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
August 9, 2022

THE UNIVERSITY OF NEW MEXICO

Motivation

- The theory of arrays with extensionality does not have quantifier-free interpolation [5]

Motivation

- The theory of arrays with extensionality does not have quantifier-free interpolation [5]

Example: $(h' = \mathtt{wr}(h, i, e), a \neq b \wedge \mathtt{rd}(h, a) \neq \mathtt{rd}(h', a) \wedge \mathtt{rd}(h, b) \neq \mathtt{rd}(h', b))$

- Interpolant in $\langle =, \mathtt{wr}, \mathtt{rd} \rangle$ is $\exists j.(\mathtt{rd}(h, j) \neq \mathtt{rd}(h', j) \wedge \forall k.(k \neq j \rightarrow \mathtt{rd}(h, k) = \mathtt{rd}(h', k)))$

Motivation

- The theory of arrays with extensionality does not have quantifier-free interpolation [5]
- If the signature of the theory is extended with an additional `diff` operator, the resulting theory becomes quantifier-free interpolating [1]

Example: $(h' = \mathtt{wr}(h, i, e), a \neq b \wedge \mathtt{rd}(h, a) \neq \mathtt{rd}(h', a) \wedge \mathtt{rd}(h, b) \neq \mathtt{rd}(h', b))$

- Interpolant in $\langle =, \mathtt{wr}, \mathtt{rd} \rangle$ is $\exists j.(\mathtt{rd}(h, j) \neq \mathtt{rd}(h', j) \wedge \forall k.(k \neq j \rightarrow \mathtt{rd}(h, k) = \mathtt{rd}(h', k)))$

Motivation

- The theory of arrays with extensionality does not have quantifier-free interpolation [5]
- If the signature of the theory is extended with an additional `diff` operator, the resulting theory becomes quantifier-free interpolating [1]

Example: $(h' = \mathtt{wr}(h, i, e), a \neq b \wedge \mathtt{rd}(h, a) \neq \mathtt{rd}(h', a) \wedge \mathtt{rd}(h, b) \neq \mathtt{rd}(h', b))$

- Interpolant in $\langle =, \mathtt{wr}, \mathtt{rd} \rangle$ is $\exists j.(\mathtt{rd}(h, j) \neq \mathtt{rd}(h', j) \wedge \forall k.(k \neq j \rightarrow \mathtt{rd}(h, k) = \mathtt{rd}(h', k)))$
- Interpolant in $\langle =, \mathtt{wr}, \mathtt{rd}, \mathtt{diff} \rangle$ is $\mathtt{let}\ j = \mathtt{diff}(h, h')$ be in $h' = \mathtt{wr}(h, j, \mathtt{rd}(h, j))$

Motivation (Cont'd)

- In the literature, the `diff` operator is a binary function skolemizing the extensionality axiom, which lacks a meaningful interpretation

Motivation (Cont'd)

- In the literature, the `diff` operator is a binary function skolemizing the extensionality axiom, which lacks a meaningful interpretation
- The `diff` operator in this work returns 0 if the two input arrays are the same and otherwise returns the biggest index where the input arrays are different

Motivation (Cont'd)

- In the literature, the `diff` operator is a binary function skolemizing the extensionality axiom, which lacks a meaningful interpretation
- The `diff` operator in this work returns 0 if the two input arrays are the same and otherwise returns the biggest index where the input arrays are different
- Endowing this semantic on `diff` allows the theory to formalize desirable specifications (in particular, the `length` function) without quantifiers of bounded arrays

THE UNIVERSITY OF NEW MEXICO

Contributions

- Implemented the proposed algorithm in [2] for the theory $\mathcal{ARD}(\mathcal{T}_{\mathcal{I}})$, where $\mathcal{T}_{\mathcal{I}}$ is an *index theory*.

Contributions

- Implemented the proposed algorithm in [2] 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 $\mathcal{LIA}$.

THE UNIVERSITY OF NEW MEXICO

Contributions

- Implemented the proposed algorithm in [2] 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 $\mathcal{LIA}$.
- Designed an architecture allowing the system to use different interpolation engines as black boxes. Currently, we support iZ3, SMTInterpol, and MathSAT.

THE UNIVERSITY OF NEW MEXICO

Theory Arrays with MaxDiff

- $\mathcal{ARD}(\mathcal{T_I})$ includes, besides the axioms of $\mathcal{T_I}$, the following axioms:

Theory Arrays with MaxDiff

- $\mathcal{ARD}(\mathcal{T}_{\mathcal{I}})$ includes, besides the axioms of $\mathcal{T}_{\mathcal{I}}$, the following axioms:

$$\begin{aligned}
\forall y, i, e. &\quad i \geq 0 \to \mathtt{rd}(\mathtt{wr}(y, i, e), i) = e && (1)\\
\forall y, i, j, e. &\quad i \neq j \to \mathtt{rd}(\mathtt{wr}(y, i, e), j) = \mathtt{rd}(y, j) && (2)\\
\forall x, y. &\quad x \neq y \to \mathtt{rd}(x, \mathtt{diff}(x, y)) \neq \mathtt{rd}(y, \mathtt{diff}(x, y)) && (3)\\
\forall x, y, i. &\quad i > \mathtt{diff}(x, y) \to \mathtt{rd}(x, i) = \mathtt{rd}(y, i) && (4)\\
\forall x. &\quad \mathtt{diff}(x, x) = 0 && (5)\\
\forall x.i &\quad i < 0 \to \mathtt{rd}(x, i) = \bot && (6)\\
\forall i. &\quad \mathtt{rd}(\varepsilon, i) = \bot && (7)
\end{aligned}$$

Theory Arrays with MaxDiff

- $\mathcal{ARD}(\mathcal{T}_\mathcal{I})$ includes, besides the axioms of $\mathcal{T}_\mathcal{I}$, the following axioms:

$$\begin{aligned}
\forall y, i, e. &\quad i \geq 0 \rightarrow \mathtt{rd}(\mathtt{wr}(y, i, e), i) = e && (1)\\
\forall y, i, j, e. &\quad i \neq j \rightarrow \mathtt{rd}(\mathtt{wr}(y, i, e), j) = \mathtt{rd}(y, j) && (2)\\
\forall x, y. &\quad x \neq y \rightarrow \mathtt{rd}(x, \mathtt{diff}(x, y)) \neq \mathtt{rd}(y, \mathtt{diff}(x, y)) && (3)\\
\forall x, y, i. &\quad i > \mathtt{diff}(x, y) \rightarrow \mathtt{rd}(x, i) = \mathtt{rd}(y, i) && (4)\\
\forall x. &\quad \mathtt{diff}(x, x) = 0 && (5)\\
\forall x. i &\quad i < 0 \rightarrow \mathtt{rd}(x, i) = \bot && (6)\\
\forall i. &\quad \mathtt{rd}(\varepsilon, i) = \bot && (7)
\end{aligned}$$

- As an effect of the above axioms, we have that an array x is undefined outside the interval $[0, |x|]$, where $|x|$ is defined as $|x| := \mathtt{diff}(x, \varepsilon)$.

THE UNIVERSITY OF NEW MEXICO

Theory Arrays with MaxDiff (Cont'd)

Given array variables a, b, we define by mutual recursion the sequence of array terms $b_1, b_2, \ldots$ and of index terms $\mathtt{diff}_1(a, b), \mathtt{diff}_2(a, b), \ldots$:

THE UNIVERSITY OF NEW MEXICO

Theory Arrays with MaxDiff (Cont'd)

Given array variables a, b, we define by mutual recursion the sequence of array terms $b_1, b_2, \ldots$ and of index terms $\texttt{diff}_1(a, b), \texttt{diff}_2(a, b), \ldots$:

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

Theory Arrays with MaxDiff (Cont'd)

Given array variables a, b, we define by mutual recursion the sequence of array terms $b_1, b_2, \ldots$ and of index terms $\mathtt{diff}_1(a,b), \mathtt{diff}_2(a,b), \ldots$:

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

Theory Arrays with MaxDiff - Lemma

The conjunctive formula

$$\texttt{diff}_1(a, b) = k_1 \wedge \cdots\cdots \wedge \texttt{diff}_l(a, b) = k_l \tag{8}$$

Theory Arrays with MaxDiff - Lemma

The conjunctive formula

$$\mathtt{diff}_1(a, b) = k_1 \wedge \cdots\cdots \wedge \mathtt{diff}_l(a, b) = k_l \tag{8}$$

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

$$k_1 \geq k_2 \wedge \cdots \wedge k_{l-1} \geq k_l \wedge k_l \geq 0 \tag{9}$$

$$\textstyle\bigwedge_{j<l}(k_j > k_{j+1} \to \mathtt{rd}(a, k_j) \neq \mathtt{rd}(b, k_j)) \tag{10}$$

$$\textstyle\bigwedge_{j<l}(k_j = k_{j+1} \to k_j = 0) \tag{11}$$

$$\textstyle\bigwedge_{j\leq l}(\mathtt{rd}(a, k_j) = \mathtt{rd}(b, k_j) \to k_j = 0) \tag{12}$$

$$\forall h\ (h > k_l \to \mathtt{rd}(a, h) = \mathtt{rd}(b, h) \vee h = k_1 \vee \cdots \vee h = k_{l-1}) \tag{13}$$

THE UNIVERSITY OF NEW MEXICO

Separated Pairs

A pair of sets of quantifier-free formulae $\Phi = (\Phi_1, \Phi_2)$ is a *separated pair* iff

Separated Pairs

A pair of sets of quantifier-free formulae $\Phi = (\Phi_1, \Phi_2)$ is a *separated pair* iff

(1) Φ_1 contains equalities of the form $\texttt{diff}_k(a, b) = i$ and $a = \texttt{wr}(b, i, e)$; moreover if it contains the equality $\texttt{diff}_k(a, b) = i$, it must also contain an equality of the form $\texttt{diff}_l(a, b) = j$ for every $l < k$;

Separated Pairs

A pair of sets of quantifier-free formulae $\Phi = (\Phi_1, \Phi_2)$ is a *separated pair* iff

(1) Φ_1 contains equalities of the form $\mathtt{diff}_k(a, b) = i$ and $a = \mathtt{wr}(b, i, e)$; moreover if it contains the equality $\mathtt{diff}_k(a, b) = i$, it must also contain an equality of the form $\mathtt{diff}_l(a, b) = j$ for every $l < k$;

(2) Φ_2 contains Boolean combinations of T_I-atoms and of atoms of the forms: $\{\mathtt{rd}(a, i) = \mathtt{rd}(b, j), \ \ \mathtt{rd}(a, i) = e, \ \ e_1 = e_2\}$, where a, b, i, j, e, e_1, e_2 are variables or constants of the appropriate sorts.

M-Instantiations

If $\mathcal{I}$ is a set of $\mathcal{T}_{\mathcal{I}}$-terms, an $\mathcal{I}$-*instance* of a universal formula of the kind $\forall i\, \phi$ is a formula of the kind $\phi(t/i)$ for some $t \in \mathcal{I}$.

M-Instantiations

If $\mathcal{I}$ is a set of $\mathcal{T}_{\mathcal{I}}$-terms, an $\mathcal{I}$-*instance* of a universal formula of the kind $\forall i\, \phi$ is a formula of the kind $\phi(t/i)$ for some $t \in \mathcal{I}$.

Let $\mathcal{I}$ be a set of $\mathcal{T}_{\mathcal{I}}$-terms and let $\Phi = (\Phi_1, \Phi_2)$ be a separated pair; we let $\Phi(\mathcal{I}) = (\Phi_1(\mathcal{I}), \Phi_2(\mathcal{I}))$ be the smallest separated pair satisfying the following conditions:

THE UNIVERSITY OF NEW MEXICO

M-Instantiations

If $\mathcal{I}$ is a set of $\mathcal{T}_{\mathcal{I}}$-terms, an $\mathcal{I}$-*instance* of a universal formula of the kind $\forall i\, \phi$ is a formula of the kind $\phi(t/i)$ for some $t \in \mathcal{I}$.

Let $\mathcal{I}$ be a set of $\mathcal{T}_{\mathcal{I}}$-terms and let $\Phi = (\Phi_1, \Phi_2)$ be a separated pair; we let $\Phi(\mathcal{I}) = (\Phi_1(\mathcal{I}), \Phi_2(\mathcal{I}))$ be the smallest separated pair satisfying the following conditions:

- $\Phi_1(\mathcal{I})$ is equal to Φ_1 and $\Phi_2(\mathcal{I})$ contains Φ_2;

M-Instantiations

If $\mathcal{I}$ is a set of $\mathcal{T}_{\mathcal{I}}$-terms, an $\mathcal{I}$-*instance* of a universal formula of the kind $\forall i\, \phi$ is a formula of the kind $\phi(t/i)$ for some $t \in \mathcal{I}$.

Let $\mathcal{I}$ be a set of $\mathcal{T}_{\mathcal{I}}$-terms and let $\Phi = (\Phi_1, \Phi_2)$ be a separated pair; we let $\Phi(\mathcal{I}) = (\Phi_1(\mathcal{I}), \Phi_2(\mathcal{I}))$ be the smallest separated pair satisfying the following conditions:

- $\Phi_1(\mathcal{I})$ is equal to Φ_1 and $\Phi_2(\mathcal{I})$ contains Φ_2;
- $\Phi_2(\mathcal{I})$ contains all $\mathcal{I}$-instances of the two formulæ $\forall i\, \mathtt{rd}(\varepsilon, i) = \bot$, $\forall i\, (i < 0 \rightarrow \mathtt{rd}(a, i) = \bot)$, where a is any array variable occurring in Φ_1 or Φ_2;

M-Instantiations

If $\mathcal{I}$ is a set of $\mathcal{T}_{\mathcal{I}}$-terms, an $\mathcal{I}$-*instance* of a universal formula of the kind $\forall i\, \phi$ is a formula of the kind $\phi(t/i)$ for some $t \in \mathcal{I}$.

Let $\mathcal{I}$ be a set of $\mathcal{T}_{\mathcal{I}}$-terms and let $\Phi = (\Phi_1, \Phi_2)$ be a separated pair; we let $\Phi(\mathcal{I}) = (\Phi_1(\mathcal{I}), \Phi_2(\mathcal{I}))$ be the smallest separated pair satisfying the following conditions:

- $\Phi_1(\mathcal{I})$ is equal to Φ_1 and $\Phi_2(\mathcal{I})$ contains Φ_2;
- $\Phi_2(\mathcal{I})$ contains all $\mathcal{I}$-instances of the two formulæ $\forall i\, \mathtt{rd}(\varepsilon, i) = \bot$, $\forall i\, (i < 0 \to \mathtt{rd}(a, i) = \bot)$, where a is any array variable occurring in Φ_1 or Φ_2;
- if Φ_1 contains $a = \mathtt{wr}(b, i, e)$, then $\Phi_2(\mathcal{I})$ contains *all the* $\mathcal{I}$*-instances of the equivalent formula* $(i \geq 0 \to \mathtt{rd}(a, i) = e) \wedge \forall h\, (h \neq i \to \mathtt{rd}(a, h) = \mathtt{rd}(b, h))$;

M-Instantiations

If $\mathcal{I}$ is a set of $\mathcal{T}_\mathcal{I}$-terms, an *$\mathcal{I}$-instance* of a universal formula of the kind $\forall i\, \phi$ is a formula of the kind $\phi(t/i)$ for some $t \in \mathcal{I}$.

Let $\mathcal{I}$ be a set of $\mathcal{T}_\mathcal{I}$-terms and let $\Phi = (\Phi_1, \Phi_2)$ be a separated pair; we let $\Phi(\mathcal{I}) = (\Phi_1(\mathcal{I}), \Phi_2(\mathcal{I}))$ be the smallest separated pair satisfying the following conditions:

- $\Phi_1(\mathcal{I})$ is equal to Φ_1 and $\Phi_2(\mathcal{I})$ contains Φ_2;
- $\Phi_2(\mathcal{I})$ contains all $\mathcal{I}$-instances of the two formulæ $\forall i\, \mathtt{rd}(\varepsilon, i) = \bot$, $\forall i\, (i < 0 \to \mathtt{rd}(a, i) = \bot)$, where a is any array variable occurring in Φ_1 or Φ_2;
- if Φ_1 contains $a = \mathtt{wr}(b, i, e)$, then $\Phi_2(\mathcal{I})$ contains *all the $\mathcal{I}$-instances of the equivalent formula* $(i \geq 0 \to \mathtt{rd}(a, i) = e) \wedge \forall h\, (h \neq i \to \mathtt{rd}(a, h) = \mathtt{rd}(b, h))$;
- if Φ_1 contains the conjunction $\bigwedge_{i=1}^{l} \mathtt{diff}_i(a, b) = k_i$, then $\Phi_2(\mathcal{I})$ contains the formulae (9), (10), (11), (12) as well as *all $\mathcal{I}$-instances of the formula* (13).

M-Instantiations (Cont'd)

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 (Cont'd)

The *complexity* $c(t)$ of a term t is defined as the number of function symbols occurring in t.
For $M \in \mathbb{N} \cup \{\infty\}$, the *M-instantiation* of $\Phi = (\Phi_1, \Phi_2)$ is the separated pair $\Phi(\mathcal{I}_\Phi^M) = (\Phi_1(\mathcal{I}_\Phi^M), \Phi_2(\mathcal{I}_\Phi^M))$, where $\mathcal{I}_\Phi^M$ is the set of T_I-terms of complexity at most M built up from the index variables occurring in Φ_1, Φ_2.

M-Instantiations (Cont'd)

The *complexity* $c(t)$ of a term t is defined as the number of function symbols occurring in t.
For $M \in \mathbb{N} \cup \{\infty\}$, the *M-instantiation* of $\Phi = (\Phi_1, \Phi_2)$ is the separated pair $\Phi(\mathcal{I}_\Phi^M) = (\Phi_1(\mathcal{I}_\Phi^M), \Phi_2(\mathcal{I}_\Phi^M))$, where $\mathcal{I}_\Phi^M$ is the set of T_I-terms of complexity at most M built up from the index variables occurring in Φ_1, Φ_2.
The *full instantiation* of $\Phi = (\Phi_1, \Phi_2)$ is the separated pair $\Phi(\mathcal{I}_\Phi^\infty) = (\Phi_1(\mathcal{I}_\Phi^\infty), \Phi_2(\mathcal{I}_\Phi^\infty))$ (which is usually not finite). A separated pair $\Phi = (\Phi_1, \Phi_2)$ is *M-instantiated* iff $\Phi = \Phi(\mathcal{I}_\Phi^M)$

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
```

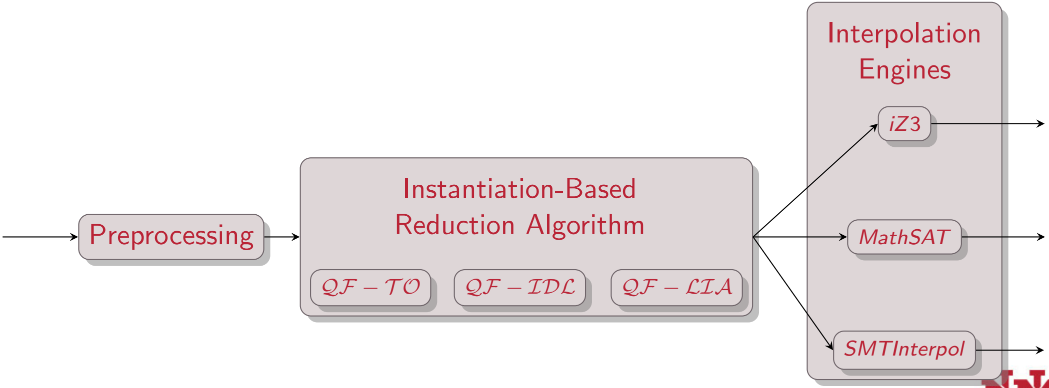
Algorithm 3 SmtSolverSetup

```
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
```

Architecture Overview



Input Files and Extended Signature

Our extended language is parameterized by the array sorts in the input formula as well as by an *index theory*. The domain sort of every array is currently implemented using the `Int` sort.

```
(declare-sort A)
(declare-fun diff'A' ((Array Int A) (Array Int A)) Int)
(declare-fun length'A' ((Array Int A)) Int)
(declare-fun empty_array'A' () (Array Int A))
(declare-fun undefined'A' () A)
```

Figure: Our extended language parameterized with a sort A.

THE UNIVERSITY OF NEW MEXICO

Demo

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 [3]

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 [3]
- We used the model checker UAutomizer [4] 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 [3]
- We used the model checker UAutomizer [4] 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*		AXD Interpolator *MathSAT*		AXD Interpolator *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.

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.
- 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.

Conclusions

- In this paper we described AXDInterpolator, the implementation of the interpolation algorithm presented in [2].

Conclusions

- In this paper we described AXDInterpolator, the implementation of the interpolation algorithm presented in [2].
- 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 [2].
- 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 [2].
- 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

Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise.
Quantifier-free interpolation of a theory of arrays.
Log. Methods Comput. Sci., 8(2), 2012.

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.

THE UNIVERSITY OF NEW MEXICO

References II

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, 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.

THE UNIVERSITY OF NEW MEXICO

References III

Deepak Kapur, Rupak Majumdar, and Calogero G. Zarba.
Interpolation for Data Structures.
In *Proc. of SIGSOFT-FSE*, pages 105–116. ACM, 2006.