AXDInterpolator

A Tool for Computing Interpolants for Arrays with MaxDiff

Jose Abel Castellanos Joo[1], Silvio Ghilardi[2], Alessandro Gianola[3], Deepak Kapur[1]

[1]Department of Computer Science
University of New Mexico, USA

[2]Dipartimento di Matematica
Università degli Studi di Milano, Italy

[3]Faculty of Computer Science
Free University of Bozen-Bolzano, Italy

19th International Workshop on Satisfiability Modulo Theories
July 19, 2021

THE UNIVERSITY OF NEW MEXICO

Motivation

- Theory of Arrays with MaxDiff $\langle =, \mathtt{rd}, \mathtt{wr}, \mathtt{diff} \rangle$, introduced in [1]

THE UNIVERSITY OF NEW MEXICO

Motivation

- Theory of Arrays with MaxDiff $\langle =, \mathtt{rd}, \mathtt{wr}, \mathtt{diff} \rangle$, introduced in [1]
  - $\forall x, y . x \neq y \rightarrow \mathtt{rd}(x, \mathtt{diff}(x, y)) \neq \mathtt{rd}(y, \mathtt{diff}(x, y))$
  - $\forall x, y, i . i > \mathtt{diff}(x, y) \rightarrow \mathtt{rd}(x, i) = \mathtt{rd}(y, i)$
  - $\forall x . \mathtt{diff}(x, x) = 0$

Motivation

- Theory of Arrays with MaxDiff $\langle =, \mathtt{rd}, \mathtt{wr}, \mathtt{diff} \rangle$, introduced in [1]
  - $\forall x, y. x \neq y \rightarrow \mathtt{rd}(x, \mathtt{diff}(x, y)) \neq \mathtt{rd}(y, \mathtt{diff}(x, y))$
  - $\forall x, y, i. i > \mathtt{diff}(x, y) \rightarrow \mathtt{rd}(x, i) = \mathtt{rd}(y, i)$
  - $\forall x. \mathtt{diff}(x, x) = 0$
- Such theory can formalize desirable specifications (in particular, the `length` function) without quantifiers of bounded arrays

THE UNIVERSITY OF NEW MEXICO

Contributions

- Implemented the proposed algorithm in [1] for the theory $\mathcal{ARD}(\mathcal{T}_{\mathcal{I}})$, where $\mathcal{T}_{\mathcal{I}}$ is an *index theory*.

THE UNIVERSITY OF NEW MEXICO

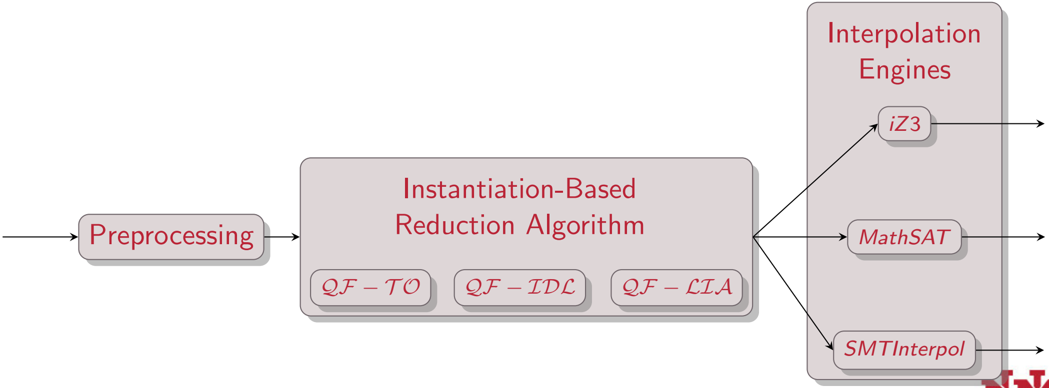
Contributions

- Implemented the proposed algorithm in [1] for the theory $\mathcal{ARD}(\mathcal{T}_{\mathcal{I}})$, where $\mathcal{T}_{\mathcal{I}}$ is an *index theory*.
- Provided support for the quantifier-free fragment of the index theories $\mathcal{TO}$, $\mathcal{IDL}$, and $\mathcal{LIA}$.

Contributions

- Implemented the proposed algorithm in [1] for the theory $\mathcal{ARD}(\mathcal{T_I})$, where $\mathcal{T_I}$ is an *index theory*.
- Provided support for the quantifier-free fragment of the index theories $\mathcal{TO}$, $\mathcal{IDL}$, and $\mathcal{LIA}$.
- Designed an architecture allowing the system to use different interpolation engines as black boxes. Currently, we support iZ3, SMTInterpol, and MathSAT.

Architecture Overview



Sequence of `diff` Applications

Suppose we are given two arrays a, b:

2	4	6	8	10	12	14	16

$\mathtt{diff}(a, b)$

- $a :=$

2	4	16	18	10	112	14	16

$\mathtt{diff}(a, b)$

- $b :=$

Then we let $b_2 = \mathtt{wr}(b, \mathtt{diff}(a, b), \mathtt{rd}(a, \mathtt{diff}(a, b)))$

Sequence of `diff` Applications

Suppose we are given two arrays a, b:

- $a :=$

2	4	6	8	10	12	14	16

$\mathtt{diff}(a, b_2)$

- $b_2 :=$

2	4	16	18	10	12	14	16

$\mathtt{diff}(a, b_2)$

Then we let $b_3 = \mathtt{wr}(b_2, \mathtt{diff}(a, b_2), \mathtt{rd}(a, \mathtt{diff}(a, b_2)))$

THE UNIVERSITY OF NEW MEXICO

Sequence of `diff` Applications

Suppose we are given two arrays a, b:

- $a :=$

2	4	6	8	10	12	14	16

$\texttt{diff}(a, b_3)$

- $b_3 :=$

2	4	16	8	10	12	14	16

$\texttt{diff}(a, b_3)$

and let $b_4 = \texttt{wr}(b_3, \texttt{diff}(a, b_3), \texttt{rd}(a, \texttt{diff}(a, b_3)))$

Sequence of `diff` Applications

Suppose we are given two arrays a, b:

- $a :=$

2	4	6	8	10	12	14	16

- $b_4 :=$

2	4	6	8	10	12	14	16

THE UNIVERSITY OF NEW MEXICO

Sequence of `diff` Applications

Suppose we are given two arrays a, b:

- $a :=$

2	4	6	8	10	12	14	16

- $b_4 :=$

2	4	6	8	10	12	14	16

$$b_1 := b; \qquad \texttt{diff}_1(a, b) := \texttt{diff}(a, b_1);$$

Sequence of `diff` Applications

Suppose we are given two arrays a, b:

- $a :=$

2	4	6	8	10	12	14	16

- $b_4 :=$

2	4	6	8	10	12	14	16

$$b_1 := b; \qquad \mathtt{diff}_1(a, b) := \mathtt{diff}(a, b_1);$$
$$b_{k+1} := \mathtt{wr}(b_k, \mathtt{diff}_k(a, b), \mathtt{rd}(a, \mathtt{diff}_k(a, b))); \qquad \mathtt{diff}_{k+1}(a, b) := \mathtt{diff}(a, b_{k+1})$$

THE UNIVERSITY OF NEW MEXICO

Sequence of `diff` Applications - Equivalence Lemma

The conjunctive formula

$$\mathtt{diff}_1(a, b) = k_1 \wedge \cdots\cdots \wedge \mathtt{diff}_l(a, b) = k_l \tag{1}$$

Sequence of `diff` Applications - Equivalence Lemma

The conjunctive formula

$$\mathtt{diff}_1(a,b) = k_1 \wedge \cdots\cdots \wedge \mathtt{diff}_l(a,b) = k_l \tag{1}$$

is equivalent modulo $\mathcal{ARD}$ to the conjunction of the following five formulæ:

$$k_1 \geq k_2 \wedge \cdots \wedge k_{l-1} \geq k_l \wedge k_l \geq 0 \tag{2}$$

$$\textstyle\bigwedge_{j<l}(k_j > k_{j+1} \to \mathtt{rd}(a,k_j) \neq \mathtt{rd}(b,k_j)) \tag{3}$$

$$\textstyle\bigwedge_{j<l}(k_j = k_{j+1} \to k_j = 0) \tag{4}$$

$$\textstyle\bigwedge_{j\leq l}(\mathtt{rd}(a,k_j) = \mathtt{rd}(b,k_j) \to k_j = 0) \tag{5}$$

$$\forall h\ (h > k_l \to \mathtt{rd}(a,h) = \mathtt{rd}(b,h) \vee h = k_1 \vee \cdots \vee h = k_{l-1}) \tag{6}$$

THE UNIVERSITY OF NEW MEXICO

Separated Pairs

A pair of formulas (ϕ_1, ϕ_2) is *separated* if

Separated Pairs

A pair of formulas (ϕ_1, ϕ_2) is *separated* if

- ϕ_1 contains sequences of `diff` applications and equations involving `wr` applications

Separated Pairs

A pair of formulas (ϕ_1, ϕ_2) is *separated* if

- ϕ_1 contains sequences of `diff` applications and equations involving `wr` applications
- ϕ_2 contains Boolean combinations of $\mathcal{T}_{\mathcal{I}}$-atoms and atoms of the form: $\{\mathtt{rd}(a, i) = \mathtt{rd}(b, j), \quad \mathtt{rd}(a, i) = e, \quad e_1 = e_2\}$, where a, b, i, j, e, e_1, e_2 are variables or constants of the form appropriate sorts.

M-Instantiations

- M-Instantiations allows us to obtain grounded formulæ from the first order formula involved in equivalence lemmas given an set of $\mathcal{T}_{\mathcal{I}}$ terms
- Additionally, this procedure ‘compiles‘ formulæ in $\mathcal{EUF} \cup \mathcal{T}_{\mathcal{I}}$ formulæ from $\mathcal{ARD}(\mathcal{T}_{\mathcal{I}})$
- The *complexity* $c(t)$ of a term t is defined as the number of function symbols occurring in t

THE UNIVERSITY OF NEW MEXICO

M-Instantiations - Pseudo Code for quantifier-free $\mathcal{IDL}$

Algorithm 1 M-Instantiation

```
procedure StandardInput::InstantiatedTerms::M-Instantiate
    for term ∈ terms do
        new-term ← (term + 1).simplify()
        if ¬ inSet(new-term, terms) then
            terms.push-back(new-term)
        end if
        new-term ← (term - 1).simplify()
        if ¬ inSet(new-term, terms) then
            terms.push-back(new-term)
        end if
    end for
end procedure
```

Interpolation Algorithm

Algorithm 2 Main Loop

```
procedure AXDInterpolator::MainLoop(StandardPair part-a, StandardPair part-b)
    if ¬(common-array-vars.areCommonPairsAvailable()) then
        SmtSolverSetup(solver, part-a)
        SmtSolverSetup(solver, part-b)
        if solver.check() = z3::unsat then
            is-unsat ← true
        end if
        return
    end if
    CircularPairIterator search-common-pairs(common-array-vars)
    while (num-attemps++ < remaining-fuel) do
        solver.push()
        SmtSolverSetup(solver, part-a)
        SmtSolverSetup(solver, part-b)
        if solver.check() = z3::unsat then
            is-unsat ← true
            return
        end if
        solver.pop()
        common-pair ← *search-common-pair
        part-a-dim ← part-a.diff-map.size-of-entry(common-pair)
        part-b-dim ← part-b.diff-map.size-of-entry(common-pair)
        dim ← min(part-a-dim, part-b-dim)
        new-index = fresh-index-constant()
        part-a.updateSaturation(common-pair, new-index, dim)
        part-b.updateSaturation(common-pair, new-index, dim)
        search-common-pair.next()
    end while
end procedure
```

Algorithm 3 SmtSolverSetup

```
procedure AXDInterpolator::SmtSolverSetup(z3::solver solver, StandardPair side-part)
    for assertion ∈ side-part.part-2 do
        solver.add(assertion)
    end for
    side-part.instantiate(solver, ∀x, i.i < 0 → rd(x, i) = ⊥)
    side-part.instantiate(solver, ∀i.rd(ε, i) = ⊥)
    for a = wr(b, i, e) ∈ side-part.write-vector do
        side-part.instantiate(solver, ∀h.h ≠ i → rd(a, h) = rd(b, h))
    end for
    for diff(a, b) = i ∈ side-part.diff-map do
        side-part.instantiate(solver, ∀h.h > i → rd(a, h) = rd(b, h))
    end for
end procedure
```

Algorithm 4 UpdateSaturation

```
procedure StandardPair::updateSaturation(z3Pair entry, z3::expr new-index, unsigned min-dim)
    a ← entry.first
    b ← entry.second
    map-element ← diff-map.find(entry)
    instantiated-terms.addVar(new-index)
    if Heuristic then
        instantiated-terms.M-instante()
    end if
    if min-dim < old-dim then
        part-2.push-back(new-index = (map-element.second)[min-dim]
    else
        prev-index ← (map-element.second)[old-min - 1]
        part-2.push-back(prev-index ≥ new-index)
        part-2.push-back(new-index ≥ 0)
        part-2.push-back(prev-index > new-index → rd(a, prev-index) ≠ rd(b, prev-index))
        part-2.push-back(prev-index = new-index → prev-index = 0)
    end if
    part-2.push-back(rd(a, new-index) = rd(b, new-index) → new-index = 0)
end procedure
```

THE UNIVERSITY OF NEW MEXICO

Benchmarks using SV-COMP and UAutomizer - Setup

- We tested our implementation using C-programs from the ReachSafety-Arrays and MemSafety-Arrays tracks of the SV-COMP [2]

THE UNIVERSITY OF NEW MEXICO

Benchmarks using SV-COMP and UAutomizer - Setup

- We tested our implementation using C-programs from the ReachSafety-Arrays and MemSafety-Arrays tracks of the SV-COMP [2]
- We used the model checker UAutomizer [3] to extract their SMT Scripts from the previous C-programs

THE UNIVERSITY OF NEW MEXICO

Benchmarks using SV-COMP and UAutomizer - Setup

- We tested our implementation using C-programs from the ReachSafety-Arrays and MemSafety-Arrays tracks of the SV-COMP [2]
- We used the model checker UAutomizer [3] to extract their SMT Scripts from the previous C-programs
- We let the machine produce SMT Scripts for 15 minutes. We used these SMT Scripts files to compare the number of interpolants computed from unsatisfiable formulas. For the latter we assigned each process up to 360 seconds and 6 GB of memory

THE UNIVERSITY OF NEW MEXICO

Benchmarks using SV-COMP and UAutomizer - Memsafety-track Results

Subtracks	AXD Interpolator					
	iZ3		*MathSAT*		*SMTInterpol*	
	Success	Timeout	Success	Timeout	Success	Timeout
array-examples	584	1	584	1	584	1
array-memsafety	118	0	118	0	118	0
termination-crafted	52	3	52	3	52	3

Table: Memsafety-track results - Our implementation

Subtracks	*iZ3*		*MathSAT*		*SMTInterpol*	
	Success	Timeout	Success	Timeout	Success	Timeout
array-examples	585	0	585	0	585	0
array-memsafety	118	0	118	0	118	0
termination-crafted	55	0	55	0	55	0

Table: Memsafety-track results - Other solvers

THE UNIVERSITY OF NEW MEXICO

Benchmarks using SV-COMP and UAutomizer - Reachsafety-track Results

Subtracks	AXD Interpolator *iZ3*		AXD Interpolator *MathSAT*		AXD Interpolator *SMTInterpol*	
	Success	Timeout	Success	Timeout	Success	Timeout
array-cav19	31	0	31	0	31	0
array-examples	50	0	50	0	50	0
array-fpi	774	21	774	21	774	21
array-industry-pattern	8	0	8	0	8	0
array-lopstr16	54	0	54	0	54	0
array-patterns	11	0	11	0	11	0
array-tiling	6	0	6	0	6	0
reducercommutativity	53	0	53	0	53	0

Table: Reachsafety-track results - Our implementation

Subtracks	*iZ3*		*MathSAT*		*SMTInterpol*	
	Success	Timeout	Success	Timeout	Success	Timeout
array-cav19	31	0	31	0	31	0
array-examples	50	0	50	0	50	0
array-fpi	795	0	795	0	795	0
array-industry-pattern	8	0	8	0	8	0
array-lopstr16	54	0	54	0	54	0
array-patterns	11	0	11	0	11	0
array-tiling	6	0	6	0	6	0
reducercommutativity	53	0	53	0	53	0

Table: Reachsafety-track results - Other Solvers

THE UNIVERSITY OF NEW MEXICO

Future Work

- We handle boolean combination of formulas using a DNF transformation. Such transformation appears to be the first target to rework since this can take exponential amount of time.

Future Work

- We handle boolean combination of formulas using a DNF transformation. Such transformation appears to be the first target to rework since this can take exponential amount of time.
- The current design does not perform incremental satisfiability checks. Incremental checks are possible to implement due to the incremental nature of the proposed interpolation algorithm by including a hash consed data structure on the terms/predicates produced in the main loop of the algorithm and because the data structure `z3::solver` can keep track of previously proven assertions.

THE UNIVERSITY OF NEW MEXICO

Conclusions

- In this paper we described AXDInterpolator, the implementation of the interpolation algorithm presented in [1].

Conclusions

- In this paper we described AXDInterpolator, the implementation of the interpolation algorithm presented in [1].
- We were able to show the feasibility of AXDInterpolator by validating it on two benchmarks taken from the SV-COMP.

Conclusions

- In this paper we described AXDInterpolator, the implementation of the interpolation algorithm presented in [1].
- We were able to show the feasibility of AXDInterpolator by validating it on two benchmarks taken from the SV-COMP.
- We also compared our implementation with state-of-the-art solvers: apart from very few timeout outcomes, our tool managed to handle all the examples the other solvers did.

THE UNIVERSITY OF NEW MEXICO

Conclusions

- In this paper we described AXDInterpolator, the implementation of the interpolation algorithm presented in [1].
- We were able to show the feasibility of AXDInterpolator by validating it on two benchmarks taken from the SV-COMP.
- We also compared our implementation with state-of-the-art solvers: apart from very few timeout outcomes, our tool managed to handle all the examples the other solvers did.
- We also found interesting examples that are not handled by other state-of-the-art solvers, which makes the option of our language extension and tool an appealing consideration.

THE UNIVERSITY OF NEW MEXICO

Thanks for your attention!

References I

Silvio Ghilardi, Alessandro Gianola, and Deepak Kapur.
Interpolation and amalgamation for arrays with maxdiff.
In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2021.

Jan Friso Groote, Kim Guldstrand Larsen, and Dirk Beyer.
Software verification: 10th comparative evaluation (sv-comp 2021).
Tools and Algorithms for the Construction and Analysis of Systems27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City,

References II

Luxembourg, March 27 – April 1, 2021, Proceedings, Part II, 12652:401—422, February 2021.

Matthias Heizmann, Jürgen Christ, Daniel Dietsch, Evren Ermis, Jochen Hoenicke, Markus Lindenmann, Alexander Nutz, Christian Schilling, and Andreas Podelski.
Ultimate Automizer with SMTInterpol.
In Nir Piterman and Scott A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 641–643, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.