Arrays Made Simpler : An Efficient, Scalable and Thorough Preprocessing

Logic for Programming, Artificial Intelligence and Reasoning

Benjamin Farinier — CEA List
Robin David — Quarkslab
Sébastien Bardin — CEA List
Matthieu Lemerre — CEA List

Overview

Software verification

- More and more rely on decision procedures

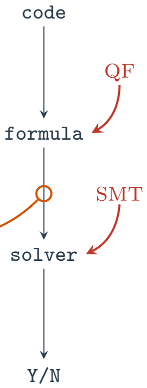
Array theory

- Useful for modelling memory or data structures
- Bottleneck of resolution for large formulas (BMC, SE)

Our proposal

- FAS, an efficient simplification for array theory

$\Longrightarrow$ Improve existing solvers



Array theory

Two basic operations on arrays

- Reading in a at index $i \in \mathcal{I}$: $a[i]$
- Writing in a an element $e \in \mathcal{E}$ at index $i \in \mathcal{I}$: $a[i] \leftarrow e$

$$\begin{aligned} \text{read} &: \text{Array}\ \mathcal{I}\ \mathcal{E} \rightarrow \mathcal{I} \rightarrow \mathcal{E} \\ \text{write} &: \text{Array}\ \mathcal{I}\ \mathcal{E} \rightarrow \mathcal{I} \rightarrow \mathcal{E} \rightarrow \text{Array}\ \mathcal{I}\ \mathcal{E} \end{aligned}$$

$$\forall a\ i\ j\ e.\ (a[i] \leftarrow e)[j] = \begin{cases} e & \text{if } i = j \\ a[j] & \text{otherwise} \end{cases}$$

Prevalent in software analysis

- Modelling memory
- Abstracting data structure (map, queue, stack...)

Hard to solve

- NP-complete
- ROW may require case-splits

Arrays in practice

Unrolling-based verification techniques (BMC, SE)

- may produce huge formula
- high number of reads and writes

In some extremes cases, solvers may spend hours of these formulas

Without proper simplification, array theory might become a bottleneck for resolution

What should we simplify ? Read-Over-Write (ROW) !

ROW simplification

$$
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\quad \Longrightarrow \quad
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_0\,[1415] = 9265)}
\end{array}
$$

ROW simplification

$$
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\Longrightarrow
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_0\,[1415] = 9265)}
\end{array}
$$

These simplifications depend on two factors

- The equality check procedure
  $\Longrightarrow$ verify that $\texttt{eax}_0 = \texttt{esp}_0 - 16$
- The underlying representation of an array
  $\Longrightarrow$ remember that $\texttt{mem}_1\,[\texttt{esp}_1 + 48] = 1415$

Outline

1. Dedicated data structure
2. Approximated equality check
3. Experimental evaluation
4. Conclusion

Section 1

Dedicated data structure

Standard ROW simplification (using list representation)

Array terms represented as write chains

How to update

Given a write of e at index i

- Add a new head node (i, e)

How to simplify ROW

Given a read at index j

- Let (i, e) be the head
- Is $i = j$ *valid*? If so return e
- Is $i \neq j$ *valid*? If so recurse
- Else stop

Standard ROW simplification (using list representation)

Array terms represented as write chains

How to update

Given a write of e at index i

- Add a new head node (i, e)

How to simplify ROW

Given a read at index j

- Let (i, e) be the head
- Is $i = j$ *valid*? If so return e
- Is $i \neq j$ *valid*? If so recurse
- Else stop

Good points

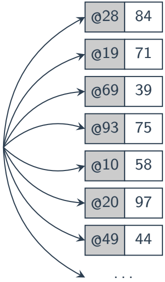
- Generic
- Full ROW simplification

Bad points

- n^2 worst case
- $k \cdot n$ if k-bounded, but then partial simplification
- Behave badly with WOW

Specific case from Symbolic Execution (using map representation)

Common preprocessing in SE when all indices are constants



Bad point

- Extremelly specific

Good points

- $n \cdot \ln n$ worst case
- Full ROW simplification
- Behave well with WOW

So here we are

	list	k-list	map
generic	✓	✓	✗
ROW	✓	✓/✗	✓
WOW	✗	✗	✓
complexity	n^2	$k \cdot n$	$n \cdot \ln n$

We would like the best of both worlds !

So here we are

	list	k-list	map
generic	✓	✓	✗
ROW	✓	✓/✗	✓
WOW	✗	✗	✓
complexity	n^2	$k \cdot n$	$n \cdot \ln n$

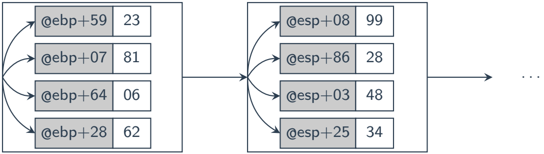
We would like the best of both worlds !

Our proposal : List-Map representation

- Sets of comparable indexes can be packed together
- Allows efficient search during ROW-simplification

$\Longrightarrow$ Improve scalability

Improving scalability : list-map representation

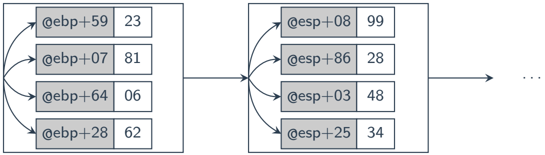


How to update

Given a write of e at index i

- Is i *comparable* with indices of elements in the head ?
- If so add (i, e) in this map
- Else add a new head map containing only (i, e)

Improving scalability : list-map representation



How to update

Given a write of e at index i

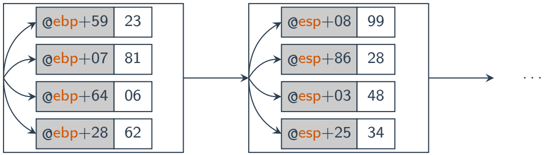
- Is i *comparable* with indices of elements in the head ?
- If so add (i, e) in this map
- Else add a new head map containing only (i, e)

How to simplify ROW

Given a read at index j

- Is j *comparable* with indices of elements in the head ?
- If so, look for (i, e) with $i{=}j$
  - if succeed then return e
  - else recurse on next map
- Else stop

Improving scalability : list-map representation



How to update

Given a write of e at index i

- Is i *comparable* with indices of elements in the head ?
- If so add (i, e) in this map
- Else add a new head map containing only (i, e)

How to simplify ROW

Given a read at index j

- Is j *comparable* with indices of elements in the head ?
- If so, look for (i, e) with $i=j$
  - if succeed then return e
  - else recurse on next map
- Else stop

Section 2

Approximated equality check

Easing (dis)equality reasoning : base normalization

Propagate “variable+constant” terms

- If $y \triangleq z + 1$ then $x \triangleq y + 2 \rightsquigarrow x \triangleq z + 3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

Easing (dis)equality reasoning : base normalization

Propagate "variable+constant" terms

- If $y \triangleq z + 1$ then $x \triangleq y + 2 \rightsquigarrow x \triangleq z + 3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

if $u \triangleq v$	then	$u + k$	$\rightsquigarrow$	$v + k$	alias inlining
if $u \triangleq v + l$	then	$u + k$	$\rightsquigarrow$	$v + (k + l)$	base/offset inlining
		$-(x + k)$	$\rightsquigarrow$	$(-k) - x$	constant negation
		$(x + k) + l$	$\rightsquigarrow$	$x + (k + l)$	constant packing
		$(x + k) + y$	$\rightsquigarrow$	$(x + y) + k$	constant lifting
		$(x + k) + (y + l)$	$\rightsquigarrow$	$(x + y) + (k + l)$	base/offset addition
		$(x + k) - (y + l)$	$\rightsquigarrow$	$(x - y) + (k - l)$	base/offset subtraction
			$\ldots$		

Easing (dis)equality reasoning : base normalization

Propagate "variable+constant" terms

- If $y \triangleq z + 1$ then $x \triangleq y + 2 \rightsquigarrow x \triangleq z + 3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

$$\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
 & & \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\quad \Longrightarrow \quad
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
 & & \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}$$

Easing (dis)equality reasoning : base normalization

Propagate "variable+constant" terms

- If $y \triangleq z + 1$ then $x \triangleq y + 2 \rightsquigarrow x \triangleq z + 3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

$$
\begin{array}{lll}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\quad \Longrightarrow \quad
\begin{array}{lll}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_0 - 64 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
$$

Easing (dis)equality reasoning : base normalization

Propagate "variable+constant" terms

- If $y \triangleq z+1$ then $x \triangleq y+2 \rightsquigarrow x \triangleq z+3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

$$\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\quad\Longrightarrow\quad
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_0 - 16] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}$$

Easing (dis)equality reasoning : base normalization

Propagate "variable+constant" terms

- If $y \triangleq z+1$ then $x \triangleq y+2 \rightsquigarrow x \triangleq z+3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

$$
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\Longrightarrow
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{eax}_0 & \triangleq & 1415 \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
$$

Easing (dis)equality reasoning : base normalization

Propagate "variable+constant" terms

- If $y \triangleq z + 1$ then $x \triangleq y + 2 \rightsquigarrow x \triangleq z + 3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

$$\begin{array}{lcl} \texttt{esp}_0 & : & \texttt{BitVec16} \\ \texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\ \\ \multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\ \texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\ \texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\ \texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\ \multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)} \end{array}$$

$\Longrightarrow$

$$\begin{array}{lcl} \texttt{esp}_0 & : & \texttt{BitVec16} \\ \texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\ \\ \multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\ \texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\ \texttt{eax}_0 & \triangleq & 1415 \\ \multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[1415] = 9265)} \end{array}$$

Easing (dis)equality reasoning : base normalization

Propagate “variable+constant” terms

- If $y \triangleq z + 1$ then $x \triangleq y + 2 \rightsquigarrow x \triangleq z + 3$
- Together with associativity, commutativity...

$\Longrightarrow$ Reduce the number of bases

$$
\begin{array}{lcl}
\mathtt{esp}_0 & : & \mathtt{BitVec16} \\
\mathtt{mem}_0 & : & \mathtt{Array\ BitVec16\ BitVec16} \\
\\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{esp}_0 > 61440)} \\
\mathtt{mem}_1 & \triangleq & \mathtt{mem}_0\,[\mathtt{esp}_0 - 16] \leftarrow 1415 \\
\mathtt{esp}_1 & \triangleq & \mathtt{esp}_0 - 64 \\
\mathtt{eax}_0 & \triangleq & \mathtt{mem}_1\,[\mathtt{esp}_1 + 48] \\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{mem}_1\,[\mathtt{eax}_0] = 9265)}
\end{array}
\Longrightarrow
\begin{array}{lcl}
\mathtt{esp}_0 & : & \mathtt{BitVec16} \\
\mathtt{mem}_0 & : & \mathtt{Array\ BitVec16\ BitVec16} \\
\\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{esp}_0 > 61440)} \\
\mathtt{mem}_1 & \triangleq & \mathtt{mem}_0\,[\mathtt{esp}_0 - 16] \leftarrow 1415 \\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{mem}_1\,[1415] = 9265)}
\end{array}
$$

How to get ride of this declaration ?

Easing (dis)equality reasoning : abstract domains

Associate to every indices i an abstract domain $i^\sharp$

- If $i^\sharp \sqcap j^\sharp = \bot$ then $(a\,[i] \leftarrow e)\,[j] = a\,[j]$
- Integrated in the list-map representation

$\Longrightarrow$ Prove disequality between different bases

Easing (dis)equality reasoning : abstract domains

Associate to every indices i an abstract domain $i^\sharp$

- If $i^\sharp \sqcap j^\sharp = \bot$ then $(a[i] \leftarrow e)[j] = a[j]$
- Integrated in the list-map representation

$\Longrightarrow$ Prove disequality between different bases

$$
\begin{aligned}
c^\sharp &= [c, c] && \text{for any constant } c \\
v^\sharp &= [m_i, M_j] && \text{if } i \le v \le j \\
\left(\text{extract}_{l,h}\, i\right)^\sharp &= [0, 2^{h-l+1} - 1] && \text{if } (M_i \gg l) - (m_i \gg l) \ge 2^{h-l+1} \\
&= [\text{extract}_{l,h}(m_i), \text{extract}_{l,h}(M_i)] && \text{if } \text{extract}_{l,h}(M_i) \ge \text{extract}_{l,h}(m_i) \\
&= [0, \text{extract}_{l,h}(M_i)] && \text{otherwise} \\
&\quad \sqcup [\text{extract}_{l,h}(m_i), 2^{h-l+1} - 1] && \\
(i + j)^\sharp &= [m_i + m_j, M_i + M_j] && \text{if } M_i + M_j < 2^N \\
&= [m_i + m_j - 2^N, M_i + M_j - 2^N] && \text{if } m_i + m_j \ge 2^N \\
&= [m_i + m_j - 2^N, 2^N - 1] && \text{otherwise} \\
&\quad \sqcup [0, M_i + M_j - 2^N] &&
\end{aligned}
$$

Easing (dis)equality reasoning : abstract domains

Associate to every indices i an abstract domain $i^\sharp$

- If $i^\sharp \sqcap j^\sharp = \bot$ then $(a\,[i] \leftarrow e)\,[j] = a\,[j]$
- Integrated in the list-map representation

$\Longrightarrow$ Prove disequality between different bases

$$
\begin{array}{lll}
\mathtt{esp}_0 & : & \mathtt{BitVec16} \\
\mathtt{mem}_0 & : & \mathtt{Array\ BitVec16\ BitVec16} \\
\\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{esp}_0 > 61440)} \\
\mathtt{mem}_1 & \triangleq & \mathtt{mem}_0\,[\mathtt{esp}_0 - 16] \leftarrow 1415 \\
\mathtt{esp}_1 & \triangleq & \mathtt{esp}_0 - 64 \\
\mathtt{eax}_0 & \triangleq & \mathtt{mem}_1\,[\mathtt{esp}_1 + 48] \\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{mem}_1\,[\mathtt{eax}_0] = 9265)}
\end{array}
\Longrightarrow
\begin{array}{lll}
\mathtt{esp}_0 & : & \mathtt{BitVec16} \\
\mathtt{mem}_0 & : & \mathtt{Array\ BitVec16\ BitVec16} \\
\\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{esp}_0 > 61440)} \\
\mathtt{mem}_1 & \triangleq & \mathtt{mem}_0\,[\mathtt{esp}_0 - 16] \leftarrow 1415 \\
\multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{mem}_1\,[1415] = 9265)} \\
\\
\multicolumn{3}{l}{\mathtt{esp}_0^\sharp = [61441, 615535]}
\end{array}
$$

Easing (dis)equality reasoning : abstract domains

Associate to every indices i an abstract domain $i^\sharp$

- If $i^\sharp \sqcap j^\sharp = \bot$ then $(a\,[i] \leftarrow e)\,[j] = a\,[j]$
- Integrated in the list-map representation

$\Longrightarrow$ Prove disequality between different bases

$$
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
 & & \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\quad\Longrightarrow\quad
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
 & & \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[1415] = 9265)} \\
 & & \\
\multicolumn{3}{l}{(\texttt{esp}_0 - 16)^\sharp = [61425, 615519]}
\end{array}
$$

Easing (dis)equality reasoning : abstract domains

Associate to every indices i an abstract domain $i^\sharp$

- If $i^\sharp \sqcap j^\sharp = \bot$ then $(a\,[i] \leftarrow e)\,[j] = a\,[j]$
- Integrated in the list-map representation

$\Longrightarrow$ Prove disequality between different bases

$$\begin{array}{lcl} \mathtt{esp}_0 & : & \mathtt{BitVec16} \\ \mathtt{mem}_0 & : & \mathtt{Array\ BitVec16\ BitVec16} \\ \\ \multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{esp}_0 > 61440)} \\ \mathtt{mem}_1 & \triangleq & \mathtt{mem}_0\,[\mathtt{esp}_0 - 16] \leftarrow 1415 \\ \mathtt{esp}_1 & \triangleq & \mathtt{esp}_0 - 64 \\ \mathtt{eax}_0 & \triangleq & \mathtt{mem}_1\,[\mathtt{esp}_1 + 48] \\ \multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{mem}_1\,[\mathtt{eax}_0] = 9265)} \end{array} \quad \Longrightarrow \quad \begin{array}{lcl} \mathtt{esp}_0 & : & \mathtt{BitVec16} \\ \mathtt{mem}_0 & : & \mathtt{Array\ BitVec16\ BitVec16} \\ \\ \multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{esp}_0 > 61440)} \\ \mathtt{mem}_1 & \triangleq & \mathtt{mem}_0\,[\mathtt{esp}_0 - 16] \leftarrow 1415 \\ \multicolumn{3}{l}{\mathtt{assert}\,(\mathtt{mem}_1\,[1415] = 9265)} \\ \\ \multicolumn{3}{l}{1415^\sharp = [1415, 1415]} \end{array}$$

Easing (dis)equality reasoning : abstract domains

Associate to every indices i an abstract domain $i^\sharp$

- If $i^\sharp \sqcap j^\sharp = \bot$ then $(a\,[i] \leftarrow e)\,[j] = a\,[j]$
- Integrated in the list-map representation

$\Longrightarrow$ Prove disequality between different bases

$$
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\quad\Longrightarrow\quad
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_0\,[1415] = 9265)} \\
\\
\multicolumn{3}{l}{(\texttt{esp}_0 - 16)^\sharp \sqcap 1415^\sharp = \bot}
\end{array}
$$

Easing (dis)equality reasoning : abstract domains

Associate to every indices i an abstract domain $i^\sharp$

- If $i^\sharp \sqcap j^\sharp = \bot$ then $(a\,[i] \leftarrow e)\,[j] = a\,[j]$
- Integrated in the list-map representation

$\Longrightarrow$ Prove disequality between different bases

$$
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\texttt{mem}_1 & \triangleq & \texttt{mem}_0\,[\texttt{esp}_0 - 16] \leftarrow 1415 \\
\texttt{esp}_1 & \triangleq & \texttt{esp}_0 - 64 \\
\texttt{eax}_0 & \triangleq & \texttt{mem}_1\,[\texttt{esp}_1 + 48] \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_1\,[\texttt{eax}_0] = 9265)}
\end{array}
\quad \Longrightarrow \quad
\begin{array}{lcl}
\texttt{esp}_0 & : & \texttt{BitVec16} \\
\texttt{mem}_0 & : & \texttt{Array BitVec16 BitVec16} \\
\\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{esp}_0 > 61440)} \\
\multicolumn{3}{l}{\texttt{assert}\,(\texttt{mem}_0\,[1415] = 9265)}
\end{array}
$$

Section 3

Experimental evaluation

Experimental setup

FAS implemented as

- preprocessor for SMT formulas
- QF_ABV theory

The implementation comprises

- 6,300 lines of OCaml
- integrated into the TFML formula preprocessing engine
- part of the BINSEC Symbolic Execution (SE) tool

Experiments are carried out on

- Intel(R) Xeon(R) CPU E5-2660 v3 @ 2,60GHz
- three of the best SMT solvers for the QF_ABV theory
  Boolector, Yices and Z3

Impact of the simplification : medium-size formulas

- 6,590 x 3 medium-size formulas from static SE
- TIMEOUT = 1,000 seconds

		simpl. time	#TIMEOUT and resolution time						#ROW
			Boolector		Yices		Z3		
concrete	default	61	0	163	2	69	0	872	866,155
	FAS	85	0	94	2	68	0	244	1,318
	FAS-itv	111	0	94	2	68	0	224	1,318
interval	default	65	0	2,584	2	465	31	155,992	866,155
	FAS	99	0	2,245	2	487	25	126,806	531,654
	FAS-itv	118	0	755	2	140	14	37,269	205,733
symbolic	default	61	0	6,173	3	1,961	65	305,619	866,155
	FAS	91	0	6,117	3	1,965	66	158,635	531,654
	FAS-itv	111	0	4,767	2	1,108	43	80,569	295,333

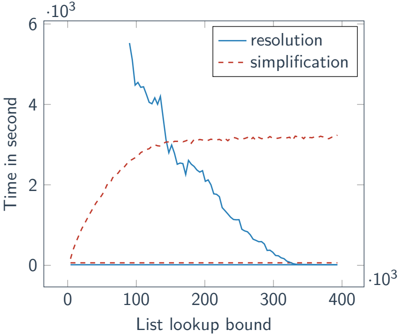
Impact of the simplification : very large formulas

- 29 x 3 very large formulas from dynamic SE
- TIMEOUT = 1,000 seconds

		simpl. time	#TIMEOUT and resolution time						#ROW
			Boolector		Yices		Z3		
concrete	default	44	10	159	4	1,098	26	3.33	1,120,798
	FAS-list	1,108	8	845	4	198	10	918	456,915
	FAS	196	8	820	4	196	10	922	456,915
	FAS-itv	210	4	654	1	12	4	1,120	0
interval	default	44	12	131	12	596	27	0.19	1,120,798
	FAS-list	222	12	129	12	595	26	236	657,594
	FAS	231	12	129	12	597	26	291	657,594
	FAS-itv	237	12	58	12	28	19	81	651,449
symbolic	default	40	12	1,522	12	1,961	27	0.13	1,120,798
	FAS-list	187	11	1,199	12	2,018	26	486	657,594
	FAS	194	11	1,212	12	2,081	26	481	657,594
	FAS-itv	200	11	1,205	12	2,063	26	416	657,594

Focus on specific case : the ASPack example

- Huge formula obtained from the ASPack packing tool
- 293 000 ROWs
- 24 hours of resolution !



Using FAS

- #ROW reduced to 2 467
- 14 sec for resolution
- 61 sec for preprocessing

Using list representation

- Same result with a bound of 385 024 and beyond...
- ...but 53 min preprocessing

Section 4

Conclusion

Conclusion and future work

Software verification

- More and more rely on decision procedures

Array theory

- Useful for modelling memory or data structures
- Bottleneck of resolution for large formulas (BMC, SE)

Our proposal

- Efficient simplification for array theory

$\Longrightarrow$ Improve existing solvers

Future work

- Deeper integration inside a dedicated array solver
- Adding more expressive domain reasoning

available online at http://benjamin.farinier.org/lpar2018/