Data abstraction, arrays, maps, and completeness, aka "Cell morphing"

Julien Braine Laure Gonnord David Monniaux

CNRS / VERIMAG

April 23, 2023

Arrays and maps

Array of elements of D: map from $\mathbb{Z}$ to D (implement bound check independently)
More generally: maps from I to D, for any I

Occurs:

- arrays
- array-like data structures (hash tables...)
- memory seen as an array of bytes
- structure field seen as an array indexed by the object
- array of processes, local variables in process indexed by process id
- participants in a protocol (indexed by participant id)

More arrays and maps

Set of elements from S: map from S to $\{0, 1\}$
Multiset of elements from S: map from S to $\mathbb{N}$
Relation between elements of A and B: map from $A \times B$ to $\{0, 1\}$

Can talk of contents of data structures (multiset of elements in an array, etc.)
Can talk of "who points where"

Invariants on single arrays

Often of the universal form

- $\forall k, P(i, j, \ldots, k, t[k])$ e.g. $\forall k, 0 \leq k < n \implies t[k] \geq 0$
- $\forall k_1 k_2, P(i, j, \ldots, k_1, k_2, t[k_1], t[k_2])$ e.g. $\forall k, 0 \leq k_1 \leq k_2 < n \implies t[k_1] \leq t[k_2]$

"At all positions (or pairs of positions) in the array, a certain relationship holds between the elements at these positions, these positions, and the rest of the program variables"

Other properties

Neighbors

$\forall k,\ P(k, t[k-1], t[k], t[k+1])$

Multidimensional arrays

$\forall i, j\ P(i, j, t[i, j])$

Index set = $\mathbb{Z}^2$

Invariants on multiple arrays

Such as

- $\forall k, P(i, j, \ldots, k, t_1[k], t_2[k])$ e.g. $\forall k, 0 \leq k < n \implies t_1[k] = t_2[k]$
- $\forall k, P(i, j, \ldots, k_1, k_2, t_1[k_1], t_2[k_2])$ e.g.
  $\forall k, 0 \leq k < n \wedge k2 = n - k1 \implies t_1[k_1] = t_2[k_2]$

Or "the multiset of elements in t_1 is the same as in t_2"

Horn clauses

```
int a[N];
for(int i=0; i<N; i++) {
  t[i] = 42;
}
```

Loop initialization $\forall N, a,\ L(N, a, 0)$

Loop exit $\forall N, a, i,\ L(N, a, i) \wedge i \geq N \implies E(N, a)$

Loop execution $\forall N, a, i,\ L(N, a, i) \wedge i < N \implies L(N, a[i \mapsto 42], i + 1)$

we may wish to prove $\forall Nai, E(N, a) \wedge 0 \leq i < N \implies a[i] = 42$

Une seule solution : la bonne abstraction !

abstract interpretation = look for solutions within an abstract domain
abstract domain = restrict the shape of invariants
e.g. “look for an invariant on tuples of integer variables in products of intervals”

Thus “look for invariants of the form $\forall k, P(i, j, \ldots, k, t_1[k], t_2[k])$” is abstract interpretation
abstract domain = $\{\{i, j, t_1, t_2 \mid \forall k, P(i, j, k, t_1[k], t_2[k])\} \mid P \subseteq \mathbb{Z} \times \mathbb{Z} \times D \times D\}$

The Galois connection

For $P^\sharp \subseteq \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times D$, $\gamma(P^\sharp) \subseteq: \mathbb{Z} \times \mathbb{Z} \times (\mathbb{Z} \to D)$

$$\gamma(P^\sharp)(N, i, a) \equiv \forall k,\ P^\sharp(N, i, k, a[k])$$

Of course $\alpha(P) = \bigwedge_{P^\sharp | P \subseteq \gamma(P^\sharp)} P^\sharp$

Our goal

Implement abstraction by syntactic transformation from Horn clause to Horn clauses.

Substituted Horn clauses

```
int a[N];
for(int i=0; i<N; i++) {
  t[i] = 42;
}
```

Loop initialization $\forall N, a\ (\forall k\ L^\sharp(N, 0, k, a[k]))$

Loop exit $\forall N, a, i\ \left(\forall k,\ L^\sharp(N, i, k, a[k])\right) \wedge i \geq N \implies \left(\forall k,\ E^\sharp(N, k, a[k])\right)$

Loop execution $\forall N, a, i,\ \left(\forall k,\ L^\sharp(N, i, k, a[k])\right) \wedge i < N \implies$
$\left(\forall k, L^\sharp(N, i+1, k, a[i \mapsto 42][k])\right)$

we may wish to prove $\forall N, a, i,\ \left(\forall k,\ E^\sharp(N, k, a[k])\right) \wedge 0 \leq i < N \implies a[i] = 42$

Quantifiers on the right hand side are easy

Loop initialization $\forall N, k, a_k \; L^\sharp(N, 0, k, a_k)$

Loop exit $\forall N, a, i, k \; \left(\forall k', \; L^\sharp(N, i, k', a[k'])\right) \wedge i \geq N \implies E^\sharp(N, k, a[k])$

Loop execution

$$\forall N, a, i, k \; \left(\forall k', \; L^\sharp(N, i, k', a[k'])\right) \wedge i < N \implies L^\sharp(N, i+1, k, a[i \mapsto 42][k])$$

Note: we still have quantifiers on the left.

Instantiating the quantifiers on the left hand side

Loop initialization $\forall N, k, a_k\ L^\sharp(N, 0, k, a_k)$

Loop exit $\forall N, a, i, k\ L^\sharp(N, i, k, a[k]) \wedge i \geq N \implies E^\sharp(N, k, a[k])$

Loop execution $\forall N, a, i, k\ L^\sharp(N, i, k, a[k]) \wedge i < N \implies L^\sharp(N, i+1, k, a[i \mapsto 42][k])$

Note: this is sound but could it be incomplete?

$P(k_1, x) \wedge P(k_2, x) \implies Q(x)$
implies
$(\forall k P(k, x)) \implies Q(x)$

Further processing

One can even get rid of arrays by “Ackermannization”

Loop initialization $\forall N, k, a_k\ L^\sharp(N, 0, k, a_k)$

Loop exit $\forall N, i, k, a_k\ L^\sharp(N, i, k, a_k) \wedge i \geq N \implies E^\sharp(N, k, a_k)$

Loop execution $\forall N, a, i, k, a_k\ L^\sharp(N, i, k, a_k) \wedge i < N \wedge k \neq i \implies L^\sharp(N, i+1, k, a_k)$

$\forall N, a, i, a_i\ L^\sharp(N, i, i, a_i) \wedge i < N \implies L^\sharp(N, i+1, i, 42)$

Note: now purely scalar problem, ready for Horn solvers!

Instantiation strategy

Collect all indices i such that $a[i]$ appears.
Use all of them for instantiating.

Is this complete? (= is the resulting system equivalent to the original?)

A simple example (I)

Two properties defining singletons:
$P(a)$ is true iff $\forall k,\ a[k] = 0$
$Q(a)$ is true iff $\forall k \neq 1,\ a[k] = 0 \wedge a[1] = 1$

In other words:
$P^\sharp(k, x) \equiv (x = 0)$
$Q^\sharp(k, x) \equiv (x = 0 \wedge k \neq 1) \vee (x = 1 \wedge k = 1)$
$P = \gamma(P^\sharp)$, $Q = \gamma(Q^\sharp)$, $P^\sharp = \alpha(P)$, $Q^\sharp = \alpha(Q)$.

A simple example (II)

Not abstracted

A Horn clause

$\forall a,\ P(a) \wedge Q(a) \implies R(a[0])$

(Obviously the left hand side is false, so the least solution for R is false. Thus some solutions can satisfy $\forall x,\ \neg R(x)$.)

Abstracted

$\forall a,\ \left(\forall k,\ P^\sharp(k, a[k])\right) \wedge \left(\forall k,\ Q^\sharp(k, a[k])\right) \implies R(a[0])$

(So far so good, the left hand side is false.)

A simple example (III)

Instantiated

$\forall a,\ P^\sharp(0, a[0]) \wedge Q^\sharp(0, a[0]) \implies R(a[0])$

Ackermannized

$\forall a_0,\ P^\sharp(0, a_0) \wedge Q^\sharp(0, a_0) \implies R(a_0)$

(The left hand side is obviously true for $a_0 = 0$. Thus no solution can satisfy $\forall x,\ \neg R(x)$.)

The instantiation scheme is incomplete!

Reason for incompleteness: holes

$$\forall a,\ \big(\forall k,\ P^\sharp(k, a[k])\big) \wedge \big(\forall k,\ Q^\sharp(k, a[k])\big) \implies R(a[0])$$

that is

$$\forall a,\ \big(\forall k,\ (P^\sharp \wedge Q^\sharp)(k, a[k])\big) \implies R(a[0])$$

$(P^\sharp \wedge Q^\sharp)(k, x) \equiv x = 0 \wedge k \neq 1$

"There is a hole at $k = 1$"

α not surjective / γ not injective

There are multiple $P^\sharp$ (and not only $\emptyset$) such that $\gamma(P^\sharp)$ is $\emptyset$.

A non minimal $P^\sharp$ can "propagate" and induce overapproximation.

Would need a reduction function ($\rho(P^\sharp) = \alpha \circ \gamma(P^\sharp)$) but how could I define it in Horn clauses...

Syntactic restrictions

Linear Horn clauses

Only one unknown predicate on the left hand side ($P(n, i, a) \wedge i \neq n \implies R(n, i, a)$ is ok, $P(n, i, a) \wedge Q(n, i, a) \wedge i \neq n \implies R(n, i, a)$ is not)

No global predicates on arrays

Only access arrays at individual locations $a[i]$, $a[j \mapsto 42][i]$ etc.
No "global" predicates ($a = a'$, etc)

Theorem: the abstraction is then complete!

Intuition of the proof

"Everything happens in the image by α."

"There are no holes in arrays."

"All partials arrays can be extended into full arrays."

Completeness in other words

Source = Horn clause problem with predicates on arrays
Question: solve this problem within an abstract domain ($\gamma(P^\sharp)$)

Replacement + instantiation + (optional) Ackermannization
= solve the question
(if syntactic restrictions not obeyed: may or may not solve the question even if there is a solution)

Implement an abstraction by syntactic transformation.

Experimental results

`https://github.com/vaphor`

Tends to generate “hard” problems for Horn clause solvers (Z3, Eldarica…)

Can prove the correctness of some classical sorting algorithms (multiset is invariant + result is sorted) by inferring the necessary invariants.

Completeness theorem

Past work

https://hal.science/hal-03214475v2
https://hal.science/tel-03771839v1
https://hal.science/hal-01337140v1
https://hal.science/hal-02948081v2
https://hal.science/hal-03321868v1

Work in progress

Clean up the completeness proof when some of the syntactic restrictions are removed.