Controlling Polyvariance for Specialization-Based Verification

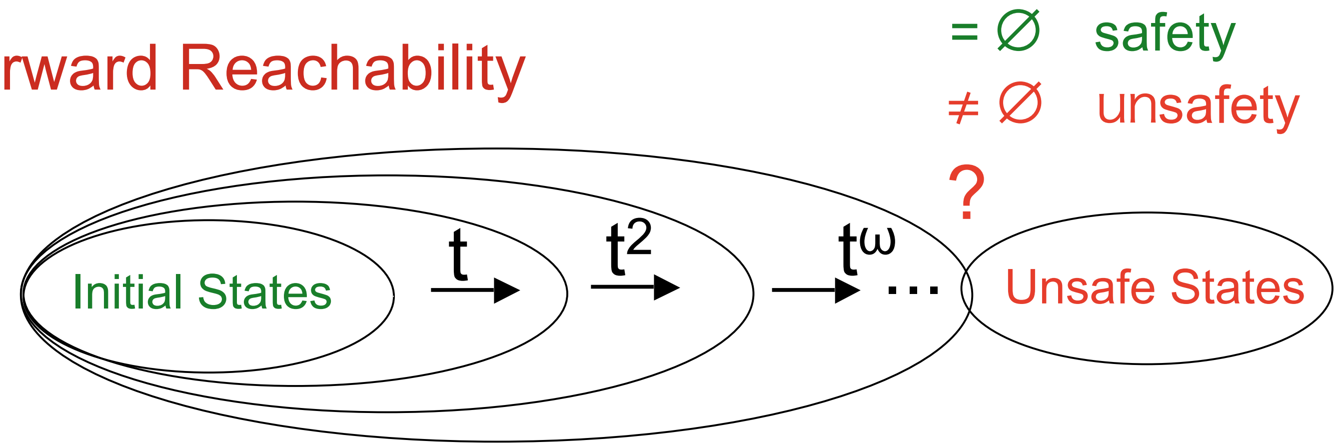
Fabio Fioravanti (Univ. D'Annunzio, Pescara, Italy),
Alberto Pettorossi (Univ. Tor Vergata, Rome, Italy),
Maurizio Proietti (IASI-CNR, Rome, Italy),
Valerio Senni (Univ. Tor Vergata, Rome, Italy)

CILC 2011, Pescara

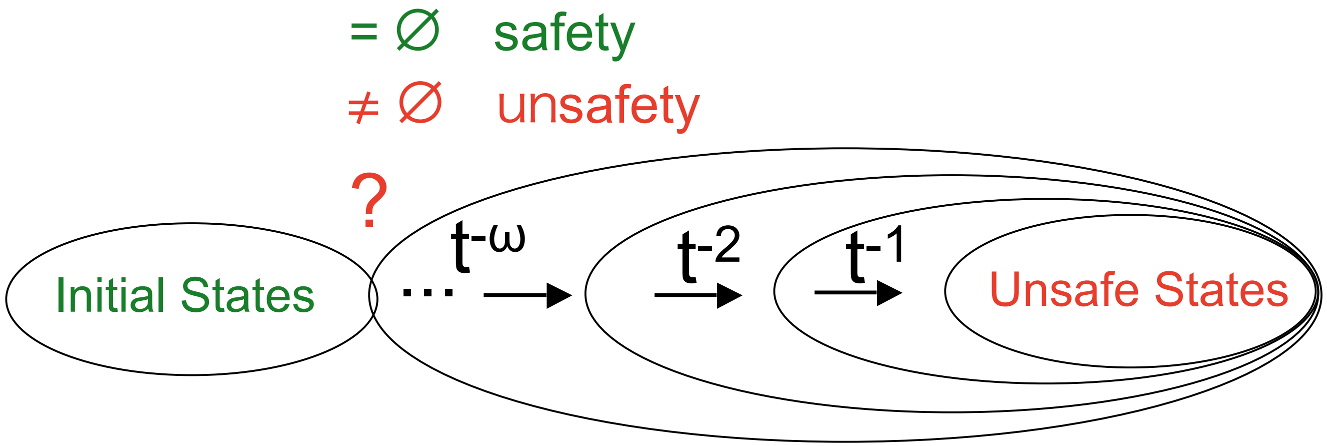
August 31 - September 2, 2011

Verification via Reachability

Forward Reachability



Backward Reachability



Backward Reachability as a Constraint Logic Program

Bw:

(I's) $\quad unsafe \leftarrow init_1(X) \wedge bwReach(X)$

$\vdots$

(T's) $\quad bwReach(X) \leftarrow t_1(X,X') \wedge bwReach(X')$

$\vdots$

(U's) $\quad bwReach(X) \leftarrow u_1(X)$

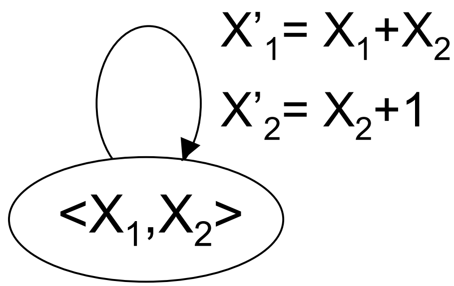
$\vdots$

Theorem:

$\boxed{\text{The system is safe iff unsafe} \notin M(Bw)}$ $\simeq (S_{Bw})\uparrow^{\omega}$

$A\vartheta$ $\qquad A \leftarrow c \quad$ with $c\vartheta$ satisf.

An Example of System Verification



$$\left[\begin{array}{l} \text{init}(<X_1,X_2>):\quad X_1\geq 1 \wedge X_2=0 \\ \text{t}(<X_1,X_2>, <X'_1,X'_2>):\quad X'_1= X_1+X_2 \;\wedge X'_2= X_2+1 \\ \text{u}(<X_1,X_2>):\; X_2>X_1 \end{array}\right.$$

Bw:

1. unsafe $\leftarrow X_1\geq 1 \wedge X_2=0 \wedge$ bwReach(X_1,X_2)
2. bwReach$(X_1,X_2) \leftarrow X'_1= X_1+X_2 \;\wedge X'_2= X_2+1 \wedge$ bwReach(X'_1,X'_2)
3. bwReach$(X_1,X_2) \leftarrow X_2>X_1$

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

Verification via Specialization:

(A) Bw $\longrightarrow$ SpBw

(B) unsafe $\notin$ M(SpBw)

Specialization via Unfold/Definition/Fold

def-intro:

4. $new1(X_1,X_2) \leftarrow \underline{X_1 \geq 1 \wedge X_2=0 \wedge bwReach(X_1,X_2)}$ ■ 0

fold:

1f. $unsafe \leftarrow X_1 \geq 1 \wedge X_2=0 \wedge new1(X_1,X_2)$

unfold:

4u. $new1(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2=0 \wedge \underline{X'_1=X_1 \wedge X'_2=1 \wedge bwReach(X'_1,X'_2)}$

def-intro:

$newp(X'_1,X'_2) \leftarrow \underline{X'_1 \geq 1 \wedge X'_2=1 \wedge bwReach(X'_1,X'_2)}$ ■ 1

fold: ...

unfold: ...

def-intro:

$newq(X''_1,X''_2) \leftarrow \underline{X''_1 \geq 1 \wedge X''_2=2 \wedge bwReach(X''_1,X''_2)}$ ■ 2

⋮ ⋮

- Nontermination of specialization

Need for Generalization

def-intro:

5. $new2(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 \geq 0 \wedge bwReach(X_1,X_2)$ (generalization)

4uf. $new1(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 = 0 \wedge X'_1 \geq X_1 \wedge X'_2 = 1 \wedge new2(X'_1,X'_2)$

From 5 by unfold-fold:

6. $new2(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 \geq 0 \wedge X'_1 = X_1 + X_2 \wedge X'_2 = X_2 + 1 \wedge new2(X'_1,X'_2)$

7. $new2(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 > X_1$

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

- Specialization has terminated (due to generalization).
- The computation of M(SpBw) terminates:

$unsafe \notin M(SpBw)$

$new1(X_1,X_2) \leftarrow false$

$new2(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 > 1$

A Different Specialization

new2 is *more general* than new1: use new2, instead of new1.

SpBw1:

1f′. unsafe $\leftarrow X_1 \geq 1 \wedge X_2 = 0 \wedge$ new2(X_1,X_2)

6. new2$(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 \geq 0 \wedge X'_1 = X_1 + X_2 \wedge X'_2 = X_2 + 1 \wedge$ new2(X_1,X_2)

7. new2$(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 > X_1$

SpBw1: 1f, 6, 7.

- Fold "immediately": use of new1 and new2.

  More polyvariance (SpBw).

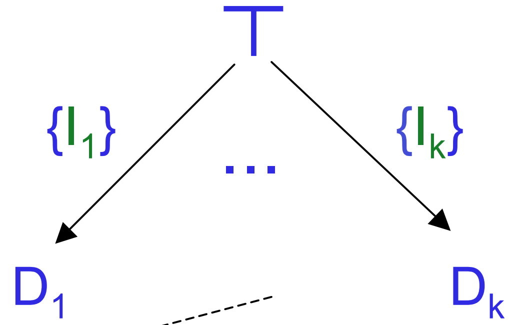
- Fold at the end "with a maximally general definition": use of new2 only.

  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(SpBw)).

Constructing the Definition Tree: DefsTree

Initialization:

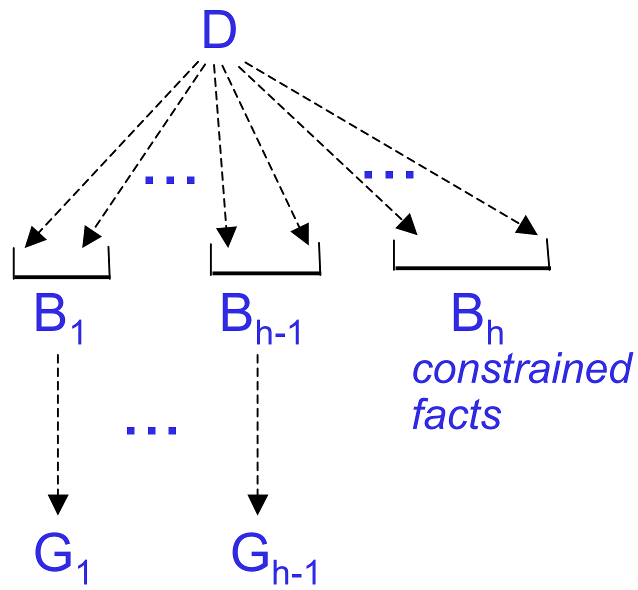


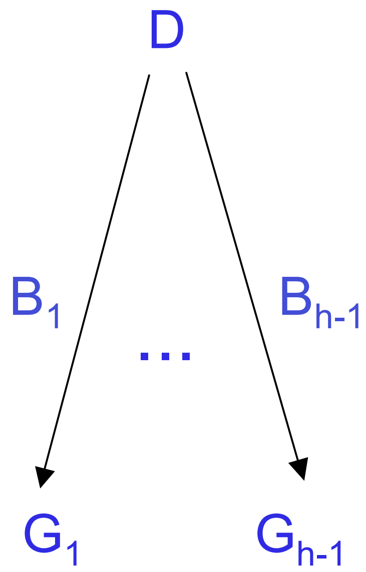
a generic node D:

Unfold using T's and U's:

Partition of clauses into blocks:

Generalize:





- Stop if node D occurs earlier in DefsTree.

DefsTree for Our Verification

Initialization:

$\top$

$\{1\}$

D_1: 4. $new1(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 = 0 \wedge bwReach(X_1,X_2)$

$\{4u\}$

D_2: 5. $new2(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 \geq 0 \wedge bwReach(X_1,X_2)$

D_1

Unfold:

4u. $new1(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 = 0 \wedge X'_1 = X_1 \wedge X'_2 = 1 \wedge bwReach(X'_1,X'_2)$

Generalize (ch+widen):

5. $new2(X_1,X_2) \leftarrow X_1 \geq 1 \wedge X_2 \geq 0 \wedge bwReach(X_1,X_2)$

Generalization: (Convex-Hull and) Widen

previous definition D_1 :

$X_1 \geq 1 \wedge X_2 \leq 0 \wedge \underline{X_2 \geq 0}$

unfold (renaming X'_i / X_i):

$X_1 \geq 1 \wedge \underline{X_2 \leq 1} \wedge X_2 \geq 1$

Convex-Hull

$b_{anc}(X_1, X_2)$:

$X_1 \geq 1 \wedge \underline{X_2 \geq 0} \wedge \underline{X_2 \leq 1}$

previous definition D_1 :

$X_1 \geq 1 \wedge X_2 \leq 0 \wedge X_2 \geq 0$

Widen

$X_1 \geq 1 \wedge X_2 \geq 0$

Another generalization operator: (Convex-Hull and) WidenSum.

It takes into account the coefficients of the variables (in our case: 1).

Generic Specialization Algorithm

Input: program Bw

Output: program SpBw such that unsafe $\in$ M(Bw) iff unsafe $\in$ M(SpBw)

Initialization: DefsTree := $\{T \longrightarrow D_1,\ldots,T \longrightarrow D_k\}$

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:

blocks

Partition(UnfD, $\{B_1,\ldots, B_h\}$) ;

Generalize(D, B_i, DefsTree, G_i) and derive a new DefsTree

od

a generalized definition

Fold(DefsTree, SpBw)

Various Partition Operators

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

(constrained facts)

Partition:

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

   (m blocks)

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

   e.g., $X'_1=a \wedge X'_2=a \simeq_{fd} X'_1=a \wedge X'_2=X'_1$

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

   (one block)

⋮

Reconstructing Known Techniques

Technique by

	Partition	Generalization	Folding
Cousot-Halbwachs:	Finite-Domain	Widen	
Peralta-Gallagher:	All	Widen	Maximally General
FPPS (Lopstr 2010):	Singleton	Widen (or WidenSum)	Immediate
our new1-new2:	Singleton	Widen	Immediate
our new2:	Singleton	Widen	Maximally General

Verification of System: Backward Reachability

Times in milliseconds. Number of definitions between parentheses.

	No-Specializat.		All_Widen	Singleton_WidenSum
Bakery 4	130	Im	19 (6)	101 (1745)
		MG	19 (6)	77 (1172)
Ticket 2	∞	Im	∞	0.02 (11)
		MG	∞	0.02 (11)
Futurebus+	15	Im	17 (6)	2.4 (19)
		MG	15 (3)	2.2 (15)
McCarthy91	∞	Im	4.13 (5)	∞
⋮		MG	4.12 (3)	∞
29 protocols:	20 verified	MG	21 verified	27 verified

∞ 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

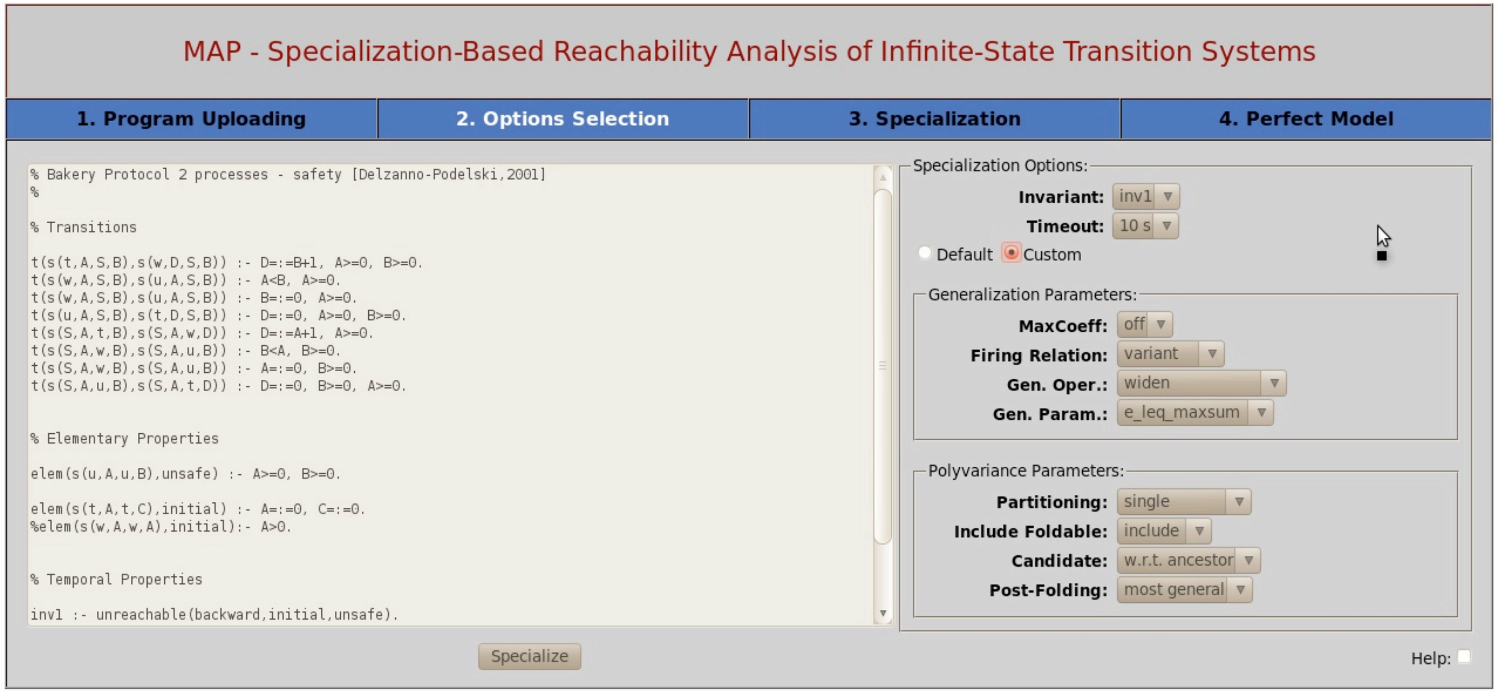
- Polyvariance control may allow fewer definitions and shorter verification times at the expense of possible loss of precision.

Tool

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

E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In Proceedings of the Fifth ACM Symposium on Principles of Programming Languages (POPL'78), 84-96. ACM Press, 1978.

F. Fioravanti, A. Pettorossi, M. Proietti, and V. Senni. *Program specialization for verifying infinite state systems: An experimental evaluation*. In Proceedings of LOPSTR '10, LNCS 6564, 164-183. Springer, 2011.

M. Leuschel, B. Martens, and D. De Schreye. *Controlling generalization and polyvariance in partial deduction of normal logic programs.* ACM Transactions on Programming Languages and Systems, 20(1):208-258, 1998.

J. C. Peralta and J. P. Gallagher. *Convex hull abstractions in specialization of CLP programs*. In Proceedings of LOPSTR '02, LNCS 2664, 90-108. Springer, 2003.