What's Decidable About Arrays?

Aaron R. Bradley

Zohar Manna   Henny B. Sipma

Computer Science Department

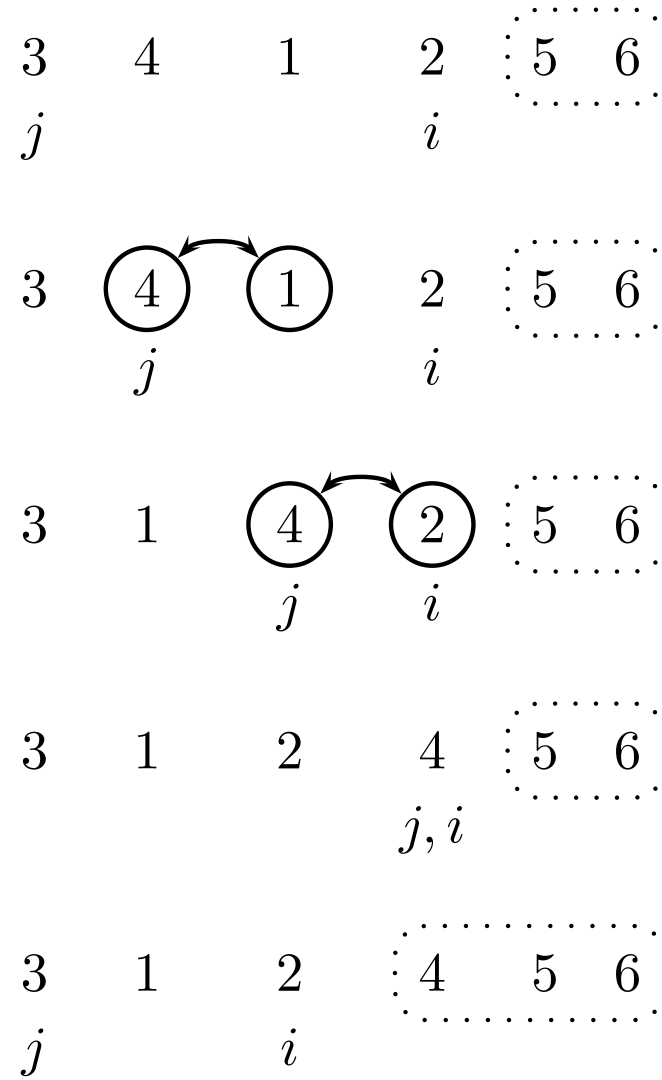
Stanford University

Outline

$\Rightarrow$ 0. Motivation

1. Theories of Arrays
2. $\mathsf{SAT_A}$
4. Undecidable Problems
5. $\mathsf{SAT_M}$
6. Conclusion

Motivation




```
int[] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        t := a[j];
        a[j] := a[j + 1];
        a[j + 1] := t;
      }
    }
  }
  return a;
}
```

Does BubbleSort return a sorted array?

Motivation

ℓ_0: @pre $|a| \geq 0$ — input specification

ℓ_f: @post $\mathsf{sorted}(\mathsf{rv}, 0, |a|-1)$ — output specification

loop assertions

```
int[] BubbleSort(int[] a) {
  int i, j, t;
  for ℓ1: @[ −1 ≤ i < |a| ∧ |a| = |a|0
             ∧ partitioned(a, 0, i, i+1, |a|−1) ∧ sorted(a, i, |a|−1) ]
  (i := |a| − 1; i > 0; i := i − 1)
    for ℓ2: @[ 1 ≤ i < |a| ∧ 0 ≤ j ≤ i ∧ |a| = |a|0
               ∧ partitioned(a, 0, i, i+1, |a|−1)
               ∧ partitioned(a, 0, j−1, j, j) ∧ sorted(a, i, |a|−1) ]
    (j := 0; j < i; j := j + 1)
      if (a[j] > a[j+1]) {
        t := a[j]; a[j] := a[j+1]; a[j+1] := t;
      }
  return a;
}
```

Does BubbleSort return a sorted array? $\boxed{\text{Yes!}}$

Motivation

Predicates:

- $\mathsf{sorted}(a, \ell, u)$: array a is sorted in range $[\ell, u]$

$$(\forall i, j)[\ell \leq i \leq j \leq u \;\rightarrow\; a[i] \leq a[j]]$$

- $\mathsf{partitioned}(a, \ell_1, u_1, \ell_2, u_2)$

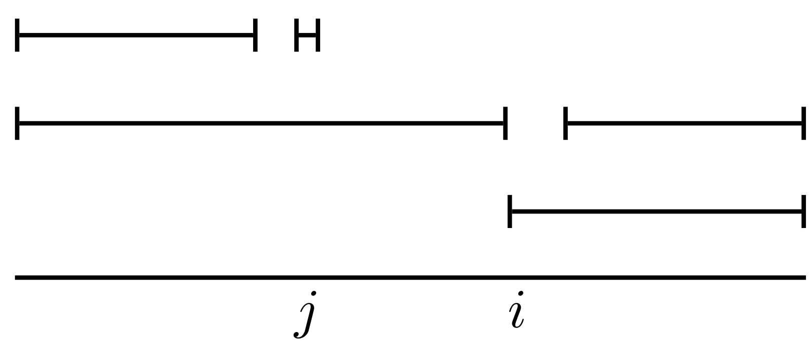
$$(\forall i, j)[\ell_1 \leq i \leq u_1 < \ell_2 \leq j \leq u_2 \;\rightarrow\; a[i] \leq a[j]]$$

At the top of the inner loop (ℓ_2):

$\mathsf{partitioned}(a, 0, j-1, j, j)$

$\mathsf{partitioned}(a, 0, i, i+1, |a|-1)$

$\mathsf{sorted}(a, i, |a|-1)$



Motivation

Verification process:

- Generate **verification conditions**.
- Prove that each verification condition is valid.

Example: verification condition (from ℓ_2 to ℓ_2, with swapping)

$$(\forall *)\left[\begin{array}{l} 1 \le i < |a| \ \wedge\ 0 \le j \le i \ \wedge\ |a| = |a|_0 \\ \wedge\ \mathsf{partitioned}(a, 0, i, i+1, |a|-1) \\ \wedge\ \mathsf{partitioned}(a, 0, j-1, j, j) \ \wedge\ \mathsf{sorted}(a, i, |a|-1) \\ \wedge\ j < i \ \wedge\ a[j] > a[j+1] \\ \rightarrow \begin{array}{l} 1 \le i < |a| \ \wedge\ 0 \le j+1 \le i \ \wedge\ |a| = |a|_0 \\ \wedge\ \mathsf{partitioned}(a\{j \triangleleft a[j+1]\}\{j+1 \triangleleft a[j]\}, 0, i, i+1, |a|-1) \\ \wedge\ \mathsf{partitioned}(a\{j \triangleleft a[j+1]\}\{j+1 \triangleleft a[j]\}, 0, j, j+1, j+1) \\ \wedge\ \mathsf{sorted}(a\{j \triangleleft a[j+1]\}\{j+1 \triangleleft a[j]\}, i, |a|-1) \end{array} \end{array}\right]$$

How do we prove that verification conditions are valid?

Decision Procedures!

Motivation: Parameterized Systems

$\texttt{int[]}\ y := \texttt{int}[0..M-1];$

θ: $y[0] = 1 \;\wedge\; (\forall j \in [1, M-1])\ y[j] = 0$

$$\mathop{\Big|\Big|}_{i\in[0,M-1]} \left[\begin{array}{l} \textbf{request}(y, i); \\ \texttt{while}\ (\texttt{true})\ \ \{ \\ \quad \textbf{critical}; \\ \quad \textbf{release}(y, i \oplus_M 1); \\ \quad \textbf{noncritical}; \\ \quad \textbf{request}(y, i); \\ \} \end{array}\right]$$

$$\begin{aligned} \textbf{request}(y, i) : &\quad y[i] > 0 \;\wedge\; y' = y\{i \triangleleft y[i] - 1\} \\ \textbf{release}(y, i) : &\quad y' = y\{i \triangleleft y[i] + 1\} \end{aligned}$$

Does SEM-N ensure mutual exclusion?

Motivation: Parameterized Systems

$\texttt{int[]}\ y := \texttt{int}[0..M-1];$

θ: $y[0] = 1 \ \wedge\ (\forall j \in [1, M-1])\ y[j] = 0$

$$\underset{i \in [0, M-1]}{||} \left[\begin{array}{l} \textbf{request}(y, i); \\ \texttt{while} \\ \quad \texttt{@}\ (\forall j \in [0, M-1])\ y[j] = 0 \ \wedge\ |y| = |y_0| \\ \quad (\texttt{true}) \\ \{ \\ \quad \textbf{critical}; \\ \quad \textbf{release}(y, i \oplus_M 1); \\ \quad \textbf{noncritical}; \\ \quad \textbf{request}(y, i); \\ \} \end{array} \right]$$

Does SEM-N ensure mutual exclusion? $\boxed{\text{Yes!}}$

Outline

0. Motivation

$\Rightarrow$ 1. Theories of Arrays

2. SAT_A

4. Undecidable Problems

5. SAT_M

6. Conclusion

Background: Extensional Theory of Arrays $T_{\mathsf{A}}^{=}$

$\Sigma = \{\cdot[\cdot],\ \cdot\{\cdot \triangleleft \cdot\},\ =\}$

- Distinguished sorts:
  - Index sort T_{index}
  - Element sort T_{elem}
- Arrays: $\mathsf{index} \rightarrow \mathsf{elem}$, $\mathsf{index} \rightarrow \mathsf{index} \rightarrow \mathsf{elem}$, $\ldots$
- Functions: $\cdot[\cdot]$ (read), $\cdot\{\cdot \triangleleft \cdot\}$ (write)
  - $a[i]$ is the value of the element of array a at index i
  - $a\{i \triangleleft e\}$ is the array a' s.t. $a'[i] = e$, and $a'[j] = a[j]$ at all other $j \neq i$
- Predicates: $=$ (equality)

Example:

$$e_1 \neq e_2 \ \wedge\ a\{i \triangleleft e_1\} = a\{i \triangleleft e_2\} \qquad T_{\mathsf{A}}^{=}\text{-unsatisfiable}$$

Background: Extensional Theory of Arrays $T_{\mathsf{A}}^{=}$

Satisfiability:

- Full: undecidable (even without extensionality)
  - encode FOL
- Quantifier-free: decidable
  - Without extensionality: [McCarthy, 62]
  - With extensionality, $T_{\mathsf{A}}^{=}$:
    [Stump, Barrett, Dill & Levitt, 01]

Parameterized Theory of Maps $T_{\mathsf{M}}^{\mathsf{elem}}$

$\Sigma_{\mathsf{M}} = \Sigma_{\mathsf{EUF}} \cup \Sigma_{\mathsf{elem}} \cup \{\cdot[\cdot], \cdot\{\cdot \triangleleft \cdot\}\}$

- Uninterpreted indices
- Elements interpreted in parameter theory: T_{elem}
  (Note: multiple element theories possible.)
- Maps: $\mathsf{EUF} \rightarrow \mathsf{elem}$, $\mathsf{EUF} \rightarrow \mathsf{EUF} \rightarrow \mathsf{elem}$, $\ldots$
- Functions: functions of T_{EUF} and T_{elem}
  and $\cdot[\cdot]$ (read), $\cdot\{\cdot \triangleleft \cdot\}$ (write)
- Predicates: predicates of T_{EUF} and T_{elem}

Example:

$$e_1 \neq e_2 \;\wedge\; a\{i \triangleleft e_1\} = a\{i \triangleleft e_2\}$$

$$\Downarrow$$

$$e_1 \neq e_2 \;\wedge\; (\forall j)(a\{i \triangleleft e_1\}[j] = a\{i \triangleleft e_2\}[j]) \qquad T_{\mathsf{M}}^{\mathsf{EUF}}\text{-unsatisfiable}$$

Theory of Maps $T_{\mathsf{M}}^{\mathsf{elem}}$

Satisfiability:

- Full: undecidable
- Quantifier-free:
  - decidable for some element theories [McCarthy, 62], [Nelson & Oppen, 79]
  - fragment augmented with permutation predicate decidable [Suzuki & Jefferson, 80]
- With **map properties**: decidable
  - One alternation of quantifiers, with syntactic constraints
  - "Natural" fragment: on the edge of decidability

Parameterized Theory of Arrays $T_{\mathsf{A}}^{\mathsf{elem}}$

$\Sigma_{\mathsf{A}} = \Sigma_{\mathbb{Z}} \cup \Sigma_{\mathsf{elem}} \cup \{\cdot[\cdot], \cdot\{\cdot \triangleleft \cdot\}\}$

- Indices interpreted in Presburger arithmetic $T_{\mathbb{Z}}$
- Elements interpreted in parameter theories: T_{elem}
  (Note: multiple element theories possible.)
- Arrays: $\mathbb{Z} \to \mathsf{elem}$, $\mathbb{Z} \to \mathbb{Z} \to \mathsf{elem}$, $\ldots$
- Functions: functions of $T_{\mathbb{Z}}$ and T_{elem}
  and $\cdot[\cdot]$ (read), $\cdot\{\cdot \triangleleft \cdot\}$ (write)
- Predicates: predicates of $T_{\mathbb{Z}}$ and T_{elem}

Example:

$$\begin{aligned} & \mathsf{sorted}(0, 5, a\{0 \triangleleft 7\}\{5 \triangleleft 9\}) \\ \wedge\ & \mathsf{sorted}(0, 5, a\{0 \triangleleft 11\}\{5 \triangleleft 13\}) \end{aligned} \qquad T_{\mathsf{A}}^{\mathbb{Z}}\text{-unsatisfiable}$$

Theory of Arrays $T_{\mathsf{A}}^{\mathsf{elem}}$

Satisfiability:

- Full: undecidable
- Quantifier-free: decidable
  - decidable for some element theories [McCarthy, 62], [Nelson & Oppen, 79]
  - fragment with sorted and partitioned predicates decidable [Mateti, 81]
- With **array properties**: decidable
  - One alternation of quantifiers, with syntactic constraints
  - "Natural" fragment: on the edge of decidability

Our Contribution

- Studied theories of arrays (integer indices) and maps (uninterpreted indices).
- Identified decidable subfragments and provided decision procedures.
- Showed that several natural extensions result in undecidable fragments.
- Implementation in $\pi\mathsf{VC}$, a verifying compiler.

Outline

0. Motivation
1. Theories of Arrays

$\Rightarrow$ 2. $\mathsf{SAT_A}$

4. Undecidable Problems
5. $\mathsf{SAT_M}$
6. Conclusion

$T_{\mathsf{A}}^{\mathsf{elem}}$: Array Property

A formula of the form

$$(\forall \bar{i})(\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i}))$$

where

- $\bar{i} \in \mathbb{Z}^n$ is vector of index variables
- n is the **height** of the property
- $\varphi_I(\bar{i})$ is the **index guard**
- $\varphi_V(\bar{i})$ is the **value constraint**

$T_{\mathsf{A}}^{\mathsf{elem}}$: Array Property

$$(\forall \bar{i})(\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i}))$$

Index guard $\varphi_I(\bar{i})$:

$$\begin{array}{rcl}
\mathsf{iguard} & \rightarrow & \mathsf{iguard} \wedge \mathsf{iguard} \mid \mathsf{iguard} \vee \mathsf{iguard} \mid \mathsf{atom} \\
\mathsf{atom} & \rightarrow & \mathsf{expr} \leq \mathsf{expr} \mid \mathsf{expr} = \mathsf{expr} \\
\mathsf{expr} & \rightarrow & uvar \mid \mathsf{pexpr} \\
\mathsf{pexpr} & \rightarrow & \mathbb{Z} \mid \mathbb{Z} \cdot evar \mid \mathsf{pexpr} + \mathsf{pexpr}
\end{array}$$

uvar is any universally quantified variable.
evar is any existentially quantified integer variable.

Example:

$$(\forall i, j)(\ell + 2 \leq i \ \wedge \ i \leq u + 2k - 1 \ \wedge \ i \leq j \ \wedge \ 2\ell \leq 3u \ \rightarrow \ \ldots)$$

$T_{\mathsf{A}}^{\mathsf{elem}}$: Array Property

$$(\forall \bar{i})(\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i}))$$

Value constraint $\varphi_V(\bar{i})$:
All occurrences of $i \in \bar{i}$ in $\varphi_V(\bar{i})$ are as reads $a[i]$.
No nested reads $a_1[a_2[i]]$.

Example:

$$(\forall i, j)(i \leq j \;\rightarrow\; a[i] \leq a[j])$$

$T_{\mathsf{A}}^{\mathsf{elem}}$: Array Property Fragment

Subfragment of $\exists^*\forall^*$-fragment of $T_{\mathsf{A}}^{\mathsf{elem}}$.

Existentially-closed Boolean combinations of array properties and quantifier-free $T_{\mathsf{A}}^{\mathsf{elem}}$-formulae.

Example: validity of

$$
(\forall *)\ \left[\begin{array}{l}
1 \leq i < |a| \ \wedge\ 0 \leq j \leq i \ \wedge\ |a| = |a|_0 \\
\wedge\ \mathsf{partitioned}(a, 0, i, i+1, |a|-1) \\
\wedge\ \mathsf{partitioned}(a, 0, j-1, j, j) \ \wedge\ \mathsf{sorted}(a, i, |a|-1) \\
\wedge\ j < i \ \wedge\ a[j] > a[j+1] \\
\quad 1 \leq i < |a| \ \wedge\ 0 \leq j+1 \leq i \ \wedge\ |a| = |a|_0 \\
\rightarrow \quad \wedge\ \mathsf{partitioned}(a\{j \triangleleft a[j+1]\}\{j+1 \triangleleft a[j]\}, 0, i, i+1, |a|-1) \\
\quad \wedge\ \mathsf{partitioned}(a\{j \triangleleft a[j+1]\}\{j+1 \triangleleft a[j]\}, 0, j, j+1, j+1) \\
\quad \wedge\ \mathsf{sorted}(a\{j \triangleleft a[j+1]\}\{j+1 \triangleleft a[j]\}, i, |a|-1)
\end{array}\right]
$$

Examples: Definable Predicates

$T_{\mathsf{A}}^{\mathsf{elem}}$:

$(\forall i)(a[i] = b[i])$ $\qquad a = b$

$(\forall i)(\ell \leq i \leq u \;\rightarrow\; a[i] = b[i])$ $\qquad \mathsf{beq}(\ell, u, a, b)$

$T_{\mathsf{A}}^{\mathbb{Z}}$, $T_{\mathsf{A}}^{\mathbb{R}}$:

$(\forall i, j)(\ell \leq i \leq j \leq u \;\rightarrow\; a[i] \leq a[j])$ $\qquad \mathsf{sorted}(\ell, u, a)$

$(\forall i, j)(\ell_1 \leq i \leq u_1 < \ell_2 \leq j \leq u_2 \;\rightarrow\; a[i] \leq a[j])$

$\mathsf{partitioned}(\ell_1, u_1, \ell_2, u_2, a)$

Also “weak” permutation as approximation to permutation of [Suzuki & Jefferson, 80]

Algorithm: SAT_A

Reduction from array property formula ψ
to equisatisfiable quantifier-free $(T_{\mathsf{EUF}} \cup T_{\mathbb{Z}} \cup T_{\mathsf{elem}})$-formula ψ'.

1. Expand definitions. Convert to negation normal form (NNF).
2. Apply (write) to remove writes.
3. Apply (exists) to remove $\exists$.
4. Construct **index set**.
   Apply (forall) to remove $\forall$.
5. Convert to equisatisfiable quantifier-free
   $(T_{\mathsf{EUF}} \cup T_{\mathbb{Z}} \cup T_{\mathsf{elem}})$-formula.
   Decide satisfiability in combined theory.

Algorithm: $\mathsf{SAT_A}$ (**Step 1**)

Replace literals with definitions.

Push negations past quantifiers.

Example:

$$\varphi: \ \neg\mathsf{sorted}(\ell, u, a\{k \triangleleft e\})$$

$$\Downarrow$$

$$\neg((\forall i, j)(\ell \leq i \leq j \leq u \ \rightarrow \ a\{k \triangleleft e\}[i] \leq a\{k \triangleleft e\}[j]))$$

$$\Downarrow$$

$$(\exists i, j)(\ell \leq i \leq j \leq u \ \wedge \ a\{k \triangleleft e\}[i] > a\{k \triangleleft e\}[j])$$

Algorithm: $\mathsf{SAT_A}$ (**Step 2**)

Apply (write) exhaustively to remove writes:

$$\frac{\psi[a\{i \triangleleft e\}]}{\psi[b] \;\wedge\; b[i] = e \;\wedge\; (\forall j)(j \neq i \;\rightarrow\; a[j] = b[j])} \text{ for fresh } b \qquad \text{(write)}$$

To meet syntactic requirements of index guard, rewrite third conjunct:

$$(\forall j)(j \leq i - 1 \;\vee\; i + 1 \leq j \;\rightarrow\; a[j] = b[j])$$

Example:

$$(\exists i, j)(\ell \leq i \leq j \leq u \;\wedge\; a\{k \triangleleft e\}[i] > a\{k \triangleleft e\}[j])$$

$$\Downarrow$$

$$(\exists i, j)(\ell \leq i \leq j \leq u \;\wedge\; b[i] > b[j]) \;\wedge\; b[k] = e$$
$$\wedge\; (\forall j)(j \neq k \;\rightarrow\; a[j] = b[j])$$

Algorithm: $\mathsf{SAT}_{\mathsf{A}}$ (**Step 3**)

Apply (exists) rule exhaustively:

$$\frac{\psi[(\exists \bar{i})(\varphi_I(\bar{i}) \wedge \neg\varphi_V(\bar{i}))}{\psi[\varphi_I(\bar{j}) \wedge \neg\varphi_V(\bar{j})]} \text{ for fresh } \bar{j} \qquad \text{(exists)}$$

Example:

$$(\exists i, j)(\ell \leq i \leq j \leq u \ \wedge \ b[i] > b[j]) \ \wedge \ b[k] = e$$
$$\wedge \ (\forall j)(j \neq k \ \rightarrow \ a[j] = b[j])$$
$$\Downarrow$$
$$\ell \leq j_1 \leq j_2 \leq u \ \wedge \ b[j_1] > b[j_2] \ \wedge \ b[k] = e$$
$$\wedge \ (\forall j)(j \neq k \ \rightarrow \ a[j] = b[j])$$

Algorithm: SAT_A (Step 4)

Form **index set** $\mathcal{I}$, and apply (forall) exhaustively.

$$\frac{\psi[(\forall\bar{i})(\varphi_I(\bar{i}) \to \varphi_V(\bar{i}))]}{\psi\left[\bigwedge_{\bar{i}\in\mathcal{I}^n} (\varphi_I(\bar{i}) \to \varphi_V(\bar{i}))\right]} \quad \textsf{(forall)}$$

What is $\mathcal{I}$?

Algorithm: Index Sets of ψ

Read Set: Array reads $\mathcal{R} = \{t \ : \ \cdot[t] \in \psi \ \wedge \ t \text{ not } \forall \text{ quantified}\}$.

Bounds Set: $\mathcal{B}$, root pexpr terms in index guards.

Full Index Set:

$$\mathcal{I} \stackrel{\text{def}}{=} \begin{cases} \{0\} & \text{if } \mathcal{R} = \mathcal{B} = \emptyset \\ \mathcal{R} \cup \mathcal{B} & \text{otherwise} \end{cases}$$

Example:

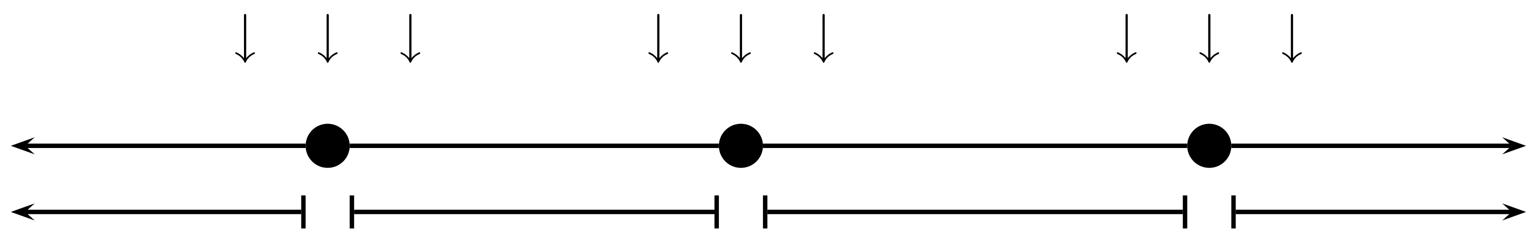
$$b[k] = e \ \wedge \ (\forall i)(\quad \underbrace{i \neq k}_{i \leq k-1 \ \vee \ i \geq k+1} \quad \rightarrow \ a[i] = b[i])$$

$$\mathcal{R} = \{k\}$$
$$\mathcal{B} = \{k-1, k+1\}$$
$$\mathcal{I} = \{k-1, k, k+1\}$$

Algorithm: Index Set of ψ

New indices $(k-1,\, k+1)$ from (write) **represent intervals**:

- closed intervals between symbolic indices and bounds
- two open intervals: to the left and to the right



Algorithm: $\mathsf{SAT_A}$ (Step 4)

Example:

$$\ell \le j_1 \le j_2 \le u \ \wedge\ b[j_1] > b[j_2] \ \wedge\ b[k] = e$$
$$\wedge\ (\forall j)(j \ne k \ \rightarrow\ a[j] = b[j])$$

$$\mathcal{R} = \{j_1, j_2, k\} \quad \mathcal{B} = \{k-1, k+1\}$$
$$\mathcal{I} = \{j_1, j_2, k-1, k, k+1\}$$

$$\Downarrow$$

$$\ell \le j_1 \le j_2 \le u \ \wedge\ b[j_1] > b[j_2] \ \wedge\ b[k] = e$$
$$\wedge\ \bigwedge_{j \in \mathcal{I}} (j \ne k \ \rightarrow\ a[j] = b[j])$$

Algorithm: $\mathsf{SAT_A}$ (**Step 5**)

- Associate with each n-dimensional array variable a a fresh uninterpreted n-ary function f_a.
- Replace each array read $a[i, \ldots, j]$ by $f_a(i, \ldots, j)$.
- Decide this (QF) formula's satisfiability in $T_{\mathsf{EUF}} \cup T_{\mathbb{Z}} \cup T_{\mathsf{elem}}$.

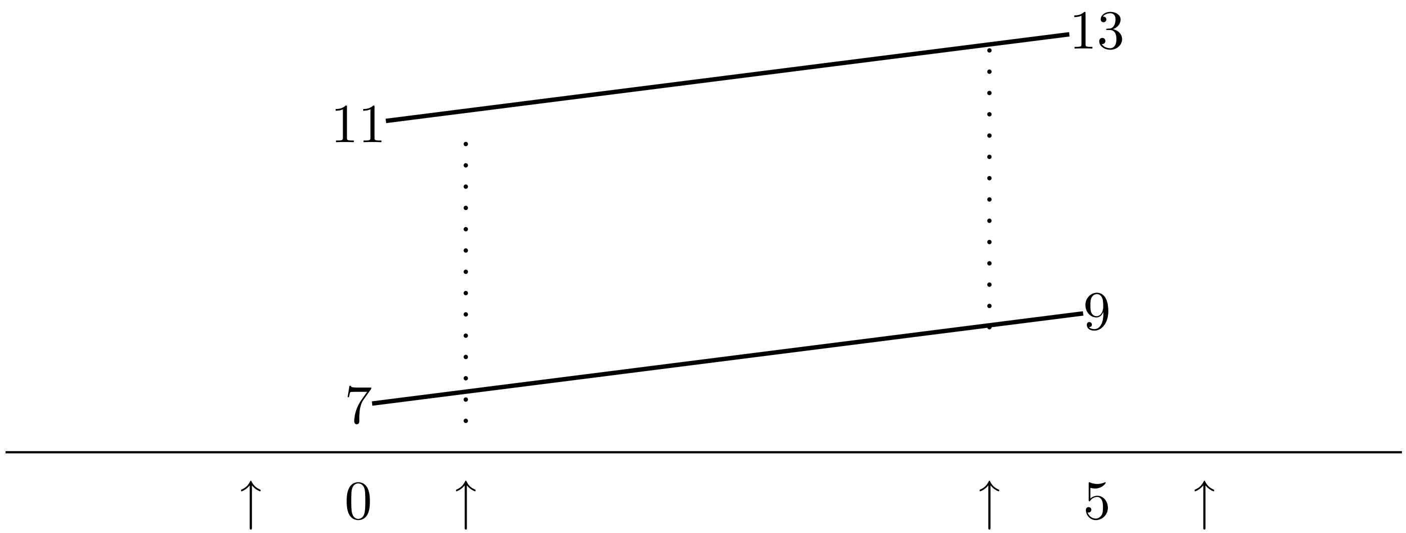
Example:

$$\ell \leq j_1 \leq j_2 \leq u \ \wedge \ b[j_1] > b[j_2] \ \wedge \ b[k] = e$$
$$\wedge \ \bigwedge_{j \in \mathcal{I}} (j \neq k \ \rightarrow \ a[j] = b[j])$$
$$\Downarrow$$
$$\ell \leq j_1 \leq j_2 \leq u \ \wedge \ f_b(j_1) > f_b(j_2) \ \wedge \ f_b(k) = e$$
$$\wedge \ \bigwedge_{j \in \mathcal{I}} (j \neq k \ \rightarrow \ f_a(j) = f_b(j))$$

$(T_{\mathsf{EUF}} \cup T_{\mathbb{Z}})$-satisfiable $\Rightarrow \varphi$ is $T_{\mathsf{A}}^{\mathbb{Z}}$-satisfiable.

Example: $T_{\mathsf{A}}^{\mathbb{Z}}$-unsatisfiable

$$\varphi : \begin{array}{l} \quad \mathsf{sorted}(0, 5, a\{0 \triangleleft 7\}\{5 \triangleleft 9\}) \\ \wedge\ \mathsf{sorted}(0, 5, a\{0 \triangleleft 11\}\{5 \triangleleft 13\}) \end{array}$$



Example: $T_{\mathsf{A}}^{\mathbb{Z}}$-unsatisfiable (**Step 1**)

$$\varphi : \quad \begin{array}{l} \quad\ \mathsf{sorted}(0, 5, a\{0 \triangleleft 7\}\{5 \triangleleft 9\}) \\ \wedge\ \mathsf{sorted}(0, 5, a\{0 \triangleleft 11\}\{5 \triangleleft 13\}) \end{array}$$

$$\Downarrow$$

$$\begin{array}{l} \quad\ (\forall i, j)(0 \leq i \leq j \leq 5 \ \rightarrow\ a\{0 \triangleleft 7\}\{5 \triangleleft 9\}[i] \leq a\{0 \triangleleft 7\}\{5 \triangleleft 9\}[j]) \\ \wedge\ (\forall i, j)(0 \leq i \leq j \leq 5 \ \rightarrow\ a\{0 \triangleleft 11\}\{5 \triangleleft 13\}[i] \leq a\{0 \triangleleft 11\}\{5 \triangleleft 13\}[j]) \end{array}$$

Example: $T_{\mathsf{A}}^{\mathbb{Z}}$-unsatisfiable (**Step 2**)

$$\begin{aligned} & (\forall i, j)(0 \leq i \leq j \leq 5 \ \rightarrow \ \underbrace{a\{0 \triangleleft 7\}}_{b}\{5 \triangleleft 9\}[i] \leq \underbrace{a\{0 \triangleleft 7\}\{5 \triangleleft 9\}}_{c}[j]) \\ \wedge \ & (\forall i, j)(0 \leq i \leq j \leq 5 \ \rightarrow \ \underbrace{a\{0 \triangleleft 11\}}_{d}\{5 \triangleleft 13\}[i] \leq \underbrace{a\{0 \triangleleft 11\}\{5 \triangleleft 13\}}_{e}[j]) \end{aligned}$$

$$\Downarrow$$

$$\begin{aligned} & (\forall i, j)(\boxed{0 \leq i \leq j \leq 5} \ \rightarrow \ c[i] \leq c[j]) \\ \wedge \quad & (\forall i, j)(0 \leq i \leq j \leq 5 \ \rightarrow \ e[i] \leq e[j]) \\ \wedge \quad & (\forall i)(\boxed{i \neq 5} \ \rightarrow \ b[i] = c[i]) \ \wedge \ c[\boxed{5}] = 9 \\ \wedge \quad & (\forall i)(\boxed{i \neq 0} \ \rightarrow \ a[i] = b[i]) \ \wedge \ b[\boxed{0}] = 7 \\ \wedge \quad & (\forall i)(i \neq 5 \ \rightarrow \ d[i] = e[i]) \ \wedge \ e[5] = 13 \\ \wedge \quad & (\forall i)(i \neq 0 \ \rightarrow \ a[i] = d[i]) \ \wedge \ d[0] = 11 \end{aligned}$$

$$\mathcal{R} = \{0, 5\} \quad \mathcal{B} = \{0, 5, \underbrace{-1, 1}_{i \neq 0}, \underbrace{4, 6}_{i \neq 5}\} \quad \mathcal{I} = \{-1, 0, 1, 4, 5, 6\}$$

Example: $T_{\mathsf{A}}^{\mathbb{Z}}$-unsatisfiable

In particular,

$$c[1] \leq c[5] = 9 < 11 = d[0] \leq d[1]$$

and

$$c[1] = b[1] = a[1] = d[1]$$

Contradiction.

$\Rightarrow \varphi$ is $T_{\mathsf{A}}^{\mathbb{Z}}$-unsatisfiable.

Note: New indices essential for proof.

Correctness

Theorem.
If satisfiability of quantifier-free $(T_{\mathsf{EUF}} \cup T_{\mathbb{Z}} \cup T_{\mathsf{elem}})$-formulae is decidable, then $\mathsf{SAT}_{\mathsf{A}}$ is a decision procedure for satisfiability in the array property fragment of $T_{\mathsf{A}}^{\mathsf{elem}}$.

Outline

0. Motivation
1. Theories of Arrays
2. SAT_A

$\Rightarrow$ 4. Undecidable Problems

5. SAT_M
6. Conclusion

Undecidable Problems

An undecidable fragment results (for some element theory) with

- $\exists^*\forall_{\mathbb{Z}}\exists_{\mathbb{Z}}$-fragment
  (with same restrictions otherwise)
- nested reads (*e.g.*, $a_1[a_2[i]]$, where i is universally quantified);
- array reads by $\forall$ variable in index guard;
- general arithmetic expressions over $\forall$ index variables
  (even just addition of 1, *e.g.*, $i + 1$)

Open Question

Fragment with $<$ in index guard (equivalently, negation).

Could express that an array has unique elements:

$$(\forall i, j)(i < j \;\rightarrow\; a[i] \neq a[j])$$

Outline

0. Motivation
1. Theories of Arrays
2. SAT_A
4. Undecidable Problems

$\Rightarrow$ 5. SAT_M

6. Conclusion

Maps: Map Property Fragment

$$(\forall \overline{k})(\varphi_K(\overline{k}) \rightarrow \varphi_V(\overline{k}))$$

Key Guard $\varphi_K(\overline{k})$

$$\begin{array}{rcl}
\mathsf{kguard} & \rightarrow & \mathsf{kguard} \wedge \mathsf{kguard} \mid \mathsf{kguard} \vee \mathsf{kguard} \mid \mathsf{atom} \\
\mathsf{atom} & \rightarrow & \mathsf{var} = \mathsf{var} \mid evar \neq \mathsf{var} \mid \mathsf{var} \neq evar \\
\mathsf{var} & \rightarrow & evar \mid uvar
\end{array}$$

Value Constraint $\varphi_V(\overline{k})$:
All occurrences of $k \in \overline{k}$ in $\varphi_V(\overline{k})$ are as reads $h[k]$.
No nested reads $h_1[h_2[k]]$.

Fragment:
Existentially-closed Boolean combinations of map property formulae and quantifier-free $T_{\mathsf{M}}^{\mathsf{elem}}$-formulae.

Algorithm: $\mathsf{SAT_M}$

1. Step 1 of $\mathsf{SAT_A}$.

2. $$\frac{\psi[h\{k \triangleleft e\}]}{\psi[h'] \wedge h'[k] = e \wedge (\forall j)(j \neq k \ \rightarrow \ h[j] = h'[j])} \quad \text{(write)}$$

3. Step 3 of $\mathsf{SAT_A}$.

4. $\mathcal{R} = \{t \ : \ \cdot[t] \in \psi\}$; $\mathcal{B}$ contains *evar*s of key guards; $\mathcal{K} = \mathcal{R} \cup \mathcal{B} \cup \{\kappa\}$, for fresh κ.

$$\frac{\psi[(\forall \overline{k})(\varphi_K(\overline{k}) \rightarrow \varphi_V(\overline{k}))]}{\psi\left[\displaystyle\bigwedge_{\overline{k} \in \mathcal{K}^n_{\psi_3}} (\varphi_K(\overline{k}) \rightarrow \varphi_V(\overline{k}))\right]} \quad \text{(forall)}$$

5. Construct $\psi_4 \ \wedge \ \bigwedge_{k \in \mathcal{K} \setminus \{\kappa\}} k \neq \kappa$.

6. Step 5 of $\mathsf{SAT_A}$; procedure for $T_{\mathsf{EUF}} \cup T_{\mathsf{elem}}$.

Maps

Theorem.
If satisfiability of quantifier-free $(T_{\mathsf{EUF}} \cup T_{\mathsf{elem}})$-formulae is decidable, then $\mathsf{SAT}_{\mathsf{M}}$ is a decision procedure for satisfiability in the map property fragment of $T_{\mathsf{M}}^{\mathsf{elem}}$.

Relevant undecidability proofs carry over to maps.

Application: DP for Hashtables

Operations:

- $\mathsf{put}(h, k, v)$: map k to v
- $\mathsf{remove}(h, k)$ remove mapping on k
- $\mathsf{get}(h, k)$: return mapped element

Predicates:

- $\mathsf{init}(h)$: true iff h maps nothing
- $k \in \mathsf{keys}(h)$: key membership
- $k \in K_1 \cup K_2$
- $k \in K_1 \cap K_2$
- $k \in \overline{K}$

Application: DP for Hashtables

1. Construct $\psi \ \wedge \ \top \neq \bot$, for fresh constants $\top$ and $\bot$.

2. Rewrite

$$\begin{aligned}
\psi[\mathsf{put}(h,k,v)] \ &\Rightarrow \ \psi[h'] \wedge h' = h\{k \triangleleft v\} \wedge \mathsf{keys}_{h'} = \mathsf{keys}_h\{k \triangleleft \top\} \\
\psi[\mathsf{remove}(h,k)] \ &\Rightarrow \ \psi[h'] \wedge \mathsf{keys}_{h'} = \mathsf{keys}_h\{k \triangleleft \bot\}
\end{aligned}$$

3. Rewrite

$$\begin{aligned}
\psi[\mathsf{get}(h,k)] \ &\Rightarrow \ \psi[h[k]] \\
\psi[\mathsf{init}(h)] \ &\Rightarrow \ \psi[(\forall k)(h[k] = \bot)] \\
\psi[k \in \mathsf{keys}(h)] \ &\Rightarrow \ \psi[\mathsf{keys}_h[k] \neq \bot] \\
\psi[k \in K_1 \cup K_2] \ &\Rightarrow \ \psi[k \in K_1 \ \vee \ k \in K_2] \\
\psi[k \in K_1 \cap K_2] \ &\Rightarrow \ \psi[k \in K_1 \ \wedge \ k \in K_2] \\
\psi[k \in \overline{K}] \ &\Rightarrow \ \psi[\neg(k \in K)]
\end{aligned}$$

Example: Hashtables

Example specification:

$$(\forall k \in \mathsf{keys}(h))(\mathsf{get}(h,k) \geq 0)$$

Example verification condition:

$$(\forall h, s, v, h')$$
$$\left[\begin{array}{l}(\forall k \in \mathsf{keys}(h))\ \mathsf{get}(h,k) \geq 0\ \wedge\ v \geq 0\ \wedge\ h' = \mathsf{put}(h,s,v) \\ \quad \rightarrow\ (\forall k \in \mathsf{keys}(h'))\ \mathsf{get}(h',k) \geq 0\end{array}\right]$$

Remark:

Key sets provide means for reasoning about modifying hashtables.

Outline

0. Motivation
1. Theories of Arrays
2. SAT_A
4. Undecidable Problems
5. SAT_M

$\Rightarrow$ 6. Conclusion

Experience

Implemented in πVC, a verifying compiler for pi language.
(pi for **Prove It**)

- Used in undergraduate/graduate course at Stanford.
- Relied heavily on array decision procedure.

Tricks:

- Simple resolution.
- Phased introduction of symbolic index terms.

Performance results (examples):

Program	**Time**	**Program**	**Time**
MergeSort	20s	BubbleSort	1s
InsertionSort	1s	QuickSort	7s
BinarySearch	1s	Sem-N	1s

Future Work

- Decidability of extension with $<$ ($\neq$).
- Complexity for particular element theories.
- Array invariant generation.

Thank you!

SAT_A is Sound

Lemma. If ψ is satisfiable, then ψ_5 is satisfiable.

Proof. Universal quantification weakened to finite conjunction.

$\mathsf{SAT_A}$ is Complete

Lemma. If ψ_5 is satisfiable, then ψ is satisfiable.

Proof.

Assume

$$I \models \psi_5$$

(I is a model of ψ_5).

Construct model J from I s.t.

$$J \models \psi$$

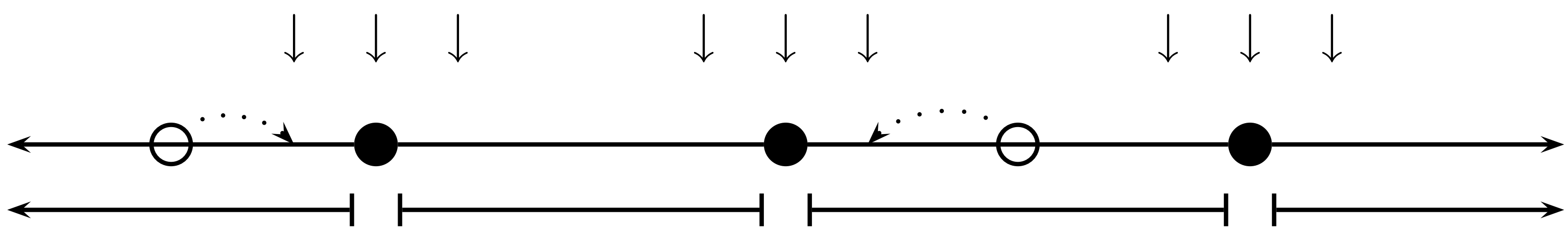
$\mathsf{SAT_A}$ is Complete

Define project operation under I

$$\mathsf{proj} : \mathbb{Z} \to \mathcal{I}^I$$

$\mathsf{proj}(z) = t^I$ such that

- $t \in \mathcal{I}$
- t^I is nearest neighbor to z
  - $t^I \leq z \ \wedge \ (\forall s \in \mathcal{I})(s^I \leq t^I \ \vee \ s^I > z)$
  - or $t^I > z \ \wedge \ (\forall s \in \mathcal{I})(s^I \geq t^I)$



Extend $\mathsf{proj}(z_1, \ldots, z_k) = (\mathsf{proj}(z_1), \ldots, \mathsf{proj}(z_k))$.

$\mathsf{SAT_A}$ is Complete

Construct model J of ψ:

- Equate all non-array variables in J and I.
- $a^J[\bar{z}] = f_a^J(\mathsf{proj}(\bar{z}))$

Now prove $J \models \psi$.

Steps 1, 3, 5 are easy.

Step 2 implements the definition of *array write*.

Focus on Step 4.

$$\frac{\psi[(\forall \bar{i})(\varphi_I(\bar{i}) \to \varphi_V(\bar{i}))]}{\psi\left[\bigwedge_{\bar{i} \in \mathcal{I}^n} (\varphi_I(\bar{i}) \to \varphi_V(\bar{i}))\right]} \quad (\mathsf{forall})$$

$\mathsf{SAT_A}$ is Complete

Strategy:

Suppose forall is applied to ψ'. Prove that if

$$J \models \psi' \left[\bigwedge_{\bar{i} \in \mathcal{I}_\psi^n} (\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i})) \right]$$

then

$$J \models \psi' \left[(\forall \bar{i})(\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i})) \right]$$

Sufficient to prove

$$J \models \left[\bigwedge_{\bar{i} \in \mathcal{I}_\psi^n} (\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i})) \ \rightarrow \ (\forall \bar{i})(\varphi_I(\bar{i}) \rightarrow \varphi_V(\bar{i})) \right]$$

$\mathsf{SAT_A}$ is Complete

$$
J \models \quad
\begin{tikzcd}
\varphi_I(\mathsf{proj}(\overline{z})) \arrow[r] & \varphi_V(\mathsf{proj}(\overline{z})) \arrow[d, dashed, "(2)"] \\
\varphi_I(\overline{z}) \arrow[u, dashed, "(1)"] \arrow[r, "?"] & \varphi_V(\overline{z})
\end{tikzcd}
$$

1. $\ell^J \leq m^J \;\Rightarrow\; \mathsf{proj}(\ell^J) \leq \mathsf{proj}(m^J)$
   $\ell^J = m^J \;\Rightarrow\; \mathsf{proj}(\ell^J) = \mathsf{proj}(m^J)$
   (structural induction over index guard)

2. $a^J[\overline{z}] = a^J[\mathsf{proj}(\overline{z})]$

Q.E.D.

$\mathsf{SAT_A}$ is a Decision Procedure

Theorem.
If satisfiability of quantifier-free $(T_{\mathsf{EUF}} \cup T_{\mathbb{Z}} \cup T_{\mathsf{elem}})$-formulae is decidable, then $\mathsf{SAT_A}$ is a decision procedure for satisfiability in the array property fragment of $T_{\mathsf{A}}^{\mathsf{elem}}$.

Theorem.
If satisfiability of quantifier-free $(T_{\mathsf{EUF}} \cup T_{\mathbb{Z}} \cup T_{\mathsf{elem}})$-formulae is in NP, then for the subfragment of the array property fragment of $T_{\mathsf{A}}^{\mathsf{elem}}$ in which all array property formulae have height at most N, satisfiability is NP-complete.

Proof.
Polynomial (in $|\psi|$) increase in size of formula.
Polynomial (in $|\psi|$) number of rule applications.

Undecidable Problems

Theorem.
Satisfiability of the $\exists^*\forall_{\mathbb{Z}}\exists_{\mathbb{Z}}$-fragment of both $T_{\mathsf{A}}^{\mathbb{R}}$ and $T_{\mathsf{A}}^{\mathbb{Z}}$ is undecidable, even with syntactic restrictions like in the array property fragment.

Proof.
Reduce from termination:
Given loop L, construct formula φ_L that is unsatisfiable iff L always terminates.

Undecidable Problems

Lemma. Termination of loops of this form is undecidable:

$$
\begin{array}{l}
\textbf{real } x_1, \ldots, x_n \\
\theta : \displaystyle\bigwedge_{i \in I \subseteq \{1,\ldots,n\}} x_i = c_i \\
\textbf{while } x_1 \geq 0 \textbf{ do} \\
\quad \textbf{choose } \tau_i : \ \mathbf{x} := A_i \mathbf{x} \\
\textbf{done}
\end{array}
$$

- $c_i \in \mathbb{Z}$, $A_i \in \mathbb{Z}^{n \times n}$
- θ: initial condition
- $x_1, \ldots, x_n$ range over $\mathbb{R}$ (or $\mathbb{Z}$)

See *Polyranking for Polynomial Loops*, available at `http://theory.stanford.edu/~arbrad`, for proof.

Undecidable Problems

- One array variable x_i per loop variable x_i.
- Encode transitions:

$$\rho_\tau(s,t) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} x_i[t] = A_{i,1} \cdot x_1[s] + \cdots + A_{i,n} \cdot x_n[s]$$

- Encode guard:

$$g(s) \stackrel{\text{def}}{=} x_1[s] \geq 0$$

- Encode initial condition:

$$\theta(s) \stackrel{\text{def}}{=} \bigwedge_{i \in I \subseteq \{1,\ldots,n\}} x_i[s] = c_i$$

- Construct φ:

$$\varphi: (\exists x_1, \ldots, x_n, z)(\forall i)(\exists j) \left[\theta(z) \ \wedge \ g(z) \ \wedge \ \bigvee_k \rho_{\tau_k}(i,j) \ \wedge \ g(j) \right]$$

Undecidable Problems

Theorem.

Extending the array property fragment with any of

- nested reads (*e.g.*, $a_1[a_2[i]]$, where i is universally quantified);
- array reads by a universally quantified variable in the index guard;
- general Presburger arithmetic expressions over universally quantified index variables (even just addition of 1, *e.g.*, $i + 1$) in the index guard or in the value constraint

results in a fragment of $T_{\mathsf{A}}^{\mathbb{Z}}$ for which satisfiability is undecidable.

Proof.

Similar reduction from termination.