C. ARTHO

KTH ROYAL INSTITUTE OF TECHNOLOGY
LINDSTEDTSVÄGEN 3, LEVEL 5
SE-100 44 STOCKHOLM
SWEDEN

C. Artho · KTH Royal Institute of Technology · SE-100 44 Stockholm · Sweden

To whom it may concern

YOUR REF.	YOUR MAIL	MY REF.	DATE
		CA	September 17, 2019

Evaluation of research activities of Pavel Parizek

To whom it may concern,

I have been contacted by the administration of Charles University to act as an independent evaluator of Pavel Parizek's research activities. Pavel Parizek applied for tenure and promotion to Associate Professor.

Currently, I work at the KTH Royal Institute of Technology, where I have been Associate Professor since 2016. Before that, I have worked as Research Scientist (2007–2012) on a tenure-track position at the National Institute of Advanced Industrial Science and Technology in Japan, where I was promoted to Senior Researcher (2012–2016).

I have met Pavel Parizek at a the SV-COMP workshop 2019, which was co-located with the ETAPS conference in April in Prague. Even though we have both carried out research on the Java Pathfinder platform (JPF) and have published several papers at the JPF workshop, we had never met in person before, due to having published in alternating years at said workshop. We have also had limited interactions as part of the Google Summer of Code program, as part of reviewing student proposals, by electronic communication.

As an evaluator, I am asked to judge Pavel research based on his publications.

Pavel's research addresses shortcomings in software model checking, and he has found several ways to improve the performance of state space analysis. The habilitation thesis has includes nine publications on this topic. The publications have all been published in well-known conferences (ASE, VMCAI, FMCAD, SAC, TACAS), workshops (SPIN), or journals (STTT). The papers are grouped according to two logical sub-themes:

1. Improvements in partial-order reduction (POR). The key to an efficient state space analysis is being able to ignore parts of the program schedule state space that does not give any new results. This is achieved by ignoring interactions between threads at places where one thread does not influence other threads. Such an analysis is implemented in Java Pathfinder (JPF), and it works by analyzing the current field access, as well as the state of other threads. It is therefore based on the *history* of program execution. Pavel's work is interesting and novel and that it combines that information with possible *future* program executions. These are approximated by static analysis. While static analysis is less precise than dynamic analysis, it is precise enough to yield good approximations of future field accesses, which in turn brings significant improvements over the existing POR. The first five papers show a gradual development of such POR algorithms, which includes the analysis of future field accesses (Ch. 4, 5), array elements (Ch. 6), inter-thread signaling (Ch. 7), and an integration that has a more detailed approximation of future field accesses (Ch. 8). Pavel's work therefore systematically covers gaps in the existing approaches and tools, and addresses their shortcomings.

Phone: +46 76 781 22 70
e–Mail: artho@kth.se
HTTP: https://people.kth.se/~artho/

2. Fast error detection within a large state space. Large programs are too complex to be analyzed fully; it is therefore important to be able to find faults early during analysis, if the entire state space is too large to be analyzed completely. The second part of Pavel's work is based on the intuition that a good analysis strategy mixes breadth and depth. Going too far into one direction is very risky, as this spends a lot of resources on that part of the state space at the expense of other parts. Randomized strategies that jump back to an earlier part of the program, to try out different outcomes, are the subject of the first two publications in that subfield (Ch. 9, 10). The next publication covers the problem of generating test drivers (or environments), where similar strategies are applicable at the level of operations that are chosen to execute the system under test (Ch. 11).

Finally, the last chapter integrates both parts of the work and brings together two rich subfields in dynamic program analysis. What impressed me that all papers combine theoretical advances with practical (tool) implementations of the idea, to show empirically that the idea translates to situations found in real programs. Producing a tool that can analyze real code is a significant effort. This effort is worth it, though, as only empirical experiments yield insights about practical aspects that would be absent when deriving the algorithm only as a concept. It is therefore well-deserved that the results were accepted for publication at high-profile research conferences. I hope that future advances will also include an integration with the current version of JPF, which is understandably difficult because Pavel's work also depends on another analysis tool, WALA.

To summarize, Pavel's research addresses important problems in the domain of software model checking, where it is well-known that a complete analysis is often impossible. Good partial-order reductions enable a more comprehensive analysis in cases that were impossible without it, and search heuristics maximize the chances of finding a defect quickly, or at all, in large programs.

I therefore recommend Pavel Parizek for tenure and promotion.

Yours sincerely,

Cyrille Artho, Ph.D.