

Improving Reachability Analysis of Infinite-State Systems by Specialization

Valerio Senni

University of Rome 'Tor Vergata', Italy

LIFC, Univ. de Franche-Comté, October 2011

joint work with Fabio Fioravanti, Alberto Pettorossi, and Maurizio Proietti

Transformation of Logic Programs

A program as a theory \implies theory transformation:
change the axioms of a theory,
while preserving models

Syntax

Semantics

logic programs

least Herbrand model

+ negation

perfect model

+ constraints

least/perfect \mathcal{D} -model

Applications :

theorem proving

$P \cup \{\varphi\} \mapsto^* \mathbf{true}$

'optimization' as

program verification

$\llbracket P \rrbracket \cup \{\varphi\} \mapsto^* \mathbf{true}$

program synthesis

$\varphi \mapsto^* P$ s.t. $\llbracket P \rrbracket \models \varphi$

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. $4x_1 - x_2 + 5x_3 \geq 0 \wedge \dots$)
- 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, $PRE(S) = \{X \mid \exists X' \in S \text{ s.t. } T(X, X')\}$

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

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



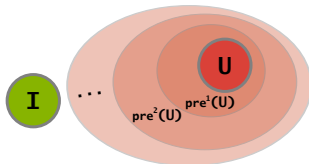
Backward Reachability

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

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

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

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



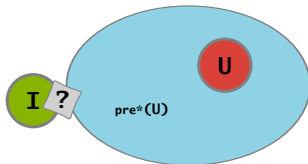
Backward Reachability

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

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

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

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



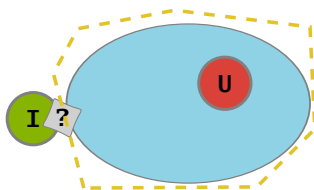
Backward Reachability

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

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

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

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



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

$\text{approximate}(\text{PRE}^\omega)(U) \cap I \neq \emptyset$???

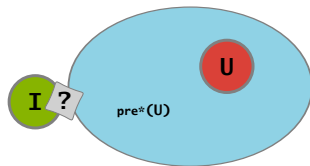
Backward Reachability

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

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

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

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



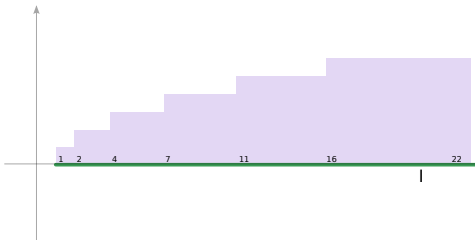
knowledge of the **target** (e.g. I)
is not taken into account
in the construction of $PRE^\omega(U)$

'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

S: $I: x_1 \geq 1 \wedge x_2 = 0;$
 $T: x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1;$
 $U: x_2 > x_1$



'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

S: $I: x_1 \geq 1 \wedge x_2 = 0;$
 $T: x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1;$
 $U: x_2 > x_1$

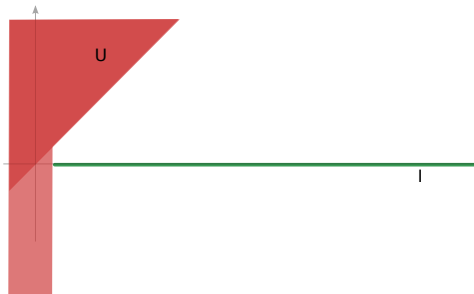


'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

S: $I: x_1 \geq 1 \wedge x_2 = 0;$
 $T: x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1;$
 $U: x_2 > x_1$

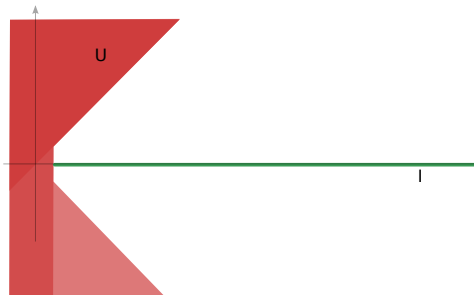


'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

S: $I: x_1 \geq 1 \wedge x_2 = 0;$
 $T: x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1;$
 $U: x_2 > x_1$

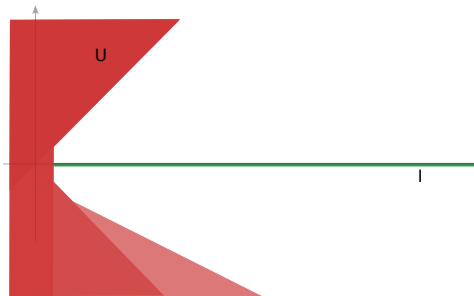


'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

S: $I: x_1 \geq 1 \wedge x_2 = 0;$
 $T: x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1;$
 $U: x_2 > x_1$

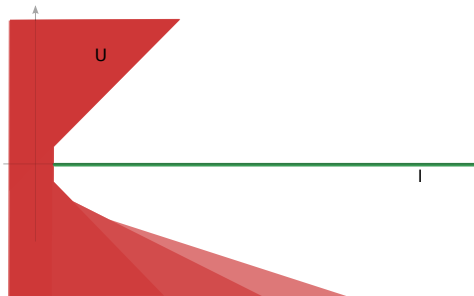


'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

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

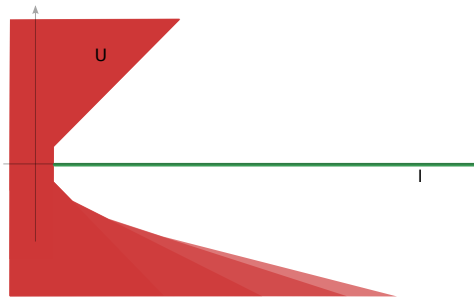


'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

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

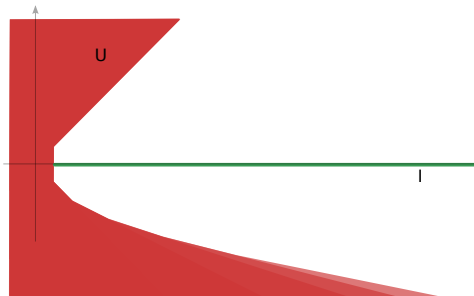


'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

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



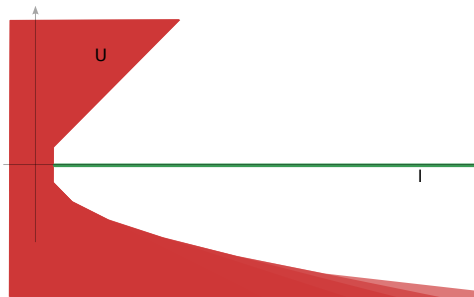
'Naive' Reachability Analysis

Example:

Var: **integer** x_1 ; **integer** x_2 ;

S: $I: x_1 \geq 1 \wedge x_2 = 0;$
 $T: x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1;$
 $U: x_2 > x_1$

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. $1 < x \leq 5$)

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

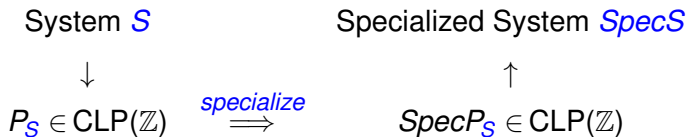
Output : A new program *SpecP* s.t.

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

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

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

Specialization-based Reachability Analysis



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

Output: $\textit{Spec}S = \langle \textit{SpecVar}, \textit{Spec}I, \textit{Spec}T, \textit{Spec}U \rangle$

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

$$\text{PRE}^\omega(U) \cap I = \emptyset \quad \text{iff} \quad \text{PRE}^\omega(\textit{Spec}U) \cap \textit{Spec}I = \emptyset$$

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

$$\text{POST}^\omega(I) \cap U = \emptyset \quad \text{iff} \quad \text{POST}^\omega(\textit{Spec}I) \cap \textit{Spec}U = \emptyset$$

Improved Reachability Analysis

Motivation:

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

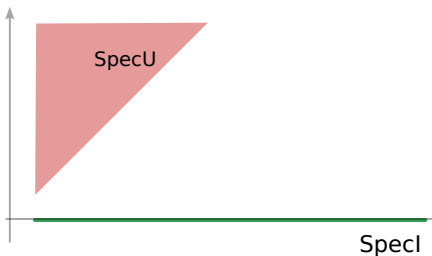
Example specialized:

Var: **integer** x_1 ; **integer** x_2 ;

SpecS: $x_1 \geq 1 \wedge x_2 = 0$;

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

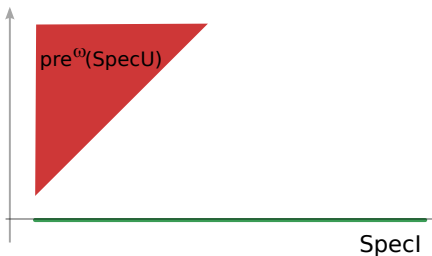
Motivation:

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

Example specialized:

Var: **integer** x_1 ; **integer** x_2 ;

SpecS: $SpecI: x_1 \geq 1 \wedge x_2 = 0$;
 $SpecT: x_1 \geq 1 \wedge x_2 \geq 0 \wedge x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1$;
 $SpecU: x_1 \geq 1 \wedge x_2 > x_1$



SpecU is a fixpoint

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

$$\Downarrow$$

$$PRE^\omega(U) \cap I = \emptyset$$

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

$$\text{Sys} = \langle \text{Var}, I, T, U \rangle \quad \text{where}$$

$$I: \text{init}_1(X) \vee \dots \vee \text{init}_k(X)$$

$$T: t_1(X, X') \vee \dots \vee t_m(X, X')$$

$$U: u_1(X) \vee \dots \vee u_n(X)$$

$$I_1: \quad \text{unsafe} \leftarrow \text{init}_1(X) \wedge \text{bwReach}(X)$$

$$\vdots$$

$$I_k: \quad \text{unsafe} \leftarrow \text{init}_k(X) \wedge \text{bwReach}(X)$$

$$T_1: \quad \text{bwReach}(X) \leftarrow t_1(X, X') \wedge \text{bwReach}(X')$$

$$\vdots$$

$$T_m: \quad \text{bwReach}(X) \leftarrow t_m(X, X') \wedge \text{bwReach}(X')$$

$$U_1: \quad \text{bwReach}(X) \leftarrow u_1(X)$$

$$\vdots$$

$$U_n: \quad \text{bwReach}(X) \leftarrow u_n(X)$$

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 $\delta : unsafe \leftarrow Init(x) \wedge bwReach(x)$

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)

$$\mapsto \text{new}p(x) \leftarrow c(x) \wedge p(x)$$

Unfolding

(application of the transition relation)

$$\boxed{\begin{array}{l} p(x) \leftarrow d_1 \wedge B_1 \\ \vdots \\ p(x) \leftarrow d_n \wedge B_n \end{array}}$$

$$H \leftarrow c \wedge p(x) \wedge R \mapsto \begin{array}{l} H \leftarrow c \wedge d_1 \wedge B_1 \wedge R \\ \vdots \\ H \leftarrow c \wedge d_n \wedge B_n \wedge R \end{array}$$

Clause Removal

(simplification of transitions and regions)

- $$H \leftarrow c \wedge B \mapsto H \leftarrow d \quad \text{if } c \sqsubseteq d \quad (c \text{ entails } d)$$
- $$H \leftarrow c \wedge B \mapsto \emptyset \quad \text{if } c \text{ is unsatisfiable}$$

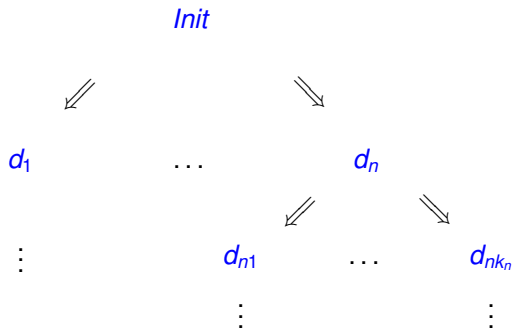
Folding

(closure of loops)

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

Specialization Illustrated - Overall View

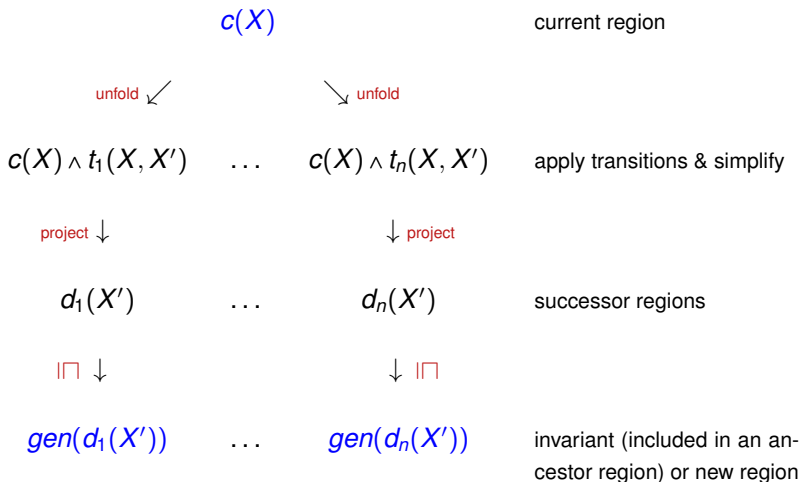
BACKWARD



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

Specialization Illustrated - Single Step

Where, one \implies step consists of:



Specialization Strategy

Input: P and a clause $\delta_0: p_{sp}(x) \leftarrow c(x) \wedge p(x)$

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**

Γ $:=$ *Unfold* δ

Δ $:=$ *ClauseRemoval* Γ

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

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

$SpecP := SpecP \cup \Phi$

od

Specialization Strategy

Input: P and a clause $\delta_0: p_{sp}(x) \leftarrow c(x) \wedge p(x)$

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**

Γ $:=$ *Unfold* δ

Δ $:=$ *ClauseRemoval* Γ

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

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

$SpecP := SpecP \cup \Phi$

od

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



Correctness of the Specialization Strategy

Let

$$P \longmapsto^* \text{Spec}P$$

be a transformation sequence produced by the Specialization Strategy

Theorem

$$p_{sp}(z) \in M(P \cup \{\delta_0\}) \text{ iff } p_{sp}(z) \in M(\text{Spec}P)$$

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

Outcome of the Specialization Strategy

Let P be:

$$\begin{aligned}
 I_1 : \quad & \text{unsafe} \leftarrow x_1 \geq 1 \wedge x_2 = 0 \wedge \text{bwReach}(x_1, x_2) \\
 T_1 : \quad & \text{bwReach}(x_1, x_2) \leftarrow x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1 \wedge \text{bwReach}(x'_1, x'_2) \\
 U_1 : \quad & \text{bwReach}(x_1, x_2) \leftarrow x_2 > x_1
 \end{aligned}$$

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

$$\begin{aligned}
 J_1 : \quad & \text{unsafe} \leftarrow x_1 \geq 1 \wedge x_2 = 0 \wedge \text{new1}(x_1, x_2) \\
 S_1 : \quad & \text{new1}(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 = 0 \wedge x'_1 = x_1 \wedge x'_2 = 1 \wedge \text{new2}(x'_1, x'_2) \\
 S_2 : \quad & \text{new2}(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 = 1 \wedge x'_1 = x_1 + 1 \wedge x'_2 = 2 \wedge \text{new3}(x'_1, x'_2) \\
 S_3 : \quad & \text{new3}(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 \geq 1 \wedge x'_1 = x_1 + x_2 \wedge x'_2 = x_2 + 1 \wedge \text{new3}(x'_1, x'_2) \\
 V_1 : \quad & \text{new3}(x_1, x_2) \leftarrow x_1 \geq 1 \wedge x_2 > x_1
 \end{aligned}$$

Step 3: From CLP back to Infinite-State Systems

$$\begin{array}{l}
 \text{unsafe} \leftarrow \text{init}_1(X) \wedge \text{newp}_1(X) \\
 \vdots \\
 \text{unsafe} \leftarrow \text{init}_k(X) \wedge \text{newp}_k(X) \\
 \\
 \text{SpecP: } \text{newq}_1(X) \leftarrow s_1(X, X') \wedge \text{newt}_1(X') \\
 \vdots \\
 \text{newq}_m(X) \leftarrow s_m(X, X') \wedge \text{newt}_m(X') \\
 \\
 \text{newu}_1(X) \leftarrow v_1(X) \\
 \vdots \\
 \text{newu}_n(X) \leftarrow v_n(X)
 \end{array}
 \quad \begin{array}{l}
 \text{newp}_i, \text{newq}_i, \text{newt}_i, \text{newu}_i \\
 \text{not necessarily distinct}
 \end{array}$$

SpecVar: **Var, enumerated** $x_p \{ \text{newp}_1, \text{newq}_1, \text{newt}_1, \text{newu}_1, \dots \}$

SpecI: $(\text{init}_1(X) \wedge x_p = \text{newp}_1) \vee \dots \vee$
 $(\text{init}_k(X) \wedge x_p = \text{newp}_k)$

SpecT: $(x_p = \text{newq}_1 \wedge s_1(X, X') \wedge x_p = \text{newt}_1) \vee \dots \vee$
 $(x_p = \text{newq}_m \wedge s_m(X, X') \wedge x_p = \text{newt}_m)$

SpecU: $(v_1(X) \wedge x_p = \text{newu}_1) \vee \dots \vee$
 $(v_n(X) \wedge x_p = \text{newu}_n)$

the specialized system is:

$\langle \text{SpecVar}, \text{SpecI}, \text{SpecT}, \text{SpecU} \rangle$

Experiments

EXAMPLES	default		A		F	
	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
<i>No. of verified properties</i>	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)

EXAMPLES	default		A		F	
	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
<i>No. of verified properties</i>	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: $P_S \mapsto P_1 \mapsto \dots \mapsto P_n = \text{Spec}P_S$

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}

- T:
- | | | |
|-----|---|--|
| (1) | $\langle X, Y \rangle \longrightarrow \langle X, Y-1 \rangle$ | if $X \geq 1$ |
| (2) | $\langle X, Y \rangle \longrightarrow \langle X, Y+2 \rangle$ | if $X \leq 2$ |
| (3) | $\langle X, Y \rangle \longrightarrow \langle X, -1 \rangle$ | if $\exists V \in \mathbb{Z} \ Y = 2V+1$ |

Relaxing System Semantics

Counter System: interpreted on \mathbb{Z}

- T:
- (1) $\langle X, Y \rangle \longrightarrow \langle X, Y-1 \rangle$ if $X \geq 1$
 - (2) $\langle X, Y \rangle \longrightarrow \langle X, Y+2 \rangle$ if $X \leq 2$
 - (3) $\langle X, Y \rangle \longrightarrow \langle X, -1 \rangle$ if $\exists V \in \mathbb{Z} \ Y = 2V+1$

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}

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

to show:

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

relax:

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

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*:

$$\text{unsafe} \in M(P_S) \quad \text{iff} \quad \text{unsafe} \in M(\text{Spec}P_S)$$

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!

MAP - Specialization-Based Reachability Analysis of Infinite-State Transition Systems

1. Program Uploading	2. Options Selection	3. Specialization	4. Perfect Model
<pre> % Bakery Protocol 2 processes - safety [Delzanno-Pedolski, 2001] % % Transitions t((t, A, S, B), s((v, D, S, B))) :- D=0, B=0, B=0. t((v, A, S, B), s((u, A, S, B))) :- B=0, A=0. t((v, A, S, B), s((u, A, S, B))) :- B=0, A=0. t((u, A, S, B), s((t, D, S, B))) :- D=0, A=0, B=0. t((S, A, T, B), s((S, A, W, D))) :- D=0, A=0, B=0. t((S, A, W, B), s((S, A, U, B))) :- B=0, B=0. t((S, A, U, B), s((S, A, T, D))) :- D=0, B=0, A=0. % Elementary Properties eElem((u, A, u, B), unsafe) :- A=0, B=0. eElem((t, A, t, C), initial) :- A=0, C=0. eElem((v, A, v, A), initial) :- A=0. % Temporal Properties lev1 :- unreachable(backward, initial, unsafe). </pre>		<p>Specialization Options:</p> <p>Invariant: <input type="text" value="inv1"/></p> <p>Timeout: <input type="text" value="10.5"/></p> <p><input type="radio"/> Default <input checked="" type="radio"/> Custom</p> <p>Generalization Parameters:</p> <p>MaxCoeff: <input type="text" value="off"/></p> <p>Firing Relation: <input type="text" value="variant"/></p> <p>Gen. Oper.: <input type="text" value="ariden"/></p> <p>Gen. Param.: <input type="text" value="e_beq_maximum"/></p> <p>Polyvariance Parameters:</p> <p>Partitioning: <input type="text" value="single"/></p> <p>Include Foldable: <input type="text" value="include"/></p> <p>Candidate: <input type="text" value="w.r.t. ancestor"/></p> <p>Post-Folding: <input type="text" value="most general"/></p>	
Specialize		Help	

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