Inferring sufficient numeric conditions with under-approximations by abstract interpretation

Internship proposal, Master 2 MPRI, year 2013–2014

Supervisor:	Antoine Miné (`mine@di.ens.fr`)
Internship location:	Département d'informatique, École normale supérieure, Paris, France
Relevant course:	M2–6: Abstract interpretation: application to verification and static analysis

Note: other internships are possible on the topic of numeric abstract domains, static analysis, and abstract interpretation in general. Please contact the internship supervisor for more information.

Motivation

The goal of the internship is to infer sufficient preconditions on the numeric variables of a program using an under-approximating backward abstract interpretation.

The theory of abstract interpretation allows the design of effective and efficient static analyzers able to compute approximations of program semantics. However, the overwhelming majority of analyzers compute over-approximations of the set of possible behaviors of a program: either forward (to compute an over-approximation of the states reachable from a given set of initial states), or backward (to refine the result of a forward analysis or to infer necessary conditions to reach a given set of final states). For instance, the numeric abstract domains (intervals, polyhedra, etc.) seen in the course are all over-approximating. Under-approximation in infinite-size abstract domains remains an under-explored area of abstract interpretation, with few articles (one being the preliminary work presented in [1]).

Propagating under-approximated semantic properties backward makes it possible to infer *sufficient conditions*, i.e., conditions on the initial states of the program so that all executions reach a given set of final states. There are numerous applications to sufficient conditions and we mention a few of them. Firstly, the inference of counter-examples, that is, of sequences of inputs that cause the program to always fail [3]. This should allow the automatic discovery of definitive bugs and exploits in programs. Secondly, the inference of procedure contracts [2], that is, of sufficient assertions to insert at the beginning of a procedure to ensure that its execution will never fail, with application to the automatic annotation and documentation of libraries. Thirdly, the design of smarter optimizing compilers. For instance, by inferring sufficient conditions such that all the array bound checks in a loop are correct, one can design a faster, check-free version of the loop and insert a dynamic test in the generated code that branches to this fast version when the condition holds, and reverts to the slow loop otherwise.

Expected work

The intern will focus primarily on numeric abstractions as these are well-understood and feature a rich collection of existing abstract domains (such as intervals, octagons, or polyhedra) with over-approximating operators. He will thus design under-approximating backward operators for these existing domains. One possible basis of work is [1]. Extensions include the design of new numeric abstract domains that may be better suited to represent under-approximated properties, the adaptation to under-approximations of generic abstract domain functors (such as partitioning, products, completions), and the design of domains for non-numeric properties.

Each proposed domain and operator shall be proved correct, and motivated by its usefulness analyzing interesting properties of actual program fragments and by its tractable complexity. While not required, implementing and evaluating the domains experimentally would be a plus.

References

[1] A. Miné. Backward under-approximations in numeric abstract domains to automatically infer sufficient program conditions. *Science of Computer Programming (SCP)*, 33 pages, Oct. 2013. `http://www.di.ens.fr/~mine/publi/article-mine-SCP13.pdf`.

[2] P. Cousot, R. Cousot, & F. Logozzo. Precondition Inference from Intermittent Assertions and Application to Contracts on Collections. In *12th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'11)*, Austin, Texas, LNCS 6538, Springer, 2011, pp. 150–168.

[3] F. Bourdoncle. Assertion-based debugging of imperative programs by abstract interpretation. In Proc. of 4th European Software Engineering Conf. (ESEC'93), pp. 501–516, vol. 717 of LNCS. Springer, 1993.