Controlling Polyvariance for Specialization-Based Verification

Fabio Fioravanti (Univ. D’Annunzio, Pescara, Italy),
Alberto Pettorossi (Univ. Tor Vergata, Rome, Italy),
Maurizio Proietti (IASI-CNR, Rome, Italy),
Valerio Senni (Univ. Tor Vergata, Rome, Italy)
Verification via Reachability

Forward Reachability

Initial States \( t \) \( t^2 \) \( t^\omega \) \( \ldots \) Unsafe States

= \( \emptyset \) safety
\( \neq \emptyset \) unsafety

Backward Reachability

Initial States \( t^-\omega \) \( t^{-2} \) \( t^{-1} \) Unsafe States

= \( \emptyset \) safety
\( \neq \emptyset \) unsafety
**Backward Reachability as a Constraint Logic Program**

Bw:

(I’s)  unsafe ← init$_1$(X) $\land$ bwReach(X)

(T’s)  bwReach(X) ← t$_1$(X,X’) $\land$ bwReach(X’)

(U’s)  bwReach(X) ← u$_1$(X)

Theorem:

The system is safe iff unsafe $\not\in$ M(Bw) $\approx$ (S$_{Bw}$)$^\omega$

A $\emptyset$  A ← c  with c$\emptyset$ satisf.
An Example of System Verification

\[
\begin{align*}
\text{init}(\langle X_1, X_2 \rangle) : & \quad X_1 \geq 1 \land X_2 = 0 \\
\text{t}(\langle X_1, X_2 \rangle, \langle X'_1, X'_2 \rangle) : & \quad X'_1 = X_1 + X_2 \land X'_2 = X_2 + 1 \\
\text{u}(\langle X_1, X_2 \rangle) : & \quad X_2 > X_1
\end{align*}
\]

Bw:

1. unsafe \leftarrow X_1 \geq 1 \land X_2 = 0 \land \text{bwReach}(X_1, X_2)
2. \text{bwReach}(X_1, X_2) \leftarrow X'_1 = X_1 + X_2 \land X'_2 = X_2 + 1 \land \text{bwReach}(X'_1, X'_2)
3. \text{bwReach}(X_1, X_2) \leftarrow X_2 > X_1

Unfortunately, the computation of M(Bw) does not terminate.

Verification via Specialization:

(A) \quad Bw \quad \rightarrow \quad \text{SpBw}

(B) \quad \text{unsafe} \notin \quad M(\text{SpBw})
Specialization via Unfold/Definition/Fold

def-intro:
4. \( \text{new1}(X_1, X_2) \leftarrow X_1 \geq 1 \land X_2 = 0 \land \text{bwReach}(X_1, X_2) \)  

fold:
1f. unsafe \( \leftarrow X_1 \geq 1 \land X_2 = 0 \land \text{new1}(X_1, X_2) \)

unfold:
4u. \( \text{new1}(X_1, X_2) \leftarrow X_1 \geq 1 \land X_2 = 0 \land X'_1 = X_1 \land X'_2 = 1 \land \text{bwReach}(X'_1, X'_2) \)  

def-intro:
\( \text{newp}(X'_1, X'_2) \leftarrow X'_1 \geq 1 \land X'_2 = 1 \land \text{bwReach}(X'_1, X'_2) \)  

fold: ...
unfold: ...

def-intro:
\( \text{newq}(X''_1, X''_2) \leftarrow X''_1 \geq 1 \land X''_2 = 2 \land \text{bwReach}(X''_1, X''_2) \)  

:  

:  

:  

Nontermination of specialization
Need for Generalization

def-intro:
5. $\text{new2}(x_1, x_2) \leftarrow x_1 \geq 1 \land x_2 \geq 0 \land \text{bwReach}(x_1, x_2)$ (generalization)
4uf. $\text{new1}(x_1, x_2) \leftarrow x_1 \geq 1 \land x_2 = 0 \land x'_1 \geq x_1 \land x'_2 = 1 \land \text{new2}(x'_1, x'_2)$

From 5 by unfold-fold:
6. $\text{new2}(x_1, x_2) \leftarrow x_1 \geq 1 \land x_2 \geq 0 \land x'_1 = x_1 + x_2 \land x'_2 = x_2 + 1 \land \text{new2}(x'_1, x'_2)$
7. $\text{new2}(x_1, x_2) \leftarrow x_1 \geq 1 \land x_2 > x_1$

SpBw: 1f, 4uf, 6, 7.

- Specialization has terminated (due to generalization).
- The computation of $M(\text{SpBw})$ terminates:

\[
\begin{align*}
\text{unsafe} & \notin M(\text{SpBw}) \\
\text{new1}(x_1, x_2) & \leftarrow \text{false} \\
\text{new2}(x_1, x_2) & \leftarrow x_1 \geq 1 \land x_2 > 1
\end{align*}
\]
new2 is *more general* than new1: use new2, instead of new1.

SpBw1:
1f'. unsafe $\leftarrow X_1 \geq 1 \land X_2 = 0 \land \text{new2}(X_1,X_2)$
6. $\text{new2}(X_1,X_2) \leftarrow X_1 \geq 1 \land X_2 \geq 0 \land X'_1 = X_1 + X_2 \land X'_2 = X_2 + 1 \land \text{new2}(X_1,X_2)$
7. $\text{new2}(X_1,X_2) \leftarrow X_1 \geq 1 \land X_2 > X_1$

SpBw1: 1f, 6, 7.

- Fold “immediately”: use of new1 and new2.
  More polyvariance (SpBw).
- Fold at the end “with a maximally general definition”: use of new2 only.
  Less polyvariance (SpBw1).

Polyvariance depends on generalization and folding and
affects the specialization time and the size of the specialized
program (and thus, the computation of the $M(\text{SpBw})$).
Constructing the Definition Tree: DefsTree

Initialization:

- a generic node $D$:
- Unfold using $T$’s and $U$’s:
- Partition of clauses into blocks:
- Generalize:

- Stop if node $D$ occurs earlier in DefsTree.
DefsWithTree for Our Verification!

Initialization:

D1: 4. new1(X1, X2) ← X1 ≥ 1 ∧ X2 = 0 ∧ bwReach(X1, X2)

D1

{1}

T

{4u}

D2: 5. new2(X1, X2) ← X1 ≥ 1 ∧ X2 ≥ 0 ∧ bwReach(X1, X2)

Unfold:

4u. new1(X1, X2) ← X1 ≥ 1 ∧ X2 = 0 ∧ X’1 = X1 ∧ X’2 = 1 ∧ bwReach(X’1, X’2)

Generalize (ch+widen):

5. new2(X1, X2) ← X1 ≥ 1 ∧ X2 ≥ 0 ∧ bwReach(X1, X2)
Another generalization operator: (Convex-Hull and) WidenSum.
It takes into account the coefficients of the variables (in our case: 1).
Inputs: program Bw

Output: program SpBw such that unsafe ∈ M(Bw) iff unsafe ∈ M(SpBw)

Initialization: DefsTree := \{T\rightarrow D_1, ..., T\rightarrow D_k\}

while there exists a definition D in DefsTree which does not occur earlier

do - unfold using T_i’s and U_i’s and derive UnfD;
  - definition introduction:
    Partition(UnfD, \{B_1, ..., B_h\})
    Generalize(D, B_i, DefsTree, G_i) and derive a new DefsTree

od

Fold(DefsTree, SpBw)

Generic Specialization Algorithm
Various Partition Operators

UnfD: clauses $C_1, \ldots, C_m, C_{m+1}, \ldots, C_n$

(constrained facts)

Partition:

1. Singleton: $\{C_1\}, \ldots, \{C_m\}$

($m$ blocks)

2. Finite Domain: clauses $C_i$ and $C_j$ in the same block iff $\text{con}(C_i)|_{X'} \approx_{fd} \text{con}(C_j)|_{X'}$

e.g., $X'_1 = a \land X'_2 = a \approx_{fd} X'_1 = a \land X'_2 = X'_1$

3. All: $\{C_1, \ldots, C_m\}$

(one block)

::
<table>
<thead>
<tr>
<th>Technique by</th>
<th>Partition</th>
<th>Generalization</th>
<th>Folding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cousot-Halbwachs:</td>
<td>Finite-Domain</td>
<td>Widen</td>
<td></td>
</tr>
<tr>
<td>Peralta-Gallagher:</td>
<td>All</td>
<td>Widen</td>
<td>Maximally General</td>
</tr>
<tr>
<td>FPPS (Lopstr 2010):</td>
<td>Singleton</td>
<td>Widen (or WidenSum)</td>
<td>Immediate</td>
</tr>
<tr>
<td>our new1-new2:</td>
<td>Singleton</td>
<td>Widen</td>
<td>Immediate</td>
</tr>
<tr>
<td>our new2:</td>
<td>Singleton</td>
<td>Widen</td>
<td>Maximally General</td>
</tr>
</tbody>
</table>
### Verification of System: Backward Reachability

<table>
<thead>
<tr>
<th></th>
<th>No-Specializat.</th>
<th>All_Widen</th>
<th>Singleton_WidenSum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bakery 4</td>
<td>130</td>
<td>19 (6)</td>
<td>101 (1745)</td>
</tr>
<tr>
<td></td>
<td>Im</td>
<td>MG</td>
<td></td>
</tr>
<tr>
<td>Ticket 2</td>
<td>∞</td>
<td>∞</td>
<td>0.02 (11)</td>
</tr>
<tr>
<td></td>
<td>Im</td>
<td>MG</td>
<td>0.02 (11)</td>
</tr>
<tr>
<td>Futurebus+</td>
<td>15</td>
<td>17 (6)</td>
<td>2.4 (19)</td>
</tr>
<tr>
<td></td>
<td>Im</td>
<td>MG</td>
<td>2.2 (15)</td>
</tr>
<tr>
<td>McCarthy91</td>
<td>∞</td>
<td>4.13 (5)</td>
<td>∞</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MG</td>
<td>4.12 (3)</td>
</tr>
<tr>
<td>29 protocols:</td>
<td>20 verified</td>
<td>MG</td>
<td>21 verified</td>
</tr>
</tbody>
</table>

Times in milliseconds. Number of definitions between parentheses.

∞ means more than 200 seconds.

- Similar results for Forward Reachability.
Conclusions

- A generic specialization algorithm reconstructing various techniques known in the literature (plus new ones), depending on:
  - partition operators (singleton, all, ...)
  - generalization operators (widen, ...)
  - folding procedure (immediate, maximally general)

- Specialization improves precision (i.e., the number of verified properties or systems) but may increment verification time

- Polyvariance control may allow fewer definitions and shorter verification times at the expense of possible loss of precision.
An implementation in SICStus Prolog as a module of the MAP transformation system.

http://map.uniroma2.it/mapweb
Future Work

- Perform more system verifications and check scalability of the approach.
- Use of polyvariance control outside the scope of the verification of reactive systems.
References


