Decision Procedures for Flat Array Properties

F. Alberti\textsuperscript{1,3}, S. Ghilardi\textsuperscript{2}, N. Sharygina\textsuperscript{1}

\textsuperscript{1}University of Lugano, Switzerland
\textsuperscript{2} University of Milan, Italy
\textsuperscript{3} Verimag, Grenoble, France

SMT
July 17, 2014

Talk based on the paper published at TACAS, 2014.
Many applications:

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

⇒ Verifying array programs:
  - CEGAR-based approaches for array programs [AlbertiBG⁺12]
  - Accelerations of relations over arrays [AlbertiGS13]
Accelerations of relations over arrays is definable via \( \exists \ast \forall \ast \)-formulæ \cite{AlbertiGS13}.

Accelerations might be outside known decidable fragments \cite{BradleyMS06, HabermehlIV08, GedM09}.

F. Alberti

Decision Procedures for Flat Array Properties

2 / 21
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

\[ \tau := G(i, a[i]) \land i' = i + \bar{k} \land a' = \text{store}(a, i, t(a[i])) \]

\[ \Downarrow \]

\[ \tau^+ := \exists y > 0. \left( \forall j. \left[ i \leq j < i + \bar{k} \cdot y \land D\bar{k}(j - i) \rightarrow G(j, a(j)) \right] \land i' = i + \bar{k} \cdot y \land \forall j. \left[ a'(j) = U(i, j, y, a(j)) \right] \right) \]
Theory of arrays: “base” theory $T +$ free functions $a$

Fragment of interest: $\varphi := \exists c \forall i \psi( c, i, a(t) )$
Quantified fragments of array theories
Related work

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

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

- In general, undecidable

- If constrained, two main strategies to show decidability:
  
  1. Instantiation-based
  2. Automata-based

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

I. Identify an index set \( I \)

II. Instantiate \( i \) over \( I \) 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: \( \text{NExpTime} \) (NP if we fix the number of index variables)
Quantified fragments of array theories
Related work


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

I. Translate $\varphi$ into a FCADBM$^1$ $A_\varphi$

II. Check the emptiness of $L(A_\varphi)$

- Complexity: unknown

---

$^1$Deterministic flat counter automata with difference bound transition rules
Quantified fragments of array theories

Our contribution wrt related work

APF

SIL
Quantified fragments of array theories

Our contribution wrt related work

Presburger

APF

SIL
Quantified fragments of array theories
Our contribution wrt related work

Presburger + exp

Presburger
APF
SIL

Real Arithmetic
Quantified fragments of array theories

Our contribution wrt related work

Presburger + exp

Flat Array Properties

Presburger

APF

Real Arithmetic

SIL
Our contribution
Flat Array Properties

\[ \varphi := \exists c \forall i. \psi( i, a(i), c, a(c) ) \]

- \( a(t) \) allowed only if \( t \) is a variable
Our contribution
Flat Array Properties

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

- Mono-sorted theory: \( T \cup \{ a_1, \ldots, a_n \} \)
  - \( |i| = 1 \)
  - Requirement: \( T \)-decidability of \( \exists^* \forall^* \)-formulæ
  - Complexity: quadratic instance of a \( \exists^* \forall^* T \)-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 \( P^+ \): \( NExpTime \)-complete
Our contribution
Flat Array Properties

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

- Mono-sorted theory: $T \cup \{ a_1, \ldots, a_n \}$
  - $|i| = 1$
  - Requirement: $T$-decidability of $\exists^* \forall^* \exists^*$-formulæ
  - Complexity: quadratic instance of a $\exists^* \forall^* T$-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 $\mathbb{P}^+$: $\text{NEXPTime}$-complete
\[ F := \exists c \, \forall i \, \psi( i, a(i), c, a(c) ) \]

\[ \mathcal{M} \models F \]
\[ F := \exists c \ \forall i \ .\psi(i, a(i), c, a(c)) \]

\[ \mathcal{M} \models F \]
Decision Procedure for the multi-sorted case

\[ F := \exists c \forall i . \psi( i, a(i), c, a(c) ) \]

\[ M \models F \]

\[ a^M \text{ is a } \text{total} \text{ function from } \text{INDEX}^M \text{ to } \text{ELEM}^M \]
Decision Procedure for the multi-sorted case

\[ F := \exists c \forall i . \psi( i, a(i), c, a(c) ) \]

**Step I.** Guess the set of INDEX types

\[ \text{INDEX}^\mathcal{M} \]

\[ \text{ELEM}^\mathcal{M} \]
Decision Procedure for the multi-sorted case

\[ F := \exists c \ \forall i. \psi(i, a(i), c, a(c)) \]

**Step I. Guess the set of INDEX types**
Decision Procedure for the multi-sorted case

\[ F := \exists c \ \forall i \ . \psi( i, a(i), c, a(c) ) \]

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

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

\[
\mathcal{M}_I \models \forall x. \left( \bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, c) \right)
\]
Decision Procedure for the multi-sorted case

\[ F := \exists c \forall i. \psi(i, a(i), c, a(c)) \]

**Step II.** For each type \( M_h \) take a \( b_h \in \text{INDEX}^M \) realizing it.

\[ \text{INDEX}^M \quad \text{ELEM}^M \]
$F := \exists c \forall i . \psi(i, a(i), c, a(c))$

**Step II.** For each type $M_h$ take a $b_h \in INDEX^M$ realizing it.

\[ INDEX^M \]

\[ ELEM^M \]
$F := \exists c \, \forall i . \psi( i, a(i), c, a(c) )$

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

1. Each $b_h$ realizes the corresponding type

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

2. The instantiation

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

is consistent
Decision Procedure for $\text{ARR}^2(T_I, T_E)$

\[
F := \exists c \forall i . \psi( i, a(i), c, a(c) )
\]

\[
F_1 := \exists b \exists c \\
\left[ \forall x. \left( \bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, c) \right) \land \\
\bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, c) \land \\
\bigwedge_{\sigma: i \to b} \psi(i\sigma, a(i\sigma), c, a(c)) \right]
\]
Step III. Substitute the tuple $a(b) \ast a(c)$ with a tuple $e$ of $ELEM$ constants.
Step III. Substitute the tuple $a(b) \ast a(c)$ with a tuple $e$ of ELEM constants.
Decision Procedure for the multi-sorted case

\[ F_1 := \exists b \exists c \left[ \ldots \land \psi(i\sigma, a(i\sigma), c, a(c)) \right] \]

**Step III.** Substitute the tuple \( a(b) \ast a(c) \) with a tuple \( e \) of \( \text{ELEM} \) constants.
Decision Procedure for the multi-sorted case

\[ F_1 := \exists b \exists c \left[ \ldots \land \psi(i\sigma, a(i\sigma), c, a(c)) \right] \]

**Step III.** Substitute the tuple \( a(b) \ast a(c) \) with a tuple \( e \) of \( \text{ELEM} \) constants

\[
F_2 := \exists b \exists c \left[ \ldots \land \neg \psi(b, c, e) \land \land_{d_m, d_n \in b \ast c} (d_m = d_n \rightarrow e_{l,m} = e_{l,n}) \right]
\]
STEP IV. “Split” the formula $F_2$ in INDEX and ELEM parts

\[
F_2 := \exists b \exists c \left[ \forall x. \left( \bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, c) \right) \land 
\bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, c) \land 
\bar{\psi}(b, c, e) \land 
\bigwedge_{d_m, d_n \in b \ast c} \bigwedge_{l=1}^{s} (d_m = d_n \rightarrow e_{l,m} = e_{l,n}) \right]
\]
**Step IV. “Split” the formula $F_2$ in INDEX and ELEM parts**

$$F_2 := \exists b \exists c \left[ \forall x. \left( \bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, c) \right) \land \right.$$ 

$$\left. \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, c) \land \right.$$ 

$$\bar{\psi}(b, c, e) \land \bigwedge_{d_m, d_n \in b \ast c} \bigwedge_{l=1}^{s} (d_m = d_n \rightarrow e_{l,m} = e_{l,n}) \right]$$

$$F_I := \exists b \exists c \left[ \forall x. \left( \bigvee_{j=1}^{q} \bigwedge_{L \in M_j} L(x, c) \right) \land \right.$$ 

$$\left. \bigwedge_{j=1}^{q} \bigwedge_{L \in M_j} L(b_j, c) \land \right.$$ 

$$\bar{\psi}(b, c) \right]$$

$$F_E := \bar{\psi}(e)$$
Decision Procedure for the multi-sorted case

STEP IV. “Split” the formula $F_2$ in INDEX and ELEM parts

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

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

$$F_E := \bar{\psi}(e)$$
STEP IV. “Split” the formula $F_2$ in INDEX and ELEM parts

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

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

$F_E := \bar{\psi}(e)$
Step V. Check if $F_I$ is $T_I$-sat and if $F_E$ is $T_E$-sat
Decision Procedure for the multi-sorted case

**Step V.** Check if $F_I$ is $T_I$-sat and if $F_E$ is $T_E$-sat

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

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

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

[---]
Step V. Check if $F_I$ is $T_I$-sat and if $F_E$ is $T_E$-sat

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

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

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

With divisibility predicates $\{D_k\}_{k \geq 2}$. 
Decision Procedure for the multi-sorted case

**Step V.** Check if $F_I$ is $T_I$-sat and if $F_E$ is $T_E$-sat

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

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

⇒ $\exists^* \forall$-fragment

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

⇒ Quantifier-free fragment

$^1$ With divisibility predicates $\{D_k\}_{k \geq 2}$.  

F. Alberti  Decision Procedures for Flat Array Properties 18 / 21
Application (I) - Deciding the safety of simple$^0_A$-programs

Application: deciding the safety of simple$^0_A$-programs
Application (I) - Deciding the safety of simple$^0_A$-programs

Application: deciding the safety of simple$^0_A$-programs

- Flat control-flow structure
- Every loop $\tau$ has a Flat Array Property as acceleration
Application (I) - Deciding the safety of \( \text{simple}^0_A \)-programs

Application: deciding the safety of \( \text{simple}^0_A \)-programs

- Flat control-flow structure
- Every loop \( \tau \) has a Flat Array Property as acceleration

Theorem

*The unbounded reachability problem for \( \text{simple}^0_A \)-programs is decidable.*
F. Alberti, S. Ghilardi, and N. Sharygina.
Booster: an acceleration-based verification framework for array programs
Application (II) - Booster
An acceleration-based software model-checker
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
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?
Francesco Alberti, Roberto Bruttomesso, Silvio Ghilardi, Silvio Ranise, and Natasha Sharygina.
Lazy abstraction with interpolants for arrays.

Francesco Alberti, Silvio Ghilardi, and Natasha Sharygina.
Definability of accelerated relations in a theory of arrays and its applications.

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 

A.L. Semënov.
Logical theories of one-place functions on the set of natural 
numbers.