Recreating the algorithms proposed in "Reducing Context-bounded Concurrent Reachability to Sequential Reachability"

Ioannis Xarchakos

University of Toronto

Course: CSC2226

1 Introduction

In this work, we recreated the algorithms proposed in [1]. [1] proposes two algorithms for reducing concurrent programs to sequential. We implemented both algorithms (Eager and Lazy). We utilized the bit permutation experiment performed by the authors, to experiment with the two algorithms. For the implementation of the Eager and Lazy algorithms we used the description of the algorithms in Sections 4 and 5, respectively, as well as the algorithmic representation in Figures 2 and 4. For the bit permutation experiment, we utilized the description of Section 3, page 8, as well as the description of the bit permutation experiment in Section 6.

2 Eager Approach

The implementation of the eager algorithm follows the implementation in Figure 2 of the paper. Additionally, we implemented the firstContext and nextContext functions, using the description provided by the paper in Section 4. Furthermore, we implemented the control code (Section 4) and

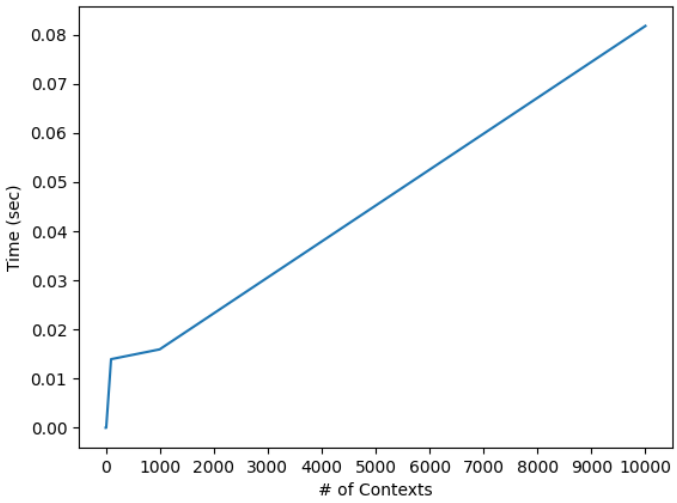

Figure 1: Eager algorithm

invoked it after shared values are read/written. As for the threads' implementation, the values of the 4 bits in thread2 are selected randomly (between the values of True or False), similar to what the authors did for the 16 bits experiment. Figure 1 shows the time performance of the Eager approach when varying the number of contexts of each thread. The time performance of the eager algorithm is much faster than the performance the authors reported in the paper. For example, for 3 context-switches, the authors reported that the algorithm could not finish due to an out of memory error while for 10000 context-switch we report 8 seconds. There are several reasons for why our results differ from the results reported in the paper. First, we implemented the 4 bit version instead of the 16 bit version. The authors reported the results of the 16 bit version in the paper. Second, our hardware (8 CPU cores, 128 GB memory) is probably far superior than the hardware used by the authors. Finally, bugs in our implementation may contribute to this result.

3 Lazy Approach

The implementation of the lazy algorithm follows the implementation of the algorithm in Figure 4 of the paper. In the implementation of the lazy algo-

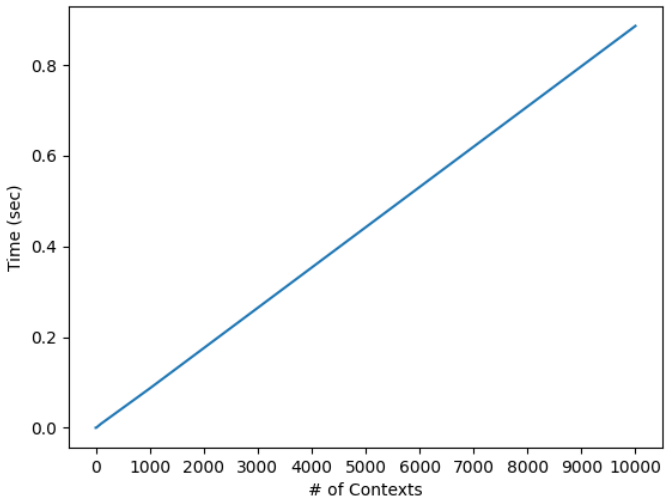

Figure 2: Lazy algorithm

rithm, we also invoke the control code after shared values are read/written. The implementation of the threads 1 and 2 is similar to the implementation found in Figure 1 of the paper. Figure 2 shows the time performance of the Eager approach when varying the number of contexts of each thread. Similar to the eager performance, the lazy algorithm's performance is superior to the performance of the algorithm reported in the paper. We believe that the same reasons (reported above) that affect the eager algorithm also affect the lazy algorithm.

References

[1] S. La Torre, P. Madhusudan, and G. Parlato. Reducing context-bounded concurrent reachability to sequential reachability. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification*, pages 477–492, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.