Loop Quasi-Invariant Peeling

Abstract presented at the 2020 Georgia Undergraduate Research Collective (GURC) Conference

Assya Sellak
Advisor: Clément Aubert

Computer programs are written in high-level languages and translated into machine-code using compilers. Compilers perform a series of program transformation and optimizations to improve memory usage and reduce the run time of the program execution. Programs consist of commands and statements such as conditionals and loops. Loops are an extremely powerful tool for programmers used to repeatedly run a sequence of commands until a specified condition is met. However, when used carelessly, loops can lead to never-halting or extremely slow programs: for this reason, many compilers and program optimizations focus on these structures. Loops can contain commands that perform unneeded residual operations instead of only being executed when necessary. This excessive performance results in an avoidable increase in run time which may arise intentionally or unintentionally either because of the programmer or other automatic transformations. Detecting which operations could have been performed fewer times than the loop requires is complex, but some optimizations try to detect this and extract portions of code that only needed to run once and successively move commands that need to run more than once, but not as many times as the loop runs. These fall short on some structures, mainly because they limit the scope of the analysis to individual operations, rather than considering sequences of operations as a whole. Thanks to quasi-interpretation [1] coming from Implicit Computational Complexity, new ways of detecting invariant sequences of commands inside loops have been developed. We extend this work along two axes: We allow for more structures, including `for`, `do...while`, loops with `break`, to be peeled. By analyzing the dependencies within the loop, we hope to allow for some parallel optimization. This allows to:

- consider more programs
- possibly significantly speed-up programs that are distributed, i.e., executed in parallel on multiple computers.

References

[1] J. Moyen, T. Rubiano, T. Seiller, Loop quasi-invariant chunk detection, in: D. D'Souza, K.N. Kumar (Eds.), ATVA, Springer, 2017. https://doi.org/10.1007/978-3-319-68167-2_7.