Aakash Saini(2018CSM1001)

March 18, 2019

- **Observations:** In C-family languages certain code fragments consist of *undefined behaviour* like pointer overflow, null pointer deference, signed integer overflow, division by zero, etc. For compiler optimization one way is to assume that program will never have undefined behaviour, but it is having many consequences like *unstable-code*. STACK use two-phase scheme for identifying the unstable code: first, running the optimizer O without taking advantage of undefined behaviour and in second, run optimizer taking advantage of undefined behaviour this time. Authors suggested certain assumptions for the well-defined program like a code fragment e is well-defined for input x iff undefined behaviour for e is never triggered after executing e. A program is well-defined iff each fragment of it is well defined for a particular input. Two oracle simplifications : first, Boolean oracle which proposes true and false enumerating all possible values and seconds is algebra oracle which eliminates common terms on both sides of comparison. STACK reduces false warning by ignoring compiler-generated code and special function calls(UB) are inserted into Information routine. Paper also suggested the Solver-based algorithms in which STACK take helps from Boolector solver for deciding the satisfiability for the elimination and simplification of queries.

- **Limitations:**It is practically difficult and infeasible to accurately compute the satisfiability using Solver-based algorithm for the large problems. In general, it is quite difficult to know that exactly how much unstable code will be missed by STACK.

- **Conclusions:** Paper proposes the study of unstable code which is often discarded by the compilers due to undefined behaviour which is important to consider. A new static-checker model, Stack is able to identify the unstable code from the program in an efficient manner which is helpful for the optimization strategies used by compiler writers.

- **Future work:** In future to solve the solver-based algorithms challenge, approximate queries can be computed by limiting the computation to a single function. More optimized mechanisms can be developed in future to compute unstable code fragments accurately.