An Essay on Abstract Interpretation

Advanced Topics in Programming Languages

Abstract interpretation (AI) allows us to capture invariants about terms in a program without having to extensively test every single branch of execution or possible input. As we cannot have our cake and eat it too, when designing the abstract domain, we are forced to make a compromise: Do we sacrifice greater precision of our invariants, or do we pay a greater computational cost?

In *'The Octagon Abstract Domain'*, Antoine Miné strikes a balance between the two, describing the octagon abstract domain that represents invariants of the form $\pm x \pm y \leq c$, with x and y variables, and c a constant. Octagons are precise enough for most useful applications while keeping their computation cost reasonable. It is therefore natural that Jourdan, et al. express their desire to extend the static analyzer *Verasco* – introduced in *'A Formally-Verified C Static Analyzer'* – with the octagon abstract domain. Such an extension would greatly reduce Verasco's runtime, while proving to fit within the analyzer's interface with relative ease.

Octagons

Miné describes the relational abstract domain of octagons by extending Difference Bound Matrices (DBMs) which had previously been shown to represent constraints of the form $x - y \leq c$. For a program with N variables, a DBM is an $N \times N$ matrix such that each variable represents both a column and a row. So, for some DBM $\mathbf{m}$ representing a constraint $v_j - v_i \leq c$, we have that $\mathbf{m}_{i,j} = c$. Miné then notices that we can duplicate the set of variables such that we have a positive version v_i^+, and a negative v_i^- and now use these to represent the full set of constraints $\pm x \pm y \leq c$. For example, the constraint $x + y \leq c$ would be represented as $x^+ - y^- \leq c$ in our desired form for DBMs. Now, we have a $2N \times 2N$ DBM $\mathbf{m}^+$ with a row and column for each enhanced variable, representing a set of transformed $(\pm x \pm y \leq c)$ constraints. This encoding is our abstraction function α, and our concretization function γ is defined descriptively as the set of all variable mappings that satisfy the constraints.

Using the octagon abstract domain, we can also deduce further constraints from the initial set. Given constraints $(x^+ - x^- \leq c) \equiv (2x \leq c)$ and $(y^+ - y^- \leq d)$, we would like to deduce $\left(x^+ - y^- \leq \frac{c+d}{2}\right) \equiv \left(x + y \leq \frac{c+d}{2}\right)$. This can be done by computing the strong closure $(\mathbf{m}^+)^\bullet$of an octagon $\mathbf{m}^+$ which essentially finds the shortest path of constraints between any two variables. This strong closure transformation is not only useful for finding deducible constraints, but it is also necessary for defining the various abstract operators over octagons, such as union, widening and variable assignment. This transformation is also where the computational cost of octagons comes in, as the algorithm for computing the strong closure of a DBM has $\mathcal{O}(N^3)$ time cost. While this is by no means cheap, it is asymptotically much better than the more precise constraint domain of polyhedra, which is exponential in cost. Note that octagons also keep memory costs to $\mathcal{O}(N^2)$, i.e. proportional to the size of the underlying matrix. Thus, we see how octagons are able to express a broad set of constraints while keeping the cost of analysis at a reasonable complexity.

Verasco

The static analyzer Verasco is based on abstract interpretation and is able to prove the soundness of any analysis implemented using it. The architecture of the analyzer is modular and is implemented in three stages:

1. The abstract interpreter which performs the abstraction function over the program, providing an abstract state consisting of type, pointer, and numerical information. The concretization of

results provided by the lower two stages allow this stage to check for potential run-time errors such as out-of-bounds array accesses, and raises corresponding alarms.

2. The state abstract domain uses abstract operations to transform the state derived above. It uses points-to analysis to track any incongruities.
3. The last stage provides an interface to which we can attach a variety of numerical abstract domains. Jourdan, et al. implement multiple of these including convex polyhedra mentioned above, and numerical intervals.

It is into the third stage where we would attach the octagon abstract domain. As described in the paper, for analyzing constraints of numerical values, Verasco is limited to convex polyhedra and numerical intervals. These two abstract domains are at opposite ends of the precision/cost tradeoff. Therefore, to calculate the somewhat complex constraints necessary for out-of-bounds array checking or arithmetic exception, we need to rely on convex polyhedra, which as discussed earlier are very inefficient. In their experimental results, it took Verasco over 300 seconds to analyze a program a few hundred lines long that featured heavy array manipulation.

We can thus see the great benefit of including octagon into Verasco's pipeline as it could greatly speed up the cost of these static analyses. Moreover, while the memory usage is not reported on in the paper, we can assume that convex polyhedra will impose a ceiling on the length of programs allowable as we eventually hit the maximum memory allowable. Hence, the increased efficiency of octagons could further allow for the analysis of a greater range of longer programs.

Squaring the Octagon

Having identified the advantage that including octagons into Verasco's pipeline would bring, we are now faced with the challenge of consolidating the two. Jourdan, et al. do not give the full interface for the numerical abstract domains under which octagons would fall under, but they do give a description.

Firstly, a domain must specify a type `t` of abstract values. In our case, this would be the extended DBMs as described above.

Second, the domain must also specify a concretization γ to the union of $\mathbb{Z}$ integers and double-precision floats. Earlier we specified that our concretization function is the set of points that satisfies the constraints specified by the DBM. Miné gives more precisely defines this set as $\mathcal{D}^{+}(\mathbf{m}^{+})$, although does not call this the concretization function outright. Moreover, we note that this concretization function is not well defined over $\mathbb{Z}$, and Miné suggest that we analyze integer variables in $\mathbb{Q}$ or $\mathbb{R}$ at the cost of some added noise to our solutions.

Third, the domain requires functions to perform "forward" and "backward" analysis of the programming language's operators. In order to implement octagons in Verasco's pipeline we must also provide Coq proofs of soundedness for these functions over constraints of integer and floating-point variables. The latter can be approximated by the rational numbers $\mathbb{Q}$, which Miné explicitly mentions as compatible with his abstract domain. We are further shown in his paper how to implement assignment over arbitrary expressions through interval arithmetic. Fortunately, Verasco provides communication channels between abstract domains, thus simplifying the implementation of octagons as we can reuse the results from the parallel interval abstract domain.

Moreover, despite the C#minor language that Verasco targets having a more complicated control flow, (e.g. `exit`, `goto` and `return` statements), Verasco's architecture takes care of this in the second stage of analysis, simplifying these operators to joins and widenings both of which are specified for octagons. Nevertheless, the constraints represented by octagons are limited to only two variables without any coefficients. This would require the application of heuristics to implement more complex expressions, without the loss of too much precision.

We have therefore seen that adding octagons to Verasco's numerical analyses can be achieved by providing a direct implementation of Miné's system, modulo heuristics to deal with complex expressions. In fact, the modular nature of Verasco allows us to use results from other abstract domains such as numerical intervals to simplify the implementation. However, we must implement the proofs of soundness for operators on DBMs to ensure Verasco's soundedness.

Miné shows that the precision provided by octagons are sufficient for complex out-of-bounds array analysis, while keeping worst case cost at $\mathcal{O}(N^3)$ for each arithmetic operator. They believe that octagons can further be applied to various analyses such as pointer and string cleanness analysis. Verasco provides the infrastructure to do so, allowing many abstract domains to be applied to the same program and thus perform these such static analyses. However, as described in Jourdan, et al.'s paper, the use of the polyhedra abstract domain poses a significant slowdown on the analysis of even small-scale programs. By implementing octagons in Verasco, we should see an asymptotic speedup that will allow for larger programs to be analyzed by Verasco. Such an implementation should be straightforward once proofs of soundness for the operations on DBMs are devised.