May-Happen-in-Parllel Analysis for C Programs*

Ian J. Hayes[1], Daniel Wainwright[1], Kirsten Winter[1], Chenyi Zhang[2]

[1] School of ITEE, University of Queensland, Australia
[2] Oracle Labs, Brisbane, Australia

In our context the aim is to detect data races in large C programs. To be effective and usable, data race detection requires three main building blocks: an analysis on shared variables and their accesses, a lockset analysis to determine which locks are held at each access point, and an analysis to distinguish between pairs of shared variable accesses that may execute in parallel from those which do not occur in parallel, namely a May-Happen-in-Parallel (MHP) analysis. MHP is one of the fundamental analyses needed when analysing concurrent programs as it is essential when pruning the number of false positives reported.

To precisely compute the set of all parallel instruction pairs is known to be NP-complete [Tay83], and hence MHP algorithms can only provide an over-approximiation. Related work is predominantly based on iterative data flow analysis, an approach that provides fairly precise results but does not seem to scale to large code bases, or is restricted to programming languages supporting specialised synchronisation features. Some results can also be found within the works on analysing data races in concurrent programs but these approaches focus mostly on the shared variable and lockset analyses.

A notable exception is the work by Barik [Bar05] on MHP analysis for concurrent Java which uses an abstract hierarchical data structure to capture the transitive spawn relation between threads in a program. This approach is closest to ours but [Bar05] presents only a complex informal description of their approach which seems incomplete in places.

Since we are targetting C code in which synchronisation is not lexically scoped flow- and context-sensitivity become important. To maintain scalability, however, only a limited degree of sensitivity is feasible and abstractions become vital. We propose a formalisation of the MHP relation in stages which stepwise introduce the abstractions applied to the problem. This formalisation provides us with a better understanding of the problem, showing where abstractions limit the flow- and context-sensitivity of the approach, and leads to an indication as to where the problem can be localised.

References

[Bar05] R. Barik. Efficient computation of May-Happen-in-Parallel information for concurrent Java programs. In *Int. Workshop for Languages and Compilers for Parallel Computing (LCPC 2005)*, pages 152–169, 2005.

[Tay83] R. N. Taylor. Complexity of analyzing the synchronization structure of concurrent programs. *ACTA INFORMATICA*, 19(1):57–84, 1983.

* This work is supported by Australian Research Council (ARC) Linkage Project LP0989643.