Notes on Arrays

The decision procedure for the array property fragment (APF) of the theory of arrays T_{A} (with uninterpreted indices) presented in [2, 1] is unsound: it can report unsatisfiable when the given Σ_{A}-formula is T_{A}-satisfiable. The source of the unsoundness is the use of λ in Steps 4 and 6 ([1], pp. 295–296): it assumes that the domain of indices is arbitrarily large so that there is always some index value that is unequal to all values of terms in $\mathcal{I}$.

To correct the problem, we add a new axiom schema to T_{A} that restricts the domains of arrays to be infinite: for each positive natural number n, there is an axiom

$$\forall x_1, \ldots, x_n.\ \exists y.\ \bigwedge_{i=1}^{n} y \neq x_i \ .$$

The decision procedure is sound and complete with respect to the APF of this augmented theory.

Notice that $T_{\mathsf{A}}^{\mathbb{Z}}$ and its decision procedure for its APF is correct because integer-indexed arrays naturally have an infinite domain (the integers).

We illustrate the effects of this axiom schema through an example.

Consider the formula

$$F:\ (\forall i.\ a[i] = v)\ \wedge\ (\forall i.\ b[i] \neq v)\ \wedge\ (\forall i.\ i \neq j\ \rightarrow\ a[i] = b[i])\ .$$

If an array can have a finite domain, the formula is satisfiable: let the interpretation's domain have one element to which j is assigned. But the procedure for T_{A} introduces λ, resulting in the constraints

$$a[\lambda] = v\ \wedge\ b[\lambda] \neq v\ \wedge\ (\lambda \neq j\ \rightarrow\ a[\lambda] = b[\lambda])\ \wedge\ \lambda \neq j\ ,$$

in particular,

$$a[\lambda] = v\ \wedge\ b[\lambda] \neq v\ \wedge\ a[\lambda] = b[\lambda]\ ,$$

which is clearly unsatisfiable. Hence, the procedure reports unsatisfiable. In the context of the new axiom schema, this formula is indeed unsatisfiable: no interpretation with a domain of cardinality greater than one (in particular, no interpretation with an infinite domain) satisfies F. But without the axiom schema, F is satisfiable.

When reasoning about arrays in programs, using T_{A} (with the necessary axiom schema) does not make sense: arrays are typically bounded. Let us consider what one might write using $T_{\mathsf{A}}^{\mathbb{Z}}$ instead:

$$\begin{aligned} G:\ & (\forall i.\ 0 \leq i \leq n-1\ \rightarrow\ a[i] = v) \\ & \wedge\ (\forall i.\ 0 \leq i \leq n-1\ \rightarrow\ b[i] \neq v) \\ & \wedge\ (\forall i.\ 0 \leq i \leq n-1\ \wedge\ i \neq j\ \rightarrow\ a[i] = b[i])\ . \end{aligned}$$

Here, n is intended to be the lengths of a and b. The atom $i \neq j$ is written $i \leq j-1\ \vee\ i \geq j+1$ in the ARP of $T_{\mathsf{A}}^{\mathbb{Z}}$. Hence,

$$\mathcal{I} = \{0,\ n-1,\ j-1,\ j+1\}\ .$$

Instantiating the quantifiers over $\mathcal{I}$ reveals that G is satisfied by any $T_{\mathsf{A}}^{\mathbb{Z}}$-interpretation in which $n = 1$ and $j = 0$. The values of a and b outside of the interval $[0, 0]$ don't matter.

In other words, while the logical entities corresponding to a and b can be indexed by any integer, the interval that we actually care about is $[0, 0]$: a and b have "length" 1. In a C program, one only wants to access an array within its declared bounds; the other positions can be indexed (unfortunately), but they are not considered to be part of our understanding of the array, and doing so often results in a program crash.

What is the value of T_{A} then? T_{A} serves as a base theory on which to model sets, multisets, and hashtables. The axiom schema introduced above parallels the requirement that the QFF fragments of the theories of sets, multisets, and hashtables must be stably infinite to work within a Nelson-Oppen combination.

References

1. Aaron R. Bradley and Zohar Manna, The Calculus of Computation: Decision Procedures with Applications to Verification, Springer, 2007.
2. Aaron R. Bradley, Safety Analysis of Systems, Stanford PhD Thesis, 2007, (available at `http://ece.colorado.edu/~bradleya`).