Improving Reachability Analysis of Infinite-State Systems by Specialization

F. Fioravanti[1], A. Pettorossi[2], M. Proietti[3], V. Senni[2]

[1]University of Chieti-Pescara 'd'Annunzio', Italy

[2]University of Rome 'Tor Vergata', Italy

[3]CNR-IASI, Rome, Italy

RP 2011, Genova, Italy

Outline

1. Reachability Analysis of Infinite-State Systems
2. Translation to Logic Programs
3. Specialization
4. Translation to Infinite-State Systems
5. Experimental Results
6. An Improvement

Context

Infinite-State Systems

- Finite automata with unbounded integer variables
- Specified by linear constraints on $\mathbb{Z}$ (e.g. $4x_1 - x_2 + 5x_3 \geq 0 \wedge \ldots$)
- A system is a 4-tuple $\langle Var, I, T, U \rangle$ of

  Var : a tuple of *enumerated* or *integer* variables

  I : a set of *initial states*, as a disjunction of *constraints* on *Var*

  T : a *transition relation*, as a disjunction of constraints on $\langle Var, Var' \rangle$ (Var' is the tuple of *primed* variables)

  U : a set of *unsafe states*, as a disjunction of constraints on *Var*

Backward Reachability

Infinite-State System : $\langle Var, I, T, U \rangle$

Given a set S of states, $\text{PRE}(S) = \{X \mid \exists X' \in S \;\; s.t. \;\; T(X, X')\}$

Goal: Check that $\text{PRE}^{\omega}(U) \cap I = \emptyset$

Problem: The computation of $\text{PRE}^{\omega}(U)$ may not terminate

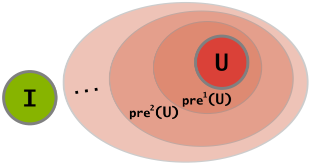
Backward Reachability

Infinite-State System : $\langle Var, I, T, U \rangle$

Given a set S of states, $\text{PRE}(S) = \{X \mid \exists X' \in S \; s.t. \; T(X, X')\}$

Goal: Check that $\text{PRE}^{\omega}(U) \cap I = \emptyset$

Problem: The computation of $\text{PRE}^{\omega}(U)$ may not terminate



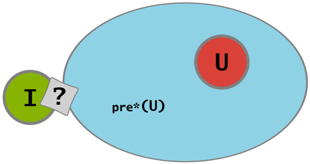
Backward Reachability

Infinite-State System : $\langle Var, I, T, U \rangle$

Given a set S of states, $\text{PRE}(S) = \{X \mid \exists X' \in S \;\; s.t. \;\; T(X, X')\}$

Goal: Check that $\text{PRE}^{\omega}(U) \cap I = \emptyset$

Problem: The computation of $\text{PRE}^{\omega}(U)$ may not terminate



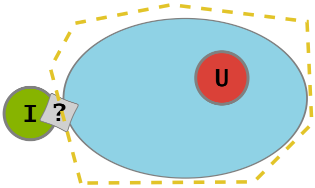
Backward Reachability

Infinite-State System : $\langle Var, I, T, U \rangle$

Given a set S of states, $\text{PRE}(S) = \{X \mid \exists X' \in S \;\; s.t. \;\; T(X, X')\}$

Goal: Check that $\text{PRE}^{\omega}(U) \cap I = \emptyset$

Problem: The computation of $\text{PRE}^{\omega}(U)$ may not terminate



approximate(PRE^{ω})$(U) \cap I = \emptyset$ ☺

approximate(PRE^{ω})$(U) \cap I \neq \emptyset$ **??**

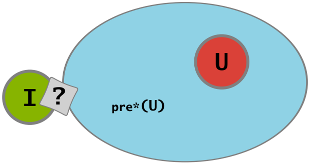
Backward Reachability

Infinite-State System : $\langle Var, I, T, U \rangle$

Given a set S of states, $\text{PRE}(S) = \{X \mid \exists X' \in S \;\; s.t. \;\; T(X, X')\}$

Goal: Check that $\text{PRE}^{\omega}(U) \cap I = \emptyset$

Problem: The computation of $\text{PRE}^{\omega}(U)$ may not terminate



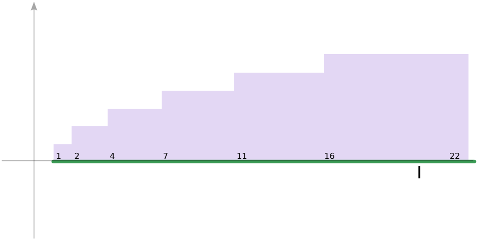
knowledge of the ***target*** (e.g. I) is not taken into account in the construction of $\text{PRE}^{\omega}(U)$

'Naive' Reachability Analysis

Example:

S:

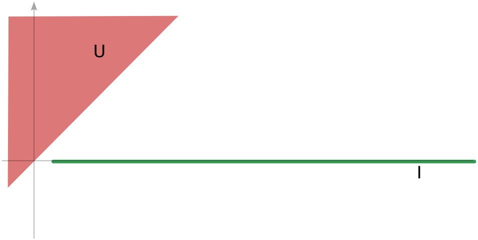
$$\begin{aligned} Var&: \textbf{integer } x_1;\ \textbf{integer } x_2; \\ I&: x_1 \geq 1 \wedge x_2 = 0; \\ T&: x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1; \\ U&: x_2 > x_1 \end{aligned}$$



'Naive' Reachability Analysis

Example:

$$S: \quad \begin{array}{rl} Var: & \textbf{integer } x_1;\ \textbf{integer } x_2; \\ I: & x_1 \geq 1 \wedge x_2 = 0; \\ T: & x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1; \\ U: & x_2 > x_1 \end{array}$$

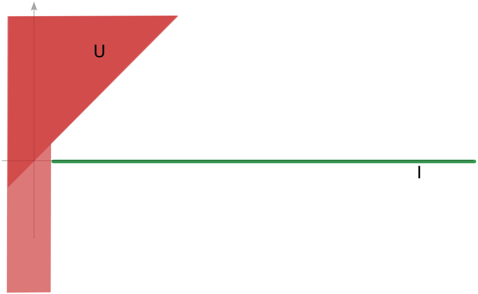


'Naive' Reachability Analysis

Example:

S:

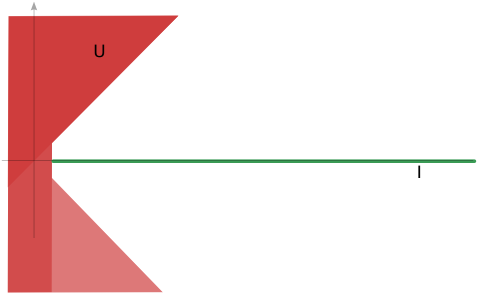
$$
\begin{aligned}
Var&: \textbf{integer } x_1; \ \textbf{integer } x_2; \\
I&: x_1 \geq 1 \wedge x_2 = 0; \\
T&: x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1; \\
U&: x_2 > x_1
\end{aligned}
$$



'Naive' Reachability Analysis

Example:

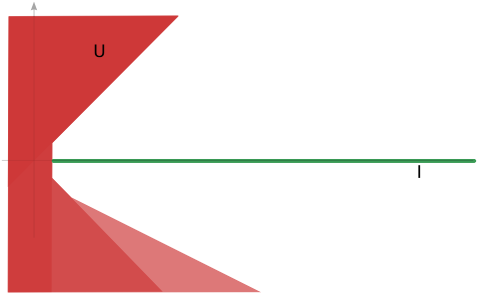
S:
- *Var*: **integer** x_1; **integer** x_2;
- *I*: $x_1 \geq 1 \wedge x_2 = 0$;
- *T*: $x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1$;
- *U*: $x_2 > x_1$



'Naive' Reachability Analysis

Example:

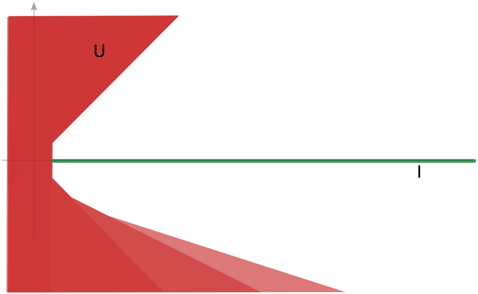
$$S: \quad \begin{array}{rl} Var: & \textbf{integer } x_1;\ \textbf{integer } x_2; \\ I: & x_1 \geq 1 \wedge x_2 = 0; \\ T: & x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1; \\ U: & x_2 > x_1 \end{array}$$



‘Naive’ Reachability Analysis

Example:

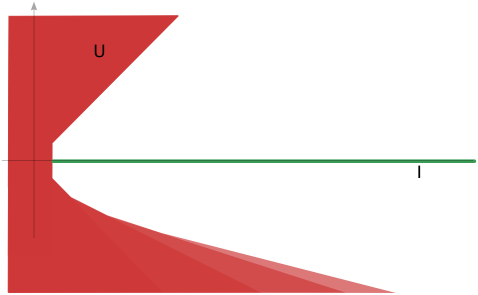
$$S: \quad \begin{aligned} Var&: \ \textbf{integer}\ x_1;\ \textbf{integer}\ x_2; \\ I&: \ x_1 \geq 1 \wedge x_2 = 0; \\ T&: \ x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1; \\ U&: \ x_2 > x_1 \end{aligned}$$



'Naive' Reachability Analysis

Example:

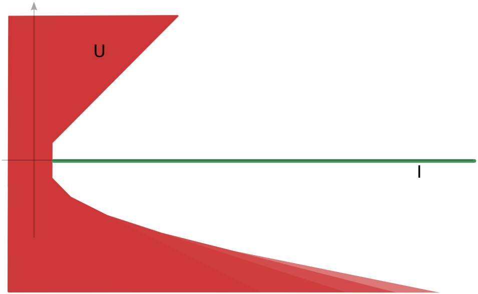
$$S: \quad \begin{array}{rl} \textit{Var}: & \textbf{integer } x_1;\ \textbf{integer } x_2; \\ I: & x_1 \geq 1 \wedge x_2 = 0; \\ T: & x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1; \\ U: & x_2 > x_1 \end{array}$$



'Naive' Reachability Analysis

Example:

$$S:\quad \begin{array}{rl} Var: & \textbf{integer } x_1;\ \textbf{integer } x_2; \\ I: & x_1 \geq 1 \wedge x_2 = 0; \\ T: & x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1; \\ U: & x_2 > x_1 \end{array}$$

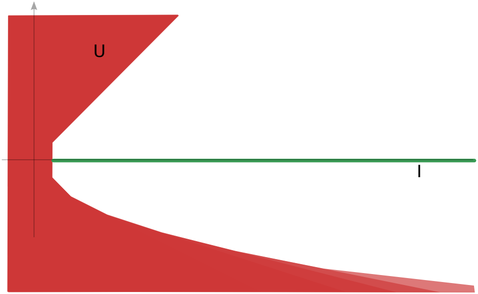


'Naive' Reachability Analysis

Example:

$$S: \quad \begin{array}{rl} Var: & \textbf{integer } x_1;\ \textbf{integer } x_2; \\ I: & x_1 \geq 1 \ \wedge\ x_2 = 0; \\ T: & x_1' = x_1 + x_2 \ \wedge\ x_2' = x_2 + 1; \\ U: & x_2 > x_1 \end{array}$$

The backward reachability strategy does not terminate



Program Specialization

A *program optimization technique* exploiting knowledge about *context of use* to obtain more *efficient* programs [Jones et al. 93, Gallagher 93]

Focus : *Constraint Logic Programs* (i.e. with linear constraints on $\mathbb{Z}$)

Input : A CLP program P, a predicate $p(x)$, a context $c(x)$ (e.g. $1 < x \leq 5$)

$$\delta : \ p_s(x) \ \leftarrow \ c(x) \wedge p(x)$$

Output : A new program *SpecP* s.t.

$$p_s(t) \in M(P \cup \{\delta\}) \ \text{ iff } \ p_s(t) \in M(SpecP)$$

where $M(P)$ is the least $\mathbb{Z}$-model of P

Advantage : *SpecP* faster that P on $p_s(t)$ queries

Specialization-based Reachability Analysis

System S $\quad\quad\quad\quad$ Specialized System $SpecS$

$$\begin{array}{ccc} \downarrow & & \uparrow \\ P_S \in \mathrm{CLP}(\mathbb{Z}) & \stackrel{specialize}{\Longrightarrow} & SpecP_S \in \mathrm{CLP}(\mathbb{Z}) \end{array}$$

Input: $S = \langle Var, I, T, U \rangle$

Output: $SpecS = \langle SpecVar, SpecI, SpecT, SpecU \rangle$

BACKWARD we specialize w.r.t. the *Initial States* (target states)

$$\mathrm{PRE}^{\omega}(U) \cap I = \emptyset \quad \textbf{iff} \quad \mathrm{PRE}^{\omega}(SpecU) \cap SpecI = \emptyset$$

FORWARD we specialize w.r.t. the *Unsafe States* (target states)

$$\mathrm{POST}^{\omega}(I) \cap U = \emptyset \quad \textbf{iff} \quad \mathrm{POST}^{\omega}(SpecI) \cap SpecU = \emptyset$$

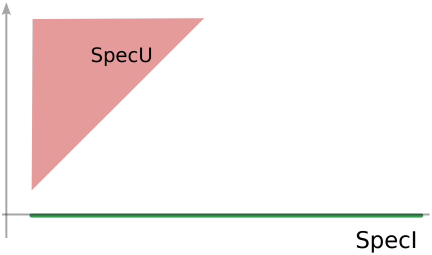
Improved Reachability Analysis

Motivation:
Termination of reachability analysis for the *Specialized system* is improved

Example specialized:

SpecS:

Var: **integer** x_1; **integer** x_2;
SpecI: $x_1 \geq 1 \wedge x_2 = 0$;
SpecT: $x_1 \geq 1 \wedge x_2 \geq 0 \wedge x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1$;
SpecU: $x_1 \geq 1 \wedge x_2 > x_1$



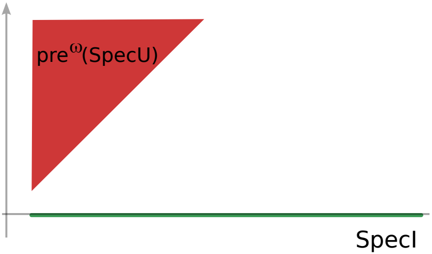
Improved Reachability Analysis

Motivation:
Termination of reachability analysis for the *Specialized system* is improved

Example specialized:

SpecS:

$$\begin{aligned} Var&: \textbf{integer } x_1;\ \textbf{integer } x_2; \\ SpecI&: x_1 \geq 1 \land x_2 = 0; \\ SpecT&: x_1 \geq 1 \land x_2 \geq 0 \land x_1' = x_1 + x_2 \land x_2' = x_2 + 1; \\ SpecU&: x_1 \geq 1 \land x_2 > x_1 \end{aligned}$$



SpecU is a fixpoint

$$\text{PRE}^{\omega}(SpecU) \cap SpecI = \emptyset$$
$$\Downarrow$$
$$\text{PRE}^{\omega}(U) \cap I = \emptyset$$

Advantages of Specialization

- Improvement of termination of reachability analysis
- Computation of invariants and their propagation on the transition relation
- Modification of the structure of the system to capture (possibly non-convex) invariants
- Specialization is equivalence preserving and terminating
- Specialization is independent of the verification tool

Specialization-based Analysis

Step 1: Encoding (Backward) Reachability in CLP

$$Sys = \langle Var, I, T, U \rangle \quad \text{where} \quad \begin{array}{rl} I: & init_1(X) \vee \ldots \vee init_k(X) \\ T: & t_1(X, X') \vee \ldots \vee t_m(X, X') \\ U: & u_1(X) \vee \ldots \vee u_n(X) \end{array}$$

$$\begin{array}{rrcl}
I_1: & unsafe & \leftarrow & init_1(X) \wedge bwReach(X) \\
 & & \vdots & \\
I_k: & unsafe & \leftarrow & init_k(X) \wedge bwReach(X) \\
T_1: & bwReach(X) & \leftarrow & t_1(X, X') \wedge bwReach(X') \\
 & & \vdots & \\
T_m: & bwReach(X) & \leftarrow & t_m(X, X') \wedge bwReach(X') \\
U_1: & bwReach(X) & \leftarrow & u_1(X) \\
 & & \vdots & \\
U_n: & bwReach(X) & \leftarrow & u_n(X)
\end{array}$$

see also [Fribourg 97, Delzanno-Podelski 99]

Step 2: Specialization of CLP - Reachability

Input :

A CLP program P defining a predicate $bwReach(x)$ and $Init(x)$

Method :

Introduce δ : $unsafe \leftarrow Init(x) \wedge bwReach(x)$

Specialize $P \cup \{\delta\}$ w.r.t. $unsafe$ queries

Output :

A specialized program $SpecP$ s.t.

$unsafe \in M(P \cup \{\delta\})$ iff $unsafe \in M(SpecP)$

Rule-Based Specialization

Definition (*introduction of a new region*)

$$\longmapsto \quad newp(x) \;\leftarrow\; c(x) \wedge p(x)$$

Unfolding (*application of the transition relation*)

$$\boxed{\begin{array}{lcl} p(x) & \leftarrow & d_1 \wedge B_1 \\ & \vdots & \\ p(x) & \leftarrow & d_n \wedge B_n \end{array}} \quad H \leftarrow c \wedge p(x) \wedge R \quad \longmapsto \quad \begin{array}{lcl} H & \leftarrow & c \wedge d_1 \wedge B_1 \wedge R \\ & \vdots & \\ H & \leftarrow & c \wedge d_n \wedge B_n \wedge R \end{array}$$

Clause Removal (*simplification of transitions and regions*)

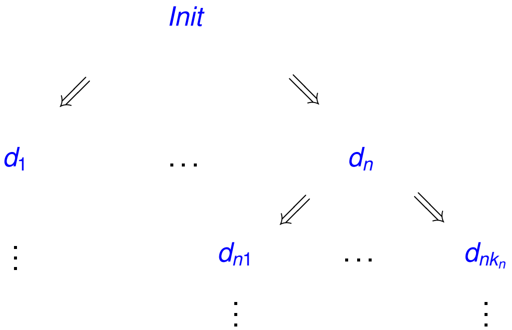
1. $\begin{array}{lcl} H & \leftarrow & c \wedge B \\ H & \leftarrow & d \end{array} \quad \longmapsto \quad H \leftarrow d$ if $c \sqsubseteq d$ (c entails d)
2. $H \leftarrow c \wedge B \quad \longmapsto \quad \emptyset$ if c is unsatisfiable

Folding (*closure of loops*)

$$\boxed{p(x) \;\leftarrow\; d \wedge B} \quad H \leftarrow c \wedge B \wedge R \quad \overset{c \sqsubseteq d}{\longmapsto} \quad H \leftarrow c \wedge p(x) \wedge R$$

Specialization Illustrated - Overall View

BACKWARD



Construct a ***finite*** tree of *regions* (represented by constraints), from which we will extract the specialized system *SpecS*

Specialization Illustrated - Single Step

Where, one $\Longrightarrow$ step consists of:

	$c(X)$		current region
unfold $\swarrow$		$\searrow$ unfold	
$c(X) \wedge t_1(X, X')$	$\dots$	$c(X) \wedge t_n(X, X')$	apply transitions & simplify
project $\downarrow$		$\downarrow$ project	
$d_1(X')$	$\dots$	$d_n(X')$	successor regions
$\sqsubseteq$ $\downarrow$		$\downarrow$ $\sqsubseteq$	
$gen(d_1(X'))$	$\dots$	$gen(d_n(X'))$	invariant (included in an ancestor region) or new region

Specialization Strategy

Input: P and a clause δ_0: $p_{sp}(x) \leftarrow c(x) \wedge p(x)$

Output: $SpecP$ s.t. $p_{sp}(z) \in M(P \cup \{\delta_0\})$ iff $p_{sp}(z) \in M(SpecP)$

$SpecP := \emptyset$;
$Defs := \{\delta_0\}$;

while $\exists\, \delta \in Defs$ **do**

$\Gamma := Unfold\ \delta$

$\Delta := ClauseRemoval\ \Gamma$

$(\Phi, NewDefs) := Generalize\&Fold\ \Delta$

$Defs := (Defs - \{\delta\}) \cup NewDefs$

$SpecP := SpecP \cup \Phi$

od

Specialization Strategy

Input: P and a clause δ_0: $p_{sp}(x) \leftarrow c(x) \wedge p(x)$

Output: $SpecP$ s.t. $p_{sp}(z) \in M(P \cup \{\delta_0\})$ iff $p_{sp}(z) \in M(SpecP)$

$SpecP := \emptyset;$
$Defs := \{\delta_0\};$

while $\exists\, \delta \in Defs$ **do**

$\Gamma := \mathit{Unfold}\ \delta$

$\Delta := \mathit{ClauseRemoval}\ \Gamma$

$(\Phi, NewDefs) := \mathit{Generalize\&Fold}\ \Delta$ $\Leftarrow$

$Defs := (Defs - \{\delta\}) \cup NewDefs$

$SpecP := SpecP \cup \Phi$

od

We ensure *termination* by a careful choice of the *generalization operator* [à la Cousot-Halbwachs 78]

Correctness of the Specialization Strategy

Let

$$P \longmapsto^{*} \mathit{SpecP}$$

be a transformation sequence produced by the Specialization Strategy

Theorem

$p_{sp}(z) \in M(P \cup \{\delta_0\})$ *iff* $p_{sp}(z) \in M(\mathit{SpecP})$

Proof: By correctness of the transformation rules, since $P \longmapsto^{*} \mathit{SpecP}$ is an *admissible* transformation sequence [Tamaki-Sato 84, Etalle-Gabrielli 96]

Outcome of the Specialization Strategy

Let P be:

$$I_1: \quad unsafe \leftarrow x_1 \geq 1 \wedge x_2 = 0 \wedge bwReach(x_1, x_2)$$

$$T_1: \quad bwReach(x_1, x_2) \leftarrow x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1 \wedge bwReach(x_1', x_2')$$

$$U_1: \quad bwReach(x_1, x_2) \leftarrow x_2 > x_1$$

By specializing P w.r.t. *unsafe* we obtain *SpecP*

$$J_1: \quad unsafe \leftarrow x_1 \geq 1 \wedge x_2 = 0 \wedge new1(x_1, x_2)$$

$$S_1: \quad new1(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 = 0 \wedge x_1' = x_1 \wedge x_2' = 1 \wedge new2(x_1', x_2')$$

$$S_2: \quad new2(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 = 1 \wedge x_1' = x_1 + 1 \wedge x_2' = 2 \wedge new3(x_1', x_2')$$

$$S_3: \quad new3(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 \geq 1 \wedge x_1' = x_1 + x_2 \wedge x_2' = x_2 + 1 \wedge new3(x_1', x_2')$$

$$V_1: \quad new3(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 > x_1$$

Step 3: From CLP back to Infinite-State Systems

$newp_i,\ newq_i,\ newt_i,\ newu_i$ not necessarily distinct

SpecP:

$$
\begin{array}{rcl}
unsafe & \leftarrow & init_1(X) \wedge newp_1(X) \\
 & \vdots & \\
unsafe & \leftarrow & init_k(X) \wedge newp_k(X) \\
newq_1(X) & \leftarrow & s_1(X, X') \wedge newt_1(X') \\
 & \vdots & \\
newq_m(X) & \leftarrow & s_m(X, X') \wedge newt_m(X') \\
newu_1(X) & \leftarrow & v_1(X) \\
 & \vdots & \\
newu_n(X) & \leftarrow & v_n(X)
\end{array}
$$

SpecVar : **Var, enumerated** $x_p\ \{newp_1, newq_1, newt_1, newu_1, \ldots\}$

SpecI : $(init_1(X) \wedge x_p = newp_1) \vee \ldots \vee$
$(init_k(X) \wedge x_p = newp_k)$

SpecT : $(x_p = newq_1 \wedge s_1(X, X') \wedge x_p = newt_1) \vee \ldots \vee$
$(x_p = newq_m \wedge s_m(X, X') \wedge x_p = newt_m)$

SpecU : $(v_1(X) \wedge x_p = newu_1) \vee \ldots \vee$
$(v_n(X) \wedge x_p = newu_n)$

the specialized system is:

$\langle SpecVar, SpecI, SpecT, SpecU \rangle$

Experiments

EXAMPLES	default *Sys*	default *SpSys*	A *Sys*	A *SpSys*	F *Sys*	F *SpSys*
Bakery2	0.03	0.05	0.03	0.05	0.06	0.04
Bakery3	0.70	0.25	0.69	0.25	∞	3.68
MutAst	1.46	0.37	1.00	0.37	0.22	0.59
Peterson	56.49	0.10	∞	0.10	∞	13.48
Ticket	∞	0.03	0.10	0.03	0.02	0.19
Berkeley RISC	0.01	0.04	$\perp$	0.04	0.01	0.02
DEC Firefly	0.01	0.02	$\perp$	0.03	0.01	0.07
IEEE Futurebus	0.26	0.68	$\perp$	$\perp$	∞	∞
Illinois University	0.01	0.03	$\perp$	0.03	∞	0.07
Barber	0.62	0.21	$\perp$	0.21	∞	0.08
CSM	56.39	7.69	$\perp$	7.69	∞	125.32
Consistency	∞	0.11	$\perp$	0.11	∞	324.14
Insertion Sort	0.03	0.06	0.04	0.06	0.18	0.02
Selection Sort	∞	0.21	$\perp$	0.21	∞	0.33
Reset Petri Net	∞	0.02	$\perp$	$\perp$	∞	0.01
Train	42.24	59.21	$\perp$	$\perp$	∞	0.46
No. of verified properties	12	16	5	13	6	15
	BACKWARD		BACKWARD		FORWARD	

Timings : *Sys* = **fixpoint only**
SpSys = **specialization + fixpoint**

fixpoint computed using ALV [Bultan et al. 09], based on the Omega library

'$\perp$' = 'Unable to verify' and '∞' = 'Timeout' (10 minutes)

	default		A		F	
EXAMPLES	*Sys*	*SpSys*	*Sys*	*SpSys*	*Sys*	*SpSys*
Bakery2	0.03	0.05	0.03	0.05	0.06	0.04
Bakery3	0.70	0.25	0.69	0.25	∞	3.68
MutAst	1.46	0.37	1.00	0.37	0.22	0.59
Peterson	56.49	0.10	∞	0.10	∞	13.48
Ticket	∞	0.03	0.10	0.03	0.02	0.19
Berkeley RISC	0.01	0.04	$\perp$	0.04	0.01	0.02
DEC Firefly	0.01	0.02	$\perp$	0.03	0.01	0.07
IEEE Futurebus	0.26	0.68	$\perp$	$\perp$	∞	∞
Illinois University	0.01	0.03	$\perp$	0.03	∞	0.07
Barber	0.62	0.21	$\perp$	0.21	∞	0.08
CSM	56.39	7.69	$\perp$	7.69	∞	125.32
Consistency	∞	0.11	$\perp$	0.11	∞	324.14
Insertion Sort	0.03	0.06	0.04	0.06	0.18	0.02
Selection Sort	∞	0.21	$\perp$	0.21	∞	0.33
Reset Petri Net	∞	0.02	$\perp$	$\perp$	∞	0.01
Train	42.24	59.21	$\perp$	$\perp$	∞	0.46
No. of verified properties	12	16	5	13	6	15
	BACKWARD		BACKWARD		FORWARD	

- Specialization *improves precision*
- Overall, it *does not deteriorate verification time*
- Applicable in both *forward and backward* analyses

An Improvement

Relaxing Constraints from $\mathbb{Z}$ to $\mathbb{R}$

Specialization: $P_S \mapsto P_1 \mapsto \ldots \mapsto P_n = SpecP_S$

a sequence of transformation steps with applicability conditions on $\mathbb{Z}$

Problem: Constraint manipulation on $\mathbb{Z}$ is costly

Idea: Relax constraints from $\mathbb{Z}$ to $\mathbb{R}$

Relaxing System Semantics

Counter System: interpreted on $\mathbb{Z}$

$$\mathsf{T}: \quad \begin{array}{lll} (1) & \langle X, Y \rangle \longrightarrow \langle X, Y-1 \rangle & \text{if} \quad X \geq 1 \\ (2) & \langle X, Y \rangle \longrightarrow \langle X, Y+2 \rangle & \text{if} \quad X \leq 2 \\ (3) & \langle X, Y \rangle \longrightarrow \langle X, -1 \rangle & \text{if} \quad \exists\, V \in \mathbb{Z} \;\; Y = 2V+1 \end{array}$$

Relaxing System Semantics

Counter System: interpreted on $\mathbb{Z}$

$$
T: \quad
\begin{array}{lll}
(1) & \langle X, Y\rangle \longrightarrow \langle X, Y-1\rangle & \text{if} \;\; X \geq 1 \\
(2) & \langle X, Y\rangle \longrightarrow \langle X, Y+2\rangle & \text{if} \;\; X \leq 2 \\
(3) & \langle X, Y\rangle \longrightarrow \langle X, -1\rangle & \text{if} \;\; \exists\, V \in \mathbb{Z} \;\; Y = 2V+1
\end{array}
$$

to show: $T^\omega(\langle 0, 0\rangle) \cap \{\langle X, Y\rangle \mid Y < 0\} = \emptyset$

Relaxing System Semantics

Counter System: interpreted on $\mathbb{Z}$

$$T: \begin{array}{lll} (1) & \langle X, Y\rangle \longrightarrow \langle X, Y-1\rangle & \text{if } X \geq 1 \\ (2) & \langle X, Y\rangle \longrightarrow \langle X, Y+2\rangle & \text{if } X \leq 2 \\ (3) & \langle X, Y\rangle \longrightarrow \langle X, -1\rangle & \text{if } \exists V \in \mathbb{R}\ \ Y = 2V+1 \quad \text{always true} \end{array}$$

to show: $T^{\omega}(\langle 0, 0\rangle) \cap \{\langle X, Y\rangle \mid Y < 0\} = \emptyset$

relax: $relax_{\mathbb{R}}(T)^{\omega}(\langle 0, 0\rangle) \cap \{\langle X, Y\rangle \mid Y < 0\} \neq \emptyset$ **??**

Relaxed system cannot be proved safe, *no matter the technique we use*

Idea: relax rules conditions rather than system interpretation

Equivalence Preserving Relaxations

Lemma

if c *is* $\mathbb{R}$*-unsatisfiable* *then* c *is* $\mathbb{Z}$*-unsatisfiable*
if $c \sqsubseteq_{\mathbb{R}} d$ *then* $c \sqsubseteq_{\mathbb{Z}} d$

Theorem (2) [LOPSTR'11]

if in the transformation rules we replace $\mathbb{Z}$*-unsatisfiability by* $\mathbb{R}$*-unsatisfiability and* $c \sqsubseteq_{\mathbb{R}} d$ *by* $c \sqsubseteq_{\mathbb{Z}} d$*, then equivalence is preserved:*

$$unsafe \in M(P_S) \qquad iff \qquad unsafe \in M(SpecP_S)$$

Outcome:

- We can use efficient and scalable solvers on $\mathbb{R}$
- We can use generalization operators on $\mathbb{R}$ (convex-hull,widening,. . .)
- We preserve equivalence of systems wrt safety properties
- Drawback: specialized system undersimplified

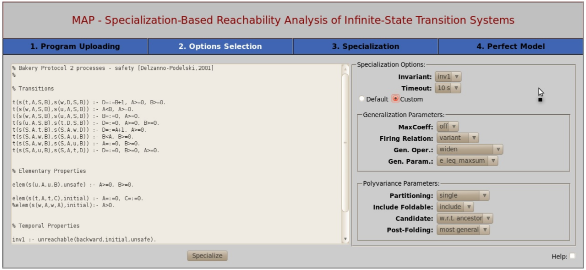
Conclusions

- A very general technique for improving termination of verification tools based on fixpoint computation
- Independent of the tool used for verification
- We are currently experimenting with:
  - other tools (e.g. FASTer [Bardin et al. 08]),
  - other classes of transition systems (Hybrid Systems), and
  - properties (LTL, CTL*)
- Experiments show that specialization improves precision without significant overhead in the verification time
- More experiments are needed to attest scalability of this technique

Advertisement

An implementation in SICStus Prolog as a module of our MAP system.

Give it a try!



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