Improving Reachability Analysis of Infinite State Systems by Specialization

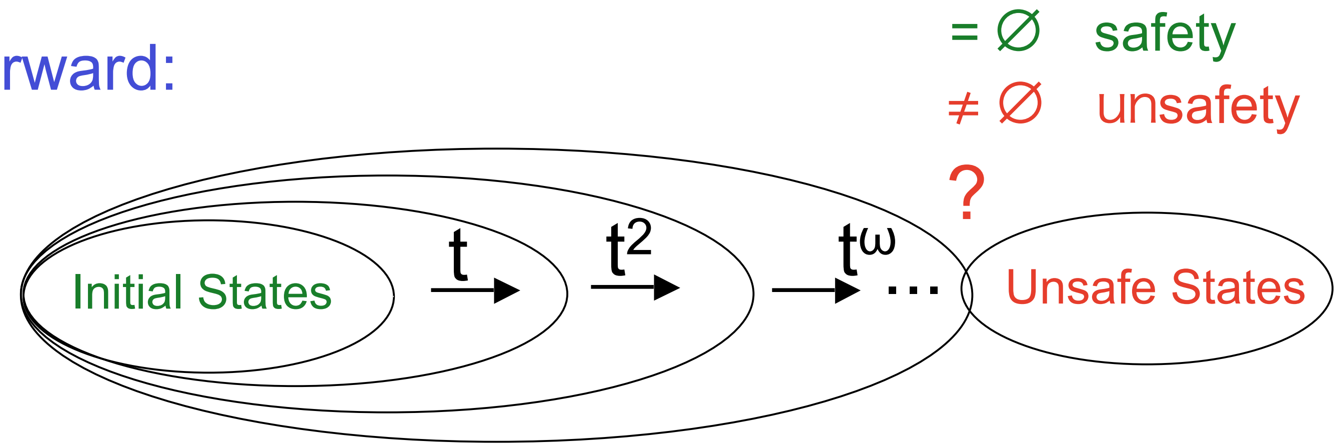
Fabio Fioravanti (University “d’Annunzio”, Pescara, Italy),
Alberto Pettorossi (University “Tor Vergata”, Rome, Italy),
Maurizio Proietti (IASI-CNR, Rome, Italy),
Valerio Senni (University “Tor Vergata”, Rome, Italy)

CS&P 2011, Pułtusk, Poland

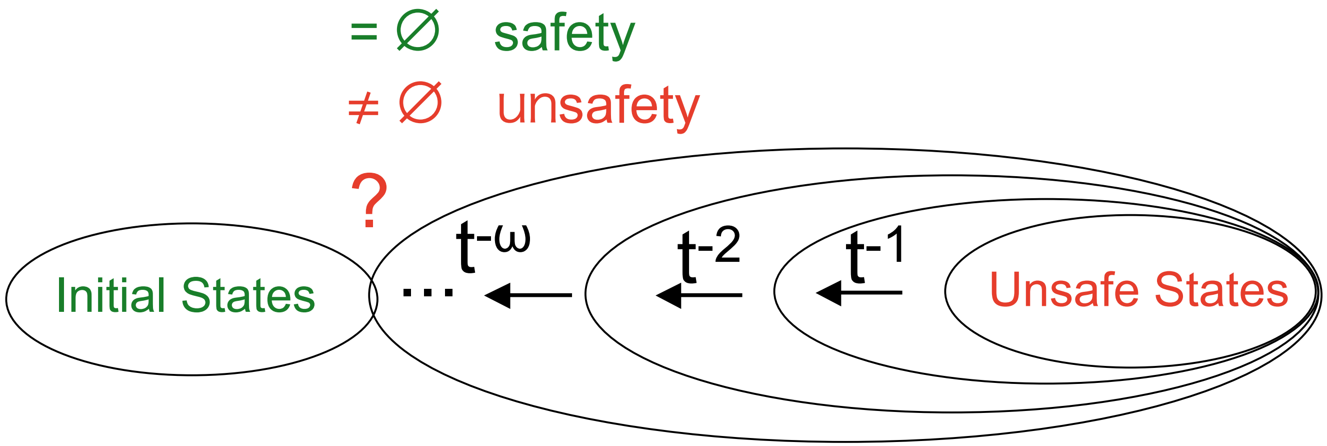
28-30 September 2011

Reachability Analysis

Forward:



Backward:



A System and its Specification

Specification:

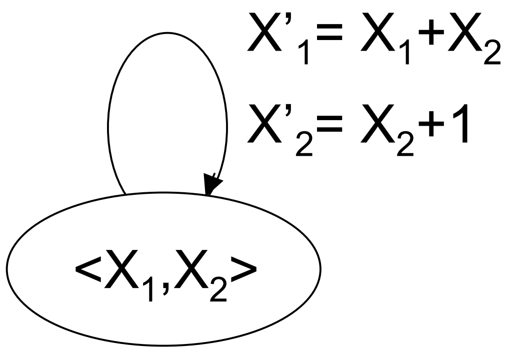
Var: integer X_1, integer X_2;

Init: $X_1 > 1 \wedge X_2 = 0$

Trans: $X'_1 = X_1 + X_2 \;\wedge\; X'_2 = X_2 + 1$

Safe: $\neg\, EF\, (X_2 > X_1)$

System:



Verifiers (ALV, FAST, LASH, TreX, etc.) may be unable to check safety because:

- the *Safe* property is in general undecidable,
- for backward reachability the initial states are not taken into account,
- for forward reachability the unsafe states are not taken into account.

Verification using Specification

Proposed method:

1. Specification ⇒ Constraint Logic Program
2. Constraint Logic Program ⇒ Specialized Constraint Logic Program
3. Specialized Constraint Logic Program ⇒ Specialized Specification

---

Verifiers are applied to the Specialized Specification.

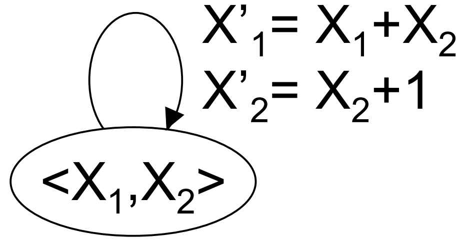
1. *Specification* ⇒ *Constraint Logic Program*

Var: integer X_1; integer X_2;

Init: $X_1 > 1 \wedge X_2 = 0$

Trans: $X'_1 = X_1 + X_2 \;\wedge\; X'_2 = X_2 + 1$

Safe: $\neg$ EF $(X_2 > X_1)$



(a Kripke structure)

⇒

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$(X_1,X_2) \leftarrow X_2 > X_1$

Th 1: the system is safe iff unsafe $\not\in$ M(Bw)

1. *Specification* ⇒ *Constraint Logic Program*

Bw :

$$unsafe \leftarrow init_1(X) \wedge bwReach(X)$$

$$\vdots$$

$$bwReach(X) \leftarrow t_1(X,X') \wedge bwReach(X')$$

$$\vdots$$

$$bwReach(X) \leftarrow u_1(X)$$

$$\vdots$$

2. *CLP* ⇒ *Specialized CLP* (1)

Bw:

1. unsafe ← $X_1 \geq 1 \wedge X_2 = 0 \wedge bwReach(X_1, X_2)$
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. $bwReach(X_1, X_2) \leftarrow X_2 > X_1$

Specialization via unfold/definition/fold

Unfold / Fold

"replace lhs by rhs"

$a(X) \leftarrow X_1 \geq 1 \wedge b(X,Y)$

$b(X,Y) \leftarrow X_1 \geq 0 \wedge c(X,Y)$

unfold →

$a(X) \leftarrow X_1 \geq 1 \wedge$ ~~$X_1 \geq 0$~~ $\wedge\ c(X,Y)$

$b(X,Y) \leftarrow X_1 \geq 0 \wedge c(X,Y)$

unfold is clause removal if the constraint is unsatisfiable.

Unfold / Fold

$a(X) \leftarrow X_1 \geq 1 \wedge b(X,Y)$

$b(X,Y) \leftarrow X_1 \geq 0 \wedge c(X,Y)$

unfold →

← fold

$a(X) \leftarrow X_1 \geq 1 \wedge c(X,Y)$

$b(X,Y) \leftarrow X_1 \geq 0 \wedge c(X,Y)$

“replace rhs by lhs”

2. *CLP* ⇒ *Specialized CLP* (2)

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. $bwReach(X_1, X_2) \leftarrow X_2 > X_1$

def-intro:

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

fold 1:

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

unfold 4:

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

def-intro:

$new2(X'_1, X'_2) \leftarrow X'_1 \geq 1 \wedge X'_2 = 1 \wedge bwReach(X'_1, X'_2)$

⋮

$new3(X''_1, X''_2) \leftarrow X''_1 \geq 1 \wedge X''_2 = 2 \wedge bwReach(X''_1, X''_2)$

⋮

nontermination ☹

2. CLP ⇒ Specialized CLP (3)

def-intro:

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

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

From 5 by unfold-fold:

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

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

SpBw:

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

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

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

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

information from the initial states

Th 2: unsafe $\not\in$ M(Bw) iff unsafe $\not\in$ M(SpBw)

Generalization: Convex-Hull + Widening (1)

definition:

$X_1 \geq 1 \wedge X_2 = 0$

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

0

new definition:

$X_1 \geq 1 \wedge X_2 = 1$

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

1

Convex-Hull

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

0 1

Generalization: Convex-Hull + Widening (2)

definition:

$X_1 \geq 1 \wedge X_2 = 0$

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

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

Widening

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

Computation of the Least Model

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. $bwReach(X_1, X_2) \leftarrow X_2 > X_1$

■ The computation of M(Bw) does not terminate. ☹

SpBw:

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

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

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

■ The computation of M(SpBw) terminates: ☺

↑ unsafe $\notin$ M(SpBw)

$new1(X_1, X_2) \leftarrow$ false

$new2(X'_1, X'_2) \leftarrow X'_1 \geq 1 \wedge \underline{X'_2 \geq 1}$

3. *Specialized CLP* ⇒ *New Specification*

SpBw:

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

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

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

⇒

SpVar: enumerated X_p {new1, new2}; integer X_1; integer X_2;

SpInit: $X_1 \geq 1 \wedge X_2 = 0 \wedge X_p = new1$

SpTrans: $(X_1 \geq 1 \wedge X_2 = 0 \wedge X_p = new1 \wedge X'_1 = X_1 \wedge X'_2 = 1 \wedge X_p = new2) \vee$
$(X_1 \geq 1 \wedge X_2 \geq 0 \wedge X_p = new2 \wedge X'_1 = X_1 + X_2 \wedge X'_2 = X_2 + 1 \wedge X_p = new2)$

SpSafe: $\neg$ EF $(X_1 \geq 1 \wedge X_2 > X_1 \wedge X_p = new2)$

Th 3: unsafe $\not\in$ M(SpBw) iff the specialized system is safe

Improvement of Termination

Specification:

Var: integer X_1; integer X_2;

Init: $X_1>1 \wedge X_2=0$

Trans: $X'_1= X_1+X_2 \wedge X'_2= X_2+1$

Safe: $\neg$ EF $(X_2>X_1)$

The verifier ALV fails. ☹

Specialized Specification:

SpVar: enumerated X_p {new1, new2}; integer X_1; integer X_2;

SpInit: $X_1 \geq 1 \wedge X_2=0 \wedge X_p=new1$

SpTrans: $(X_1 \geq 1 \wedge X_2=0 \wedge X_p=new1 \wedge X'_1=X_1 \wedge X'_2=1 \wedge X_p=new2) \vee$

$(X_1 \geq 1 \wedge X_2 \geq 0 \wedge X_p=new2 \wedge X'_1=X_1+X_2 \wedge X'_2=X_2+1 \wedge X_p=new2)$

SpSafe: $\neg$ EF $(X_1 \geq 1 \wedge X_2>X_1 \wedge X'_p=new2)$

The verifier ALV succeeds! ☺

Verification using Specialization

Proposed method:

algorithm 1

1. Specification ⇒ Constraint Logic Program

use of specializers (MAP, etc.)

2. Constraint Logic Program ⇒ Specialized Constraint Logic Program

algorithm 3

3. Specialized Constraint Logic Program ⇒ Specialized Specification

---

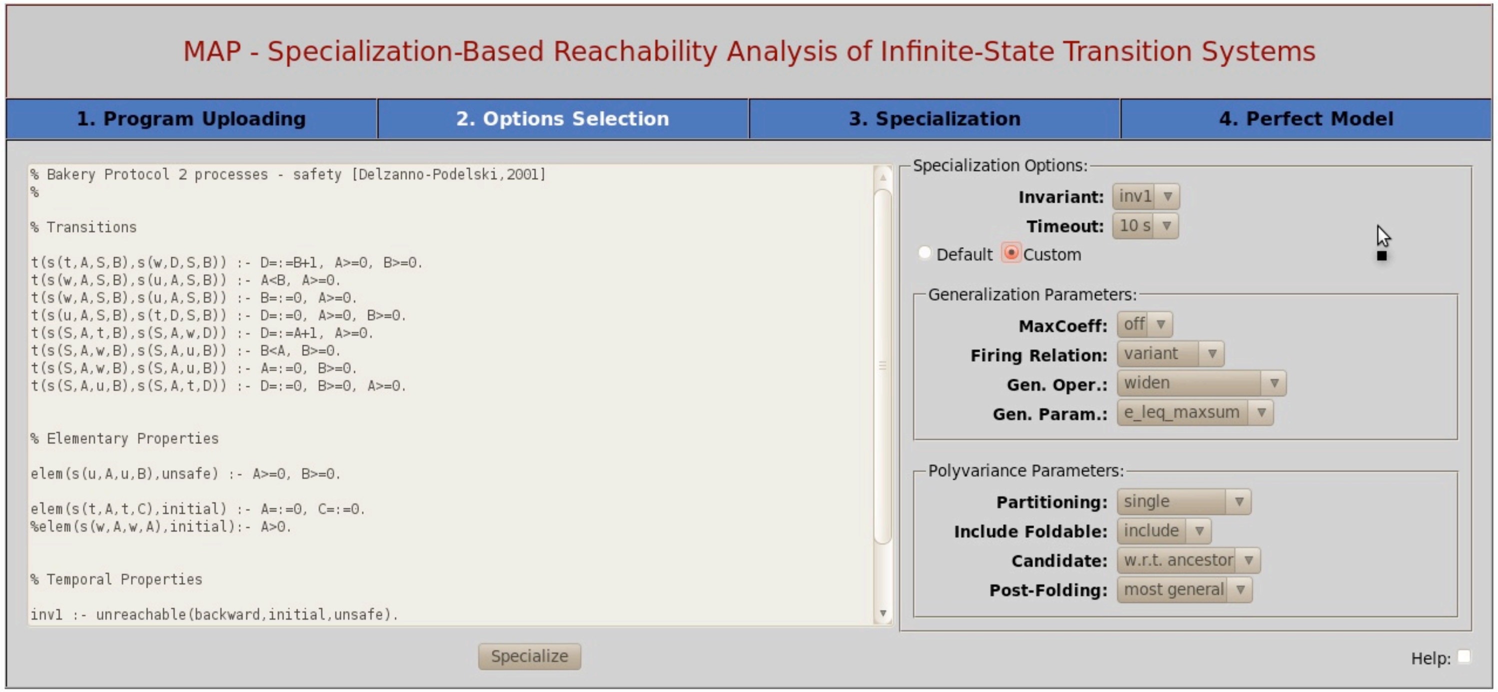
termination of the verifier (ALV, etc.) is improved.

The verifier is applied to the Specialized Specification.

MAP

MAP: a tool for program transformation, program specialization, etc.

http://map.uniroma2.it/mapweb



Generic Specialization Algorithm

Input: program Bw

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

Initialization: SpBw := {unsafe ← $init_1$(X) ∧ $newu_1$(X),...};

InDefs := {$newu_1$(X) ← $init_1$(X) ∧ bwReach(X),...};

Defs := InDefs;

while there exists a definition C: newp(X) ← c(X) ∧ bwReach(X) in InDefs

do unfold C, thereby getting a set SpC of clauses;

while there exists a clause E: newp(X) ← e(X,X') ∧ bwReach(X,X') in SpC

do definition introduction (use of generalization w.r.t. Defs);

fold, thereby getting new sets of clauses: Defs, InDefs, and SpC;

od

SpBw := SpBw ∪ SpC;

od

System Verification using ALV

Times in milliseconds. SpSys *includes also the specialization time.*
∞ *means "more than 600 seconds".* ⊥ *means "unable to verify".*

	Backward : default		Backward : A		Forward : F	
22 Systems	Sys	SpSys	Sys	SpSys	Sys	SpSys
Bakery 3	0.70	0.25	0.69	0.25	∞	0.25
Peterson	56.49	0.10	∞	0.10	∞	13.48
Futurebus	0.26	0.68	⊥	⊥	∞	∞
MESI	0.01	0.02	⊥	0.03	0.02	0.07
Reset Petri Nets	∞	0.02	⊥	⊥	∞	0.01
⋮						
Verified:	18	22	7	19	11	21

Conclusions

- A program specialization, uniform technique for improving performance of verifiers based on “fixpoint computations”.

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

Future Work

- Perform more system verifications and check scalability of the approach.
- Study of powerful generalization operators.
- Combined use of program specialization and abstract interpretation techniques.

References

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

P. Cousot and N. Halbwachs. *Automatic discover y of linear restraints among variables of a program.* In POPL, 84–96, 1978.

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

LASH: http://www.montefiore.ulg.ac.be/~boigelot/research/lash

M. Leuschel, T. Massart. *Infinite State Model Checking by Abstract Interpretations and Program Specialization.* In LOPSTR '99, LNCS 1817, 63-82. Springer, 2000.

ALV: T. Yavuz-Kahveci and T. Bultan. *Action Language Verifier: An Infinite-State Model Checker for Reactive Software Specifications*. Formal Methods in System Design, 35(3):325-367, 2009.