Data Abstraction, Arrays, Maps, and Completeness, aka "Cell Morphing"

David Monniaux

Data Abstraction, Arrays, Maps, and Completeness, aka "Cell Morphing"

David Monniaux
Univ. Grenoble Alpes, CNRS, Grenoble INP*, VERIMAG, 38000 Grenoble, France
David.Monniaux@univ-grenoble-alpes.fr

(Invited talk abstract. This discusses joint work with Laure Gonnord and Julien Braine.)

Arrays and array-like data structures (such as hash-tables) are widely used in software. Furthermore, many semantic aspects of programming, such as byte-addressed memory, data structure fields, etc., can be modeled as arrays. Static analyzers that aim at proving properties of programs thus often need to deal with arrays. An array, in our context, is a map a from an index set I (not necessarily an integer range) to a value set V, with "get" $a[i]$ and "set" $a[i \leftarrow v]$ operations satisfying the relations:

$$a[i \leftarrow v][i] = v$$

$$a[i \leftarrow v][i'] = t[i'] \text{ when } i' \neq i$$

Many interesting invariants about arrays are universally quantified over indices. For instance, "the array a contains 42 at all indices from 0 included to n excluded" is written $\forall k,\ 0 \leq k < n \Rightarrow a[k] = 42$"; "the array a is sorted from indices from 0 included to n excluded" can be written $\forall k_1 k_2,\ 0 \leq k_1 \leq k_2 < n \Rightarrow a[k_1] \leq a[k_2]$. More generally, we consider invariants of the form $\forall k_1, \ldots, k_m P(k_1, \ldots, k_m, a[k_1], \ldots, a[k_m], v_1, \ldots, v_{|V|})$ where a is an array and the v_i are other variables of the program, and P is an arbitrary predicate. We also consider the case of multiple arrays, so as to be able to express properties such as $\forall k,\ 0 \leq k < n \Rightarrow a[k] = b[k]$.

Numerous approaches have been proposed to automatically infer such invariants. We have proposed an approach [BGM21, BGM16, Bra22, MG16], that converts an invariant inference problem (expressed as a solution to a system of Horn clauses), constrained by a safety property and restricted to such universally quantified invariants into an invariant inference problem on ordinary invariants (without universal quantification), through suitable *quantifier instantiation*. By further processing, through Ackermannization, these problems can even be converted, without loss of precision, to invariant inference problems over purely scalar variables (no more arrays).

While that second step does not incur loss of precision, the first step (quantifier instantiation), is in general sound (if the transformed invariant problem has a solution, then so has the original problem) but *incomplete*: it may be the case that it converts an invariant inference problem (a system of Horn clauses) that has a solution among universally quantified array invariants into a problem that has no solution. We however show that under certain syntactic restrictions (mostly, that the original Horn clause problem should be *linear*, which includes all problems obtained from the inductiveness conditions of control-flow graphs), that transformation is complete.

We thus show that under these conditions, our approach for solving invariant inference problems within universally quantified array properties is thus relatively complete to our ability to solve the resulting array-free ordinary invariant inference problem, in general over non-linear Horn clauses. This is the main limitation to our approach, since solvers for such kinds of problems tend to have unpredictable performance.

*Institute of Engineering Univ. Grenoble Alpes

Submitted to:
HCVS 2023

References

[BGM16] Julien Braine, Laure Gonnord & David Monniaux (2016): *Verifying Programs with Arrays and Lists*. Internship report, ENS Lyon. Available at `https://hal.archives-ouvertes.fr/hal-01337140`.

[BGM21] Julien Braine, Laure Gonnord & David Monniaux (2021): *Data Abstraction: A General Framework to Handle Program Verification of Data Structures*. In: *Static analysis (SAS)*, *Lecture Notes in Computer Science* 12913, Springer, pp. 215–235, doi:10.1007/978-3-030-88806-0_11. Available at `https://inria.hal.science/hal-03321868`.

[Bra22] Julien Braine (2022): *The Data-abstraction Framework: abstracting unbounded data-structures in Horn clauses, the case of arrays. (La Méthode Data-abstraction: une technique d'abstraction de structures de données non-bornées dans des clauses de Horn, le cas des tableaux)*. Ph.D. thesis, University of Lyon, France. Available at `https://tel.archives-ouvertes.fr/tel-03771839`.

[MG16] David Monniaux & Laure Gonnord (2016): *Cell Morphing: From Array Programs to Array-Free Horn Clauses*. In: *Static analysis*, *Lecture Notes in Computer Science* 9837, Springer Verlag, pp. 361–382, doi:10.1007/978-3-662-53413-7_18. arXiv:1509.09092.