrate this point, the authors do not even clarify what the acronym “VC” means.

- It is important, to improve readability, to include a figure presenting ESBMC-GPU architecture. It is also important to use this figure to improve the discussion on architectural differences to the original ESBMC tool.

- The reference section can be considerably improved. For instance, seminal works on “Bounded Model Checking” and “Satisfiability Module Theories” (section 1) should be referenced. The authors should also include references for the SMT solvers listed in subsection 4.2.

- The legends for tables I e II should explain what the numbers in the tables mean.

Reviewer: 3

Comments to the Author

Summary:

The paper proposes a context-bounded model checking for CUDA programs. In particular,

the paper leverages on the advances of satisfiability modulo theory (SMT). Compared

to previously proposed approaches, this work could verify more properties in CUDA

programs chosen from a standard pool of benchmarks. It also shows the application of

monotonic partial order reduction (MPOR) in verifying GPGPU programs.

Reviews:

The paper is oriented towards a tool. However, such publications are encouraged for

the journal. Specifically, the work shows the application of several well-known

model checking techniques, such as SAT-based and explicit state-space based model

checking, for CUDA programs. The paper does not propose any novel technique. However,

it has designed a tool, which is evaluated well and it has also been made available

in public. I appreciate such effort by authors.

The organization of the paper is somewhat very informal. The current presentation

mostly revolves around showing code snippets. However, there is a lack of formal

presentation, especially for a paper that discusses model checking and verification.

It would be good to highlight, very concretely in a separate section, the reasons

for the tool being more effective compared to any other existing verifiers for

CUDA programs. Is it due to the operational model being rich for ESBMC-GPU? Is it

due to the encoding via SMT solvers? I would suggest to take a simple example of

CUDA programs and show (with sufficient detail) the reason for ESBMC-GPU being

more effective than other existing tools. Whereas the evaluation is promising, but

we need a general idea of the reason for such positive evaluation.

Authors use "pthread\_create" to use different threads for CUDA programs. Whereas

CUDA threads are different in the sense that they operate in lockstep fashion

within a warp (i.e. the SIMD feature of GPU). It is unclear whether warp-level

threads in GPU can be simulated by "pthread". Authors should provide sufficient

justification that such a transformation via pthread does not affect the soundness

of the verifier.

Since the approach described in the paper is quite informal, I could not see

any proof regarding the soundness of the operational model. I do not think it

is straightforward. Authors could try to provide some sketches of proof for

the soundness.

Authors should provide some summary of these benchmarks in the paper. I know

that the benchmarks are available in the website. However, providing some

salient features of these benchmarks in the paper will help improve the quality.

A minor comment: Why do we even need Figure 11? I think it's trivial, unless it

is expanded enough with some details of two-thread analysis.