Decision Procedures for Flat Array Properties

F. Alberti[1,3], S. Ghilardi[2], N. Sharygina[1]

[1]University of Lugano, Switzerland
[2] University of Milan, Italy
[3] Verimag, Grenoble, France

SMT

July 17, 2014

Talk based on the paper published at TACAS, 2014.

Context: quantified fragments of array theories

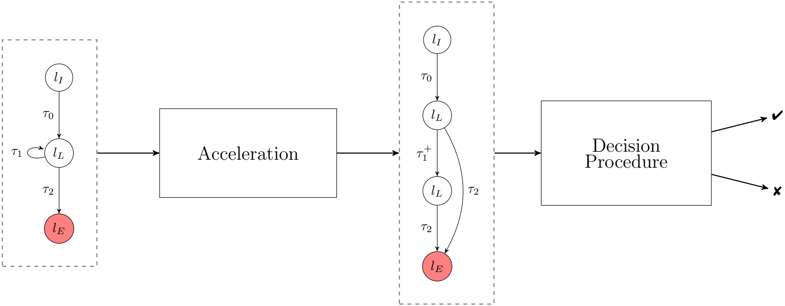
Many applications:

- Properties of the heap
- Checking user provided assertions
- Parameterized systems

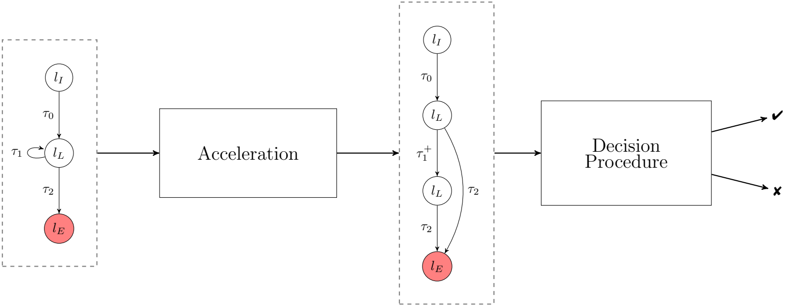
$\Rightarrow$ Verifying array programs:

- CEGAR-based approaches for array programs [AlbertiBG$^+$12]
- **Accelerations of relations over arrays** [AlbertiGS13]

Accelerations of relations over arrays



Accelerations of relations over arrays



- ✔ Accelerations of a class of relation over arrays is definable via $\exists^*\forall^*$-formulæ [AlbertiGS13]
- Accelerations might be outside known decidable fragments [BradleyMS06, HabermehlIV08, GedM09].

Accelerations of relations over arrays

$$\tau := G(i, \mathbf{a}[i]) \quad \wedge \quad i' = i + \bar{k} \quad \wedge \quad \mathbf{a}' = \mathsf{store}(\mathbf{a}, i, \mathbf{t}(\mathbf{a}[i]))$$

$$\Downarrow$$

$$\tau^{+} := \exists y > 0. \begin{pmatrix} \forall j.\ [\ i \leq j < i + \bar{k} \cdot y \wedge D_{\bar{k}}(j - i) \quad \rightarrow \quad G(\ j, \mathbf{a}(j)\ )\ ]\ \wedge \\ i' = i + \bar{k} \cdot y \wedge \\ \forall j.\ [\ \mathbf{a}'(j) = \mathbf{U}(\ i, j, y, \mathbf{a}(j)\ )\ ] \end{pmatrix}$$

Theory of arrays: “base” theory T + free functions $\mathbf{a}$

Fragment of interest: $\varphi := \exists \mathbf{c} \, \forall \mathbf{i} \; \psi(\; \mathbf{c} \; , \; \mathbf{i} \; , \; \mathbf{a}(t) \;)$

Quantified fragments of array theories

Related work

Theory of arrays: "base" theory T + free functions $\mathbf{a}$

Fragment of interest: $\varphi := \exists \mathbf{c} \, \forall \mathbf{i} \; \psi(\; \mathbf{c} \; , \; \mathbf{i} \; , \; \mathbf{a}(t) \;)$

- In general, undecidable

- If constrained, two main strategies to show decidability:

  1. Instantiation-based

  2. Automata-based

Quantified fragments of array theories

Related work

Bradley et al. “What’s decidable about arrays?”, VMCAI 2006.

- Array property: $\varphi \ := \ \forall \mathbf{i}.F(\mathbf{i}) \to G(\ \mathbf{a}(\mathbf{i})\ )$
  - $F(\mathbf{i})$ is a conjunction of atoms of the kind $i \leq j$, $i \leq t$, $t \leq i$

I. Identify an *index set* $\mathcal{I}$
II. Instantiate $\mathbf{i}$ over $\mathcal{I}$ to obtain a quantifier-free $\psi_1 \land \cdots \land \psi_n$
III. Standard theory-combination approaches on $\psi_1 \land \cdots \land \psi_n$

- Complexity: NExpTime (NP if we fix the number of index variables)

Habermehl et al. “A Logic of Singly Indexed Arrays”, LPAR 2008.

- $\varphi \ := \ \forall \mathbf{i}.F(\mathbf{i}) \to G(\mathbf{i}, \mathbf{a}(\mathbf{i} + \bar{\mathbf{k}}))$
  - No disjunctions in G
  - Atoms are difference logic constraints (with equations modulo $\bar{\mathbf{k}}$)

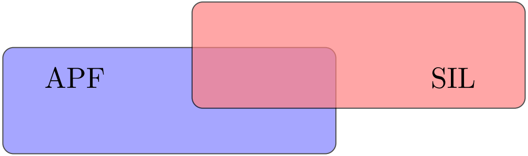
I. Translate φ into a FCADBM[1] $\mathcal{A}_\varphi$
II. Check the emptiness of $\mathcal{L}(\mathcal{A}_\varphi)$

- Complexity: unknown

[1] Deterministic flat counter automata with difference bound transition rules

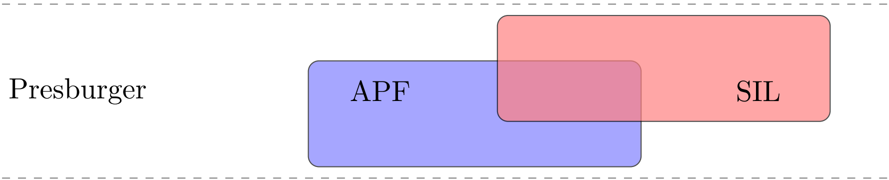
Quantified fragments of array theories

Our contribution wrt related work



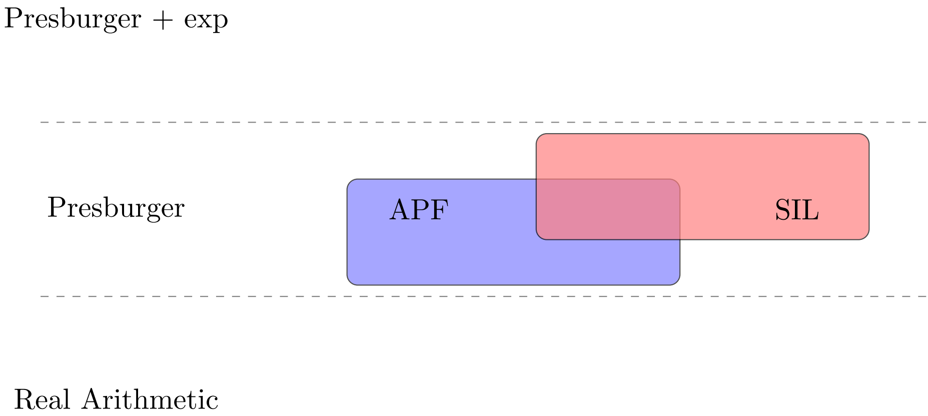
Quantified fragments of array theories

Our contribution wrt related work



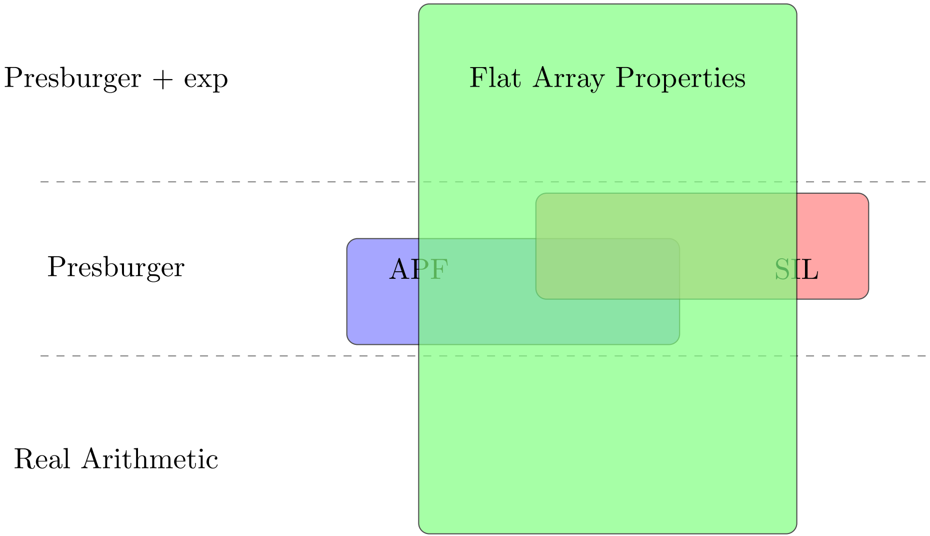
Quantified fragments of array theories

Our contribution wrt related work



Quantified fragments of array theories

Our contribution wrt related work



Our contribution

Flat Array Properties

- $\varphi \ := \ \exists \mathbf{c} \, \forall \mathbf{i}. \psi(\ \mathbf{i} \ , \ \mathbf{a}(\mathbf{i}) \ , \ \mathbf{c} \ , \ \mathbf{a}(\mathbf{c}) \ )$
  - $a(t)$ allowed only if t is a variable

Our contribution
Flat Array Properties

- $\varphi \ := \ \exists \mathbf{c} \, \forall \mathbf{i}. \psi(\ \mathbf{i} \ , \ \mathbf{a}(\mathbf{i}) \ , \ \mathbf{c} \ , \ \mathbf{a}(\mathbf{c}) \ )$
  - $a(t)$ allowed only if t is a variable

- Mono-sorted theory: $T \cup \{a_1, \ldots, a_n\}$
  - $|\mathbf{i}| = 1$
  - Requirement: T-decidability of $\exists^*\forall\exists^*$-formulæ
  - Complexity: quadratic instance of a $\exists^*\forall\exists^*$ T-satisfiability problem

Our contribution

- $\varphi \ := \ \exists \mathbf{c} \, \forall \mathbf{i}.\psi(\ \mathbf{i} \ , \ \mathbf{a}(\mathbf{i}) \ , \ \mathbf{c} \ , \ \mathbf{a}(\mathbf{c}) \ )$
  - $a(t)$ allowed only if t is a variable

- Mono-sorted theory: $T \cup \{a_1, \ldots, a_n\}$
  - $|\mathbf{i}| = 1$
  - Requirement: T-decidability of $\exists^*\forall\exists^*$-formulæ
  - Complexity: quadratic instance of a $\exists^*\forall\exists^*$ T-satisfiability problem

- Multi-sorted theory: $T_I \cup T_E \cup \{a_1, \ldots, a_n\}$
  - `INDEX` atoms with at most one universally quantified variable
  - Requirement: T_I-decidability of $\exists^*\forall$-formulæ
  - Requirement: T_E-decidability of quantifier-free formulæ
  - Complexity if T_I, T_E are $\mathbb{P}^+$: NExpTime-complete

Decision Procedure for the multi-sorted case

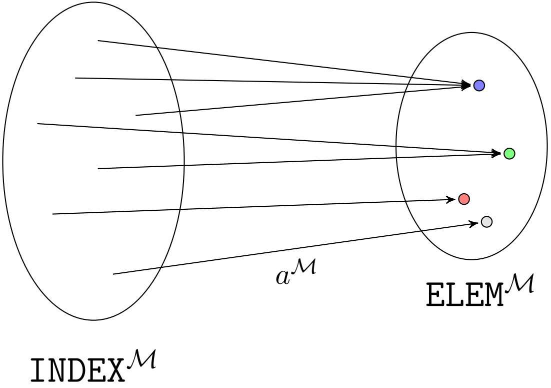
$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

$$\mathcal{M} \models F$$

Decision Procedure for the multi-sorted case

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

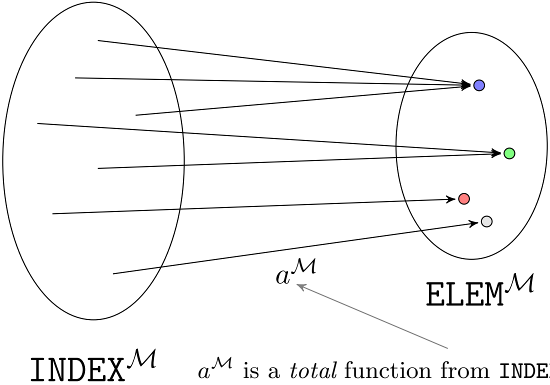
$$\mathcal{M} \models F$$



Decision Procedure for the multi-sorted case

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

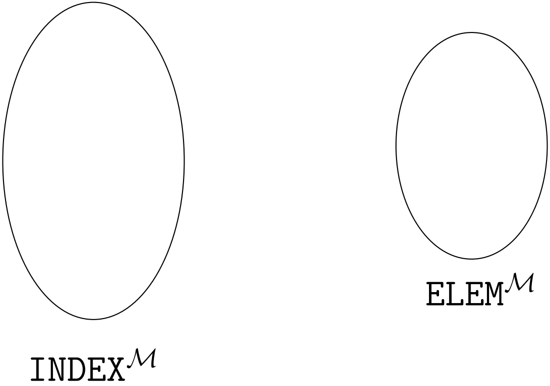
$$\mathcal{M} \models F$$



$a^{\mathcal{M}}$ is a *total* function from $\mathtt{INDEX}^{\mathcal{M}}$ to $\mathtt{ELEM}^{\mathcal{M}}$

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

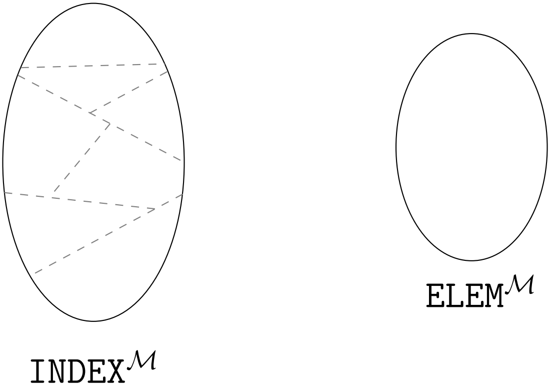
STEP I. Guess the set of `INDEX` *types*



Decision Procedure for the multi-sorted case

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

$\boxed{\text{S}\textsc{tep}\ \text{I. Guess the set of } \texttt{INDEX}\ \textit{types}}$



Decision Procedure for the multi-sorted case

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

$\boxed{\text{S}\textsc{tep}\text{ I. Guess the set of } \texttt{INDEX}\ \textit{types}}$

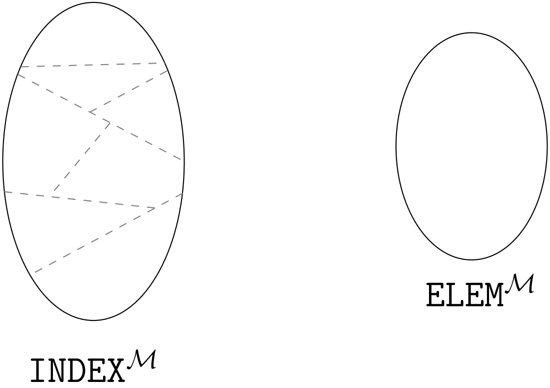
- Consider the set K of all `INDEX` *atoms* in F (plus equalities with the $\mathbf{c}$ constants)
- Let $\{M_1, \ldots, M_q\}$ be the the set of *maximal* and *consistent* sets of literals built out of K
  - Each $L(x, \mathbf{c})$ in every M_h is an atom of K or its negation
  - All the M_h's are mutually exclusive
- Every element of $\texttt{INDEX}^{\mathcal{M}}$ has to realize a type M_h:

$$\mathcal{M}_I \models \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right)$$

Decision Procedure for the multi-sorted case

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

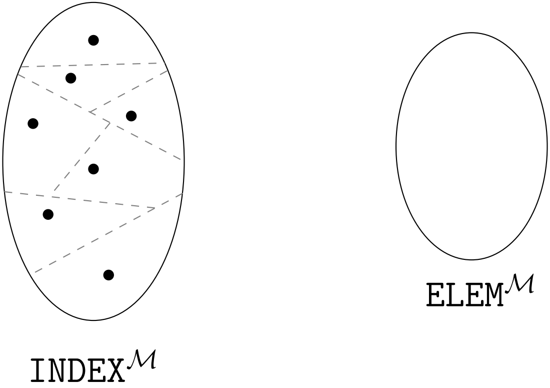
$\boxed{\text{S}\textsc{tep}\ \text{II. For each } \textit{type}\ M_h \text{ take a } b_h \in \texttt{INDEX}^{\mathcal{M}} \text{ realizing it}}$



Decision Procedure for the multi-sorted case

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

$\boxed{\text{STEP II. For each } \textit{type } M_h \text{ take a } b_h \in \texttt{INDEX}^{\mathcal{M}} \text{ realizing it}}$



Decision Procedure for the multi-sorted case

$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

$\boxed{\textsc{Step II. For each } \textit{type } M_h \text{ take a } b_h \in \texttt{INDEX}^{\mathcal{M}} \text{ realizing it}}$

1. Each b_h realizes the corresponding type

$$\mathcal{M}_I \models \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c})$$

2. The instantiation

$$\bigwedge_{\sigma : \mathbf{i} \to \mathbf{b}} \psi(\ \mathbf{i}\sigma, a(\mathbf{i}\sigma), \mathbf{c}, a(\mathbf{c})\ )$$

is consistent

Decision Procedure for $\mathtt{ARR}^2(T_I, T_E)$

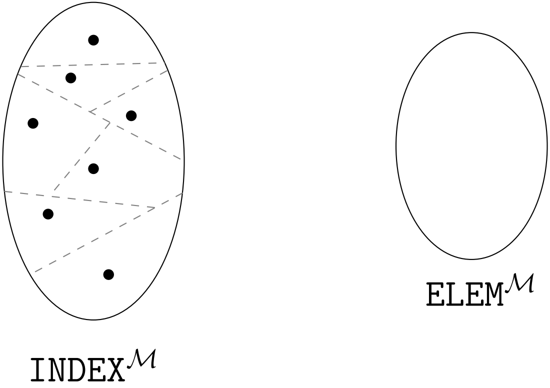
$$F := \exists \mathbf{c}\ \forall \mathbf{i}\ .\psi(\ \mathbf{i}, a(\mathbf{i}), \mathbf{c}, a(\mathbf{c})\ )$$

$$\rightsquigarrow$$

$$F_1 := \exists \mathbf{b}\, \exists \mathbf{c}\ \left[\begin{array}{l} \forall x.\left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right)\ \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c})\ \wedge \\ \bigwedge_{\sigma : \mathbf{i} \to \mathbf{b}} \psi(\mathbf{i}\sigma, a(\mathbf{i}\sigma), \mathbf{c}, a(\mathbf{c})) \end{array} \right]$$

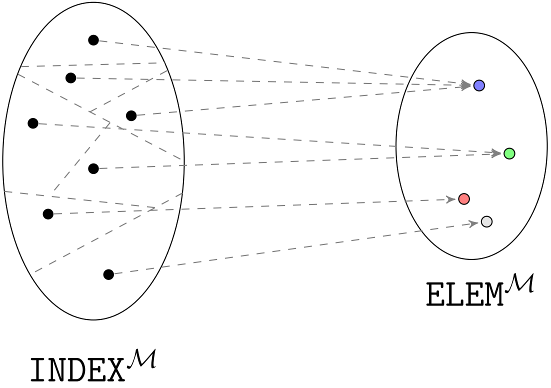
Decision Procedure for the multi-sorted case

STEP III. Substitute the tuple $a(\mathbf{b}) * a(\mathbf{c})$ with a tuple $\mathbf{e}$ of `ELEM` constants



Decision Procedure for the multi-sorted case

STEP III. Substitute the tuple $a(\mathbf{b}) * a(\mathbf{c})$ with a tuple $\mathbf{e}$ of `ELEM` constants



$$F_1 := \exists \mathbf{b}\, \exists \mathbf{c} \left[\begin{array}{l} \ldots \;\wedge \\ \displaystyle\bigwedge_{\sigma:\mathbf{i}\to\mathbf{b}} \psi(\mathbf{i}\sigma, a(\mathbf{i}\sigma), \mathbf{c}, a(\mathbf{c})) \end{array} \right]$$

STEP III. Substitute the tuple $a(\mathbf{b}) * a(\mathbf{c})$ with a tuple $\mathbf{e}$ of `ELEM` constants

Decision Procedure for the multi-sorted case

$$F_1 := \exists \mathbf{b}\, \exists \mathbf{c} \left[\begin{array}{l} \ldots\ \wedge \\ \displaystyle\bigwedge_{\sigma : \mathbf{i} \to \mathbf{b}} \psi(\mathbf{i}\sigma, a(\mathbf{i}\sigma), \mathbf{c}, a(\mathbf{c})) \end{array} \right]$$

STEP III. Substitute the tuple $a(\mathbf{b}) * a(\mathbf{c})$ with a tuple $\mathbf{e}$ of `ELEM` constants

$a(\mathbf{b}) * a(\mathbf{c}) \ \rightsquigarrow\ \mathbf{e}$

$$F_2 := \exists \mathbf{b}\, \exists \mathbf{c} \left[\begin{array}{l} \ldots\ \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}, \mathbf{e}) \ \wedge \\ \displaystyle\bigwedge_{d_m, d_n \in \mathbf{b} * \mathbf{c}} \ \bigwedge_{l=1}^{s} (d_m = d_n \to e_{l,m} = e_{l,n}) \end{array} \right]$$

functional consistency

Decision Procedure for the multi-sorted case

STEP IV. "Split" the formula F_2 in `INDEX` and `ELEM` parts

$$F_2 := \exists \mathbf{b}\, \exists \mathbf{c} \left[\begin{array}{l} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}, \mathbf{e}) \wedge \bigwedge_{d_m, d_n \in \mathbf{b}*\mathbf{c}} \bigwedge_{l=1}^{s} (d_m = d_n \to e_{l,m} = e_{l,n}) \end{array} \right]$$

Decision Procedure for the multi-sorted case

Step IV. "Split" the formula F_2 in `INDEX` and `ELEM` parts

$$F_2 := \exists \mathbf{b}\, \exists \mathbf{c} \left[\begin{array}{l} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \ \wedge \\ \bigwedge_{j=1}^{q} \quad \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \ \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}, \mathbf{e}) \ \wedge \quad \bigwedge_{d_m, d_n \in \mathbf{b} * \mathbf{c}} \ \bigwedge_{l=1}^{s} (d_m = d_n \to e_{l,m} = e_{l,n}) \end{array} \right]$$

$$F_I := \exists \mathbf{b}\, \exists \mathbf{c} \left[\begin{array}{l} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \ \wedge \\ \bigwedge_{j=1}^{q} \quad \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \ \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}) \end{array} \right]$$

$$F_E := \bar{\psi}(\mathbf{e})$$

STEP IV. "Split" the formula F_2 in `INDEX` and `ELEM` parts

$$F_2 := \exists \mathbf{b}\, \exists \mathbf{c} \begin{bmatrix} \boxed{\begin{array}{l} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \wedge \end{array}} \\ \bar{\psi}(\mathbf{b}, \mathbf{c}, \mathbf{e}) \wedge \bigwedge_{d_m, d_n \in \mathbf{b} * \mathbf{c}} \bigwedge_{l=1}^{s} (d_m = d_n \rightarrow e_{l,m} = e_{l,n}) \end{bmatrix}$$

$$F_I := \exists \mathbf{b}\, \exists \mathbf{c} \begin{bmatrix} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}) \end{bmatrix}$$

$$F_E := \bar{\psi}(\mathbf{e})$$

STEP IV. "Split" the formula F_2 in `INDEX` and `ELEM` parts

$$F_2 := \exists \mathbf{b}\, \exists \mathbf{c} \begin{bmatrix} \boxed{\begin{array}{l} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \ \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \ \wedge \end{array}} \\ \boxed{\bar{\psi}(\mathbf{b}, \mathbf{c}, \mathbf{e}) \ \wedge \bigwedge_{d_m, d_n \in \mathbf{b} * \mathbf{c}} \bigwedge_{l=1}^{s} (d_m = d_n \to e_{l,m} = e_{l,n})} \end{bmatrix}$$

SAT

assignment

$$F_I := \exists \mathbf{b}\, \exists \mathbf{c} \begin{bmatrix} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \ \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \ \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}) \end{bmatrix}$$

$$F_E := \bar{\psi}(\mathbf{e})$$

Decision Procedure for the multi-sorted case

$\boxed{\textsc{Step V. Check if } F_I \text{ is } T_I\text{-sat and if } F_E \text{ is } T_E\text{-sat}}$

[1*] With divisibility predicates $\{D_k\}_{k \geq 2}$.

Decision Procedure for the multi-sorted case

$\boxed{\textsc{Step V. Check if } F_I \text{ is } T_I\text{-sat and if } F_E \text{ is } T_E\text{-sat}}$

$$F_I := \exists \mathbf{b}\, \exists \mathbf{c} \begin{bmatrix} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}) \end{bmatrix}$$

$$F_E := \bar{\psi}(\mathbf{e})$$

[1*] With divisibility predicates $\{D_k\}_{k \geq 2}$.

Decision Procedure for the multi-sorted case

$\boxed{\text{STEP V. Check if } F_I \text{ is } T_I\text{-sat and if } F_E \text{ is } T_E\text{-sat}}$

$$F_I := \exists \mathbf{b}\, \exists \mathbf{c} \begin{bmatrix} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}) \end{bmatrix}$$

$$F_E := \bar{\psi}(\mathbf{e})$$

$\Rightarrow$ $\exists^*\forall$-fragment

$\Rightarrow$ Quantifier-free fragment

[1*] With divisibility predicates $\{D_k\}_{k \geq 2}$.

Decision Procedure for the multi-sorted case

$\boxed{\textsc{Step V. Check if } F_I \text{ is } T_I\text{-sat and if } F_E \text{ is } T_E\text{-sat}}$

$$F_I := \exists \mathbf{b}\, \exists \mathbf{c} \left[\begin{array}{l} \forall x. \left(\bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, \mathbf{c}) \right) \wedge \\ \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, \mathbf{c}) \wedge \\ \bar{\psi}(\mathbf{b}, \mathbf{c}) \end{array} \right]$$

$$F_E := \bar{\psi}(\mathbf{e})$$

$\Rightarrow$ $\exists^*\forall$-fragment

- ✔ Difference Logic*
- ✔ Presburger*
- ✔ Presburger* + exp [Semënov84]
- ✔ Real Arithmetic

...

$\Rightarrow$ Quantifier-free fragment

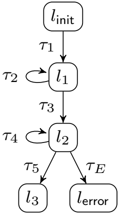
[1]* With divisibility predicates $\{D_k\}_{k \geq 2}$.

Application (I) - Deciding the safety of $\mathsf{simple}^0_{\mathcal{A}}$-programs

Application: deciding the safety of $\mathsf{simple}^0_{\mathcal{A}}$-*programs*

Application: deciding the safety of $\mathsf{simple}^0_{\mathcal{A}}$-*programs*

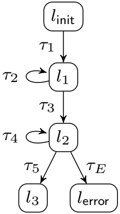
- Flat control-flow structure
- Every loop τ has a Flat Array Property as acceleration



Application (I) - Deciding the safety of $\mathsf{simple}^0_{\mathcal{A}}$-programs

Application: deciding the safety of $\mathsf{simple}^0_{\mathcal{A}}$-*programs*

- Flat control-flow structure
- Every loop τ has a Flat Array Property as acceleration

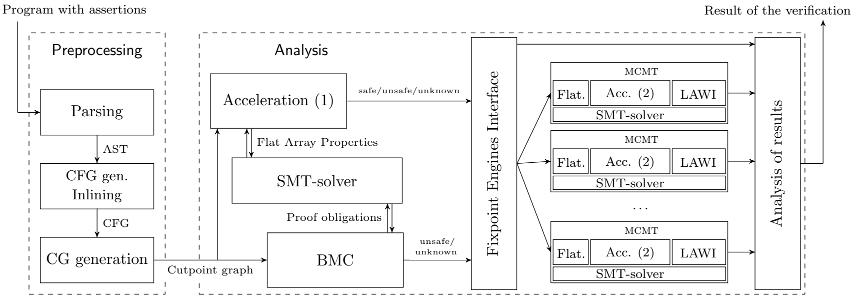


Theorem

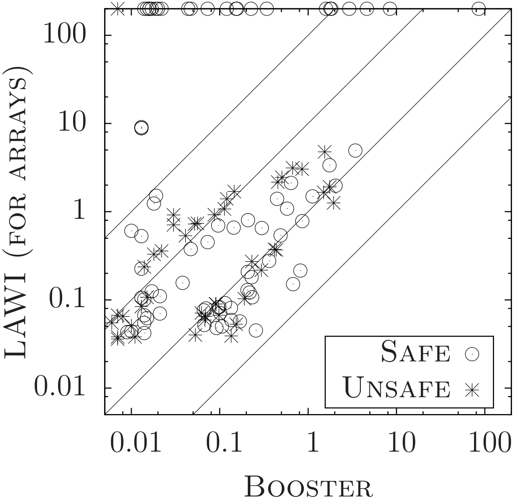
The unbounded reachability problem for $\mathsf{simple}^0_{\mathcal{A}}$*-programs is decidable.*

Application (II) - Booster

An acceleration-based software model-checker



F. Alberti, S. Ghilardi, and N. Sharygina.
Booster: an acceleration-based verification framework for array programs
In *ATVA*, Springer, 2014. To appear.



100
10
1
0.1
0.01
LAWI (for arrays)
0.01
0.1
1
10
100
Booster
Safe
Unsafe

Conclusion

1. New decidability results for quantified fragments of theories of arrays
   - Fully declarative
   - Parametric in the theories of indexes and elements

Conclusion

1. New decidability results for quantified fragments of theories of arrays
   - Fully declarative
   - Parametric in the theories of indexes and elements

2. Full decidability result for checking the safety of a class of array programs

Conclusion

1. New decidability results for quantified fragments of theories of arrays
   - Fully declarative
   - Parametric in the theories of indexes and elements

2. Full decidability result for checking the safety of a class of array programs

3. Application in software model-checking
   - ☞ BOOSTER – `inf.usi.ch/phd/alberti/prj/booster`

Conclusion

1. New decidability results for quantified fragments of theories of arrays
   - Fully declarative
   - Parametric in the theories of indexes and elements

2. Full decidability result for checking the safety of a class of array programs

3. Application in software model-checking
   - ☞ Booster – `inf.usi.ch/phd/alberti/prj/booster`

Thank you! Questions?

References I

Francesco Alberti, Roberto Bruttomesso, Silvio Ghilardi, Silvio Ranise, and Natasha Sharygina.
Lazy abstraction with interpolants for arrays.
In Nikolaj Bjørner and Andrei Voronkov, editors, *LPAR*, volume 7180 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2012.

Francesco Alberti, Silvio Ghilardi, and Natasha Sharygina.
Definability of accelerated relations in a theory of arrays and its applications.
In *FroCos*, pages 23–39, 2013.

References II

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma.
What's decidable about arrays?
In E. Allen Emerson and Kedar S. Namjoshi, editors, *VMCAI*, volume 3855 of *Lecture Notes in Computer Science*, pages 427–442. Springer, 2006.

Yeting Ge and Leonardo M. de Moura.
Complete instantiation for quantified formulas in satisfiability modulo theories.
In Ahmed Bouajjani and Oded Maler, editors, *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.

References III

Peter Habermehl, Radu Iosif, and Tomás Vojnar.
A logic of singly indexed arrays.
In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *LPAR*, volume 5330 of *Lecture Notes in Computer Science*, pages 558–573. Springer, 2008.

A.L. Semënov.
Logical theories of one-place functions on the set of natural numbers.
Izvestiya: Mathematics, 22:587–618, 1984.