Static Analysis of Array Contents

Mathias Péron, Nicolas Halbwachs

Verimag/CNRS
Grenoble

Considering arrays in static analysis

- array bound checking: solved in 90% cases
- array contents:
  - expansion (small known size)
  - summarization
  - symbolic partitioning + summarization

Array summarization in Astrée

- Abstract each array A by a single variable a
- Interpretation: $\psi(a) \Leftrightarrow \forall \ell = 1..n, \psi(A[\ell])$
- Each assignment A[i] := exp is interpreted as a weak assignment to a:

$$a \sqcup= exp \quad \equiv \quad a := exp \; [] \; a := a$$

Problems:

- no information gained from conditionals
- weak assignment can only lose information
- information about the initial content of arrays must be obtained by other means

Symbolic partitioning + summarization (1/3)

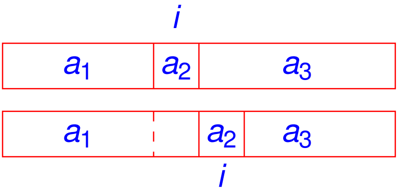
[Gopan, Reps, Sagiv - Popl05]

- partition each array into symbolic slices, e.g.,

$$A_1 = A[1..i-1],\ A_2 = A[i],\ A_3 = A[i+1..n]$$

- abstract each slice A_p by a single variable a_p
- interpret $A[i]$:= exp as a strong assignment a_2 := exp
- interpret an index incrementation i++ as

$$a_1 \sqcup = a_2\ ;\ a_2 := a_3$$



Interpretation: $\psi(a_p) \Leftrightarrow \forall \ell \in I_p, \psi(A[\ell])$

Symbolic partitioning + summarization (2/3)

```
m:= A[1]
for(i:=2; i≤n; i++){
   if (A[i]>m)
      m := A[i];
}
```

Symbolic partitioning + summarization (2/3)

Partition: $I_1 = [1..i-1]$, $I_2 = \{i\}$, $I_3 = [i+1..n]$

```
m:= A[1]
for(i:=2; i≤n; i++){
   if (A[i]>m)
       m := A[i];
}
```

Symbolic partitioning + summarization (2/3)

Partition: $I_1 = [1..i-1],\ I_2 = \{i\},\ I_3 = [i+1..n]$

m:= A[1]	$m := a_1;$
for(i:=2; i≤n; i++){	for (i:=2; i≤ n; i++){
if (A[i]>m)	if ($a_2 > m$)
m := A[i];	$m := a_2;$
	$a_1 \sqcup = a_2; a_2 := a_3$
}	}

Symbolic partitioning + summarization (2/3)

Partition: $I_1 = [1..i-1],\ I_2 = \{i\},\ I_3 = [i+1..n]$

m:= A[1]	$m := a_1;$	$m = a_1$
for(i:=2; i≤n; i++){	for (i:=2; i≤ n; i++){	$2 \leq i \leq n, m \geq a_1$
if (A[i]>m)	if ($a_2 > m$)	
m := A[i];	$m := a_2;$	$2 \leq i \leq n, m = a_2 \geq a_1$
	$a_1 \sqcup = a_2; a_2 := a_3$	$2 \leq i \leq n, m \geq a_1, m \geq a_2$
}	}	$i = n+1, m \geq a_1$

Symbolic partitioning + summarization (3/3)

- able to discover unary properties about array elements
- unable to discover relations between array elements
- able to check (with TVLA) such relations
    (provided by the user)
  e.g., $\forall \ell = 1..n, A[\ell] = B[\ell]$

Symbolic partitioning + summarization (3/3)

- able to discover unary properties about array elements
- unable to discover relations between array elements
- able to check (with TVLA) such relations
  (provided by the user)
  e.g., $\forall \ell = 1..n, A[\ell] = B[\ell]$

This work: generalization to discover simple relations:

$$\forall \ell \in I, \psi(A1[\ell + k_1], \ldots, Am[\ell + k_m])$$

for “simple programs” (pointwise relations between array slices, inspired by Lustre-V4)

Simple programs

- one-dimensional arrays
- simple traversal: i:=e1; while(cond)$\{\ldots;$ i++$\}$
- simple array access: A[i] := exp(B[i+k])

Simple programs

- one-dimensional arrays
- simple traversal: i:=e1; while(cond){. . . ; i++}
- simple array access: A[i] := exp(B[i+k])

Examples:

```
for i:=1 to n do
   A[i] := B[i];
end
```

```
m:= A[1];
for i:=2 to n do
   if m < A[i]
      then m := A[i]
   endif
end
```

```
for i:= 2 to n do
   x:= A[i]; j:=i-1;
   while j≥ 1 and A[j]>x do
      A[j+1]:=A[j]; j:=j-1
   end
   A[j+1]:= x
end
```

$\mathscr{I}$ set of index variables, $\mathscr{I}' = \mathscr{I} \cup \{\ell\}$
$\mathscr{C}$ set of content variables, $\mathscr{A}$ set of arrays.

Partitions and Properties (1/3)

Two basic lattices:

- L_N ($\ni \varphi$): properties of indexes (at least DBMs)
- L_C ($\ni \psi$): properties of contents

Partition: $\{\varphi_p\}_{p \in P} \subset L_N(\mathscr{I}')$ such that

$$\bigvee_{p \in P} \varphi_p = \top_N \quad \text{and} \quad p \neq p' \Rightarrow \varphi_p \sqcap \varphi_{p'} = \bot_N$$

ex: $\varphi_1 = (1 \leq \ell < i), \varphi_2 = (1 \leq \ell = i \leq n), \varphi_3 = (i < \ell \leq n)$

Partitions and Properties (2/3)

Slice variables: $\{a_p^z\}$, $a \leftrightarrow A \in \mathscr{A}, p \in P, z \in \mathbb{Z}$

$$a_p^z \text{ represents the slice } A[\ell + z \mid \varphi_p(\ell)]$$

Properties: given a partition $\{\varphi_p\}_{p \in P}$,

$$\Psi = (\varphi, (\psi_p)_{p \in P})$$

$$\gamma(\Psi) = \{ \quad (\mathscr{I}, \mathscr{C}, \mathscr{A}) \text{ such that} \\ \varphi(\mathscr{I}), \\ \forall p \in P, \forall \ell, \quad \varphi_p(\mathscr{I} \cup \{\ell\}) \Rightarrow \psi_p[A[\ell + z]/a_p^z]\}$$

Partitions and Properties (3/3)

Examples:

	φ_1	φ_2	φ_3
	$(1 \leq \ell < i)$	$(\ell = i \leq n)$	$(i < \ell \leq n)$
$\varphi = (i = n+1)$	$\psi_1 = (a_1^0 = b_1^0)$	$\psi_2 = \text{False}$	$\psi_3 = \text{False}$

	φ_1	φ_2	φ_3	φ_4
	$(\ell = 1)$	$(2 \leq \ell < i)$	$(\ell = i)$	$(i < \ell \leq n)$
φ	ψ_1	ψ_2	ψ_3	ψ_4
$(2 \leq i \leq n)$	$(a_1^0 \leq a_1^1)$	$(a_2^0 \geq a_2^{-1})$	$(a_3^0 = x)$	True

Partitions and Properties (3/3)

Examples:

$$\begin{array}{c|c|c|c}
 & \varphi_1 & \varphi_2 & \varphi_3 \\
 & (1 \leq \ell < i) & (\ell = i \leq n) & (i < \ell \leq n) \\
\varphi = (i = n+1) & \psi_1 = (a_1^0 = b_1^0) & \psi_2 = \text{False} & \psi_3 = \text{False}
\end{array}$$

$$\begin{array}{c|c|c|c|c}
 & \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 \\
 & (\ell = 1) & (2 \leq \ell < i) & (\ell = i) & (i < \ell \leq n) \\
\varphi & \psi_1 & \psi_2 & \psi_3 & \psi_4 \\
(2 \leq i \leq n) & (a_1^0 \leq a_1^1) & (a_2^0 \geq a_2^{-1}) & (a_3^0 = x) & \text{True}
\end{array}$$

Remark: if $\varphi \Rightarrow \neg(\exists \ell \varphi_p)$, ψ_p can be normalized to False:

$$\forall \ell,\ \ell \in \emptyset \ \Rightarrow\ \text{False}(\ell)$$

Example of analysis

```
i := 1;
while (i ≤ n){
    A[i] := B[i];
    i := i+1;
}
```

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;				
while (i $\leq$ n){				
A[i] := B[i];				
i := i+1;				
}				

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){				
A[i] := B[i];				
i := i+1;				
}				

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){	$(i = 1 \leq n)$	False	True	True
A[i] := B[i];				
i := i+1;				
}				

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){	$(i = 1 \leq n)$	False	True	True
A[i] := B[i];	$(i = 1 \leq n)$	False	$(a_2 = b_2)$	True
i := i+1;				
}				

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){	$(i = 1 \leq n)$	False	True	True
A[i] := B[i];	$(i = 1 \leq n)$	False	$(a_2 = b_2)$	True
i := i+1;	$(i = 2 \leq n+1)$	$(a_1 = b_1)$	True	True
}				

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){	$(1 \leq i \leq n)$	$(a_1 = b_1)$	True	True
A[i] := B[i];				
i := i+1;	$(i = 2 \leq n+1)$	$(a_1 = b_1)$	True	True
}				

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){	$(1 \leq i \leq n)$	$(a_1 = b_1)$	True	True
A[i] := B[i];	$(1 \leq i \leq n)$	$(a_1 = b_1)$	$(a_2 = b_2)$	True
i := i+1;				
}				

Example of analysis

Partition: $\varphi_1 = (1 \le \ell < i)$, $\varphi_2 = (1 \le \ell = i \le n)$, $\varphi_3 = (i < \ell \le n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\le$ n){	$(1 \le i \le n)$	$(a_1 = b_1)$	True	True
A[i] := B[i];	$(1 \le i \le n)$	$(a_1 = b_1)$	$(a_2 = b_2)$	True
i := i+1;	$(2 \le i \le n+1)$	$(a_1 = b_1)$	True	True
}				

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){	$(1 \leq i \leq n)$	$(a_1 = b_1)$	True	True
A[i] := B[i];	$(1 \leq i \leq n)$	$(a_1 = b_1)$	$(a_2 = b_2)$	True
i := i+1;	$(2 \leq i \leq n+1)$	$(a_1 = b_1)$	True	True
}	$(i = n+1)$	$(a_1 = b_1)$	False	False

Example of analysis

Partition: $\varphi_1 = (1 \leq \ell < i)$, $\varphi_2 = (1 \leq \ell = i \leq n)$, $\varphi_3 = (i < \ell \leq n)$

	φ	ψ_1	ψ_2	ψ_3
i := 1;	$(i = 1)$	False	True	True
while (i $\leq$ n){	$(1 \leq i \leq n)$	$(a_1 = b_1)$	True	True
A[i] := B[i];	$(1 \leq i \leq n)$	$(a_1 = b_1)$	$(a_2 = b_2)$	True
i := i+1;	$(2 \leq i \leq n+1)$	$(a_1 = b_1)$	True	True
}	$(i = n+1)$	$(a_1 = b_1)$	False	False

$\forall \ell, 1 \leq \ell \leq n, A[\ell] = B[\ell]$

Some results

- Insertion sort

$$\forall \ell, (2 \leq \ell \leq n) \Rightarrow (A[\ell - 1] \leq A[\ell]$$

- Find (QuickSort segmentation)

$$\begin{aligned}
&(i = n \wedge \forall \ell, (1 \leq \ell \leq n-1) \Rightarrow (A[\ell] \leq A[i]) \\
\vee \quad & i = 1 \wedge \forall \ell, (2 \leq \ell \leq n) \Rightarrow (A[i] < A[\ell]) \\
\vee \quad & 1 < i < n \wedge \forall \ell, \ (1 \leq \ell \leq i-1) \Rightarrow (A[\ell] \leq A[i]) \\
& \qquad\qquad\qquad\qquad \wedge (i+1 \leq \ell \leq n) \Rightarrow (A[i] < A[\ell])
\end{aligned}$$

Future work

- improve the implementation

Future work

- improve the implementation
- more general programs (”for” loops with steps, recursivity...)

Future work

- improve the implementation
- more general programs (”for” loops with steps, recursivity...)
- more general properties (non convex slices)

Future work

- improve the implementation
- more general programs (”for” loops with steps, recursivity...)
- more general properties (non convex slices)
- multi-dimensional arrays?

Future work

- improve the implementation
- more general programs ("for" loops with steps, recursivity...)
- more general properties (non convex slices)
- multi-dimensional arrays?
- generalization to function properties?

Future work

- improve the implementation
- more general programs ("for" loops with steps, recursivity...)
- more general properties (non convex slices)
- multi-dimensional arrays?
- generalization to function properties?
- properties about (multi-)sets of array values