

A Technique for Invariant Generation

Ashish Tiwari, Harald Rueß, Hassen Saidi, N. Shankar

SRI International

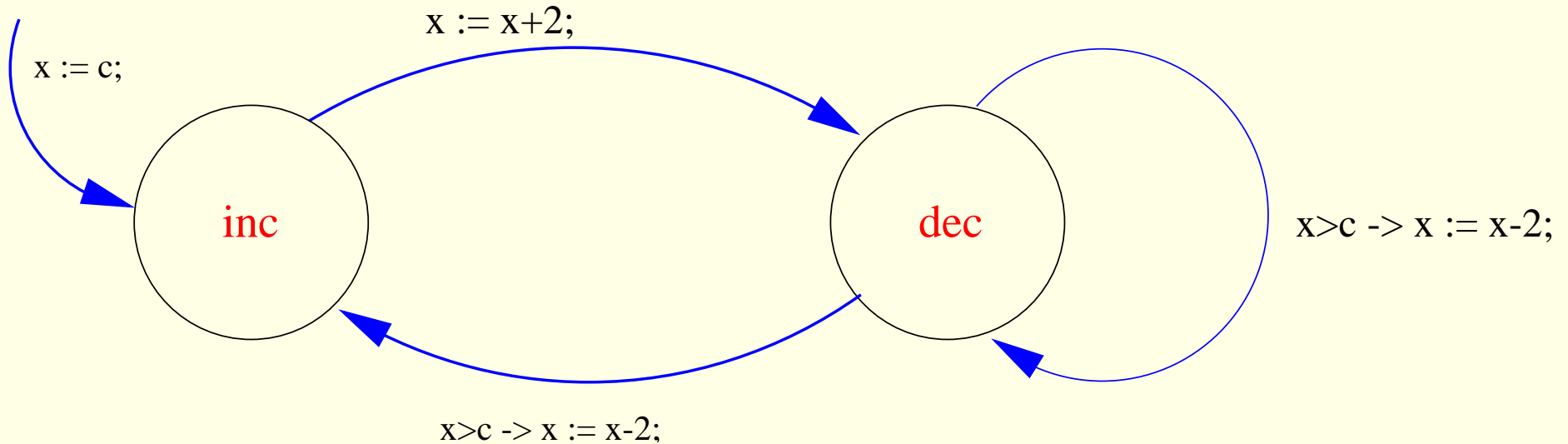
<mailto:ruess@csl.sri.com>

March 30, 2001

Overview

- Motivation
- Concepts of Invariants
- Symbolic Simulation
- Widening and Narrowing
- Algorithm
- Example
- Failure-Tolerant Theorem Proving

A Simple Program: Twos



Property. x is either c or $c + 2$.

Model-checking not applicable, since state space is infinite.

Classical proof fails, since property is not inductive.

Objective. Automatic generation of inductive invariants for infinite state systems.

Logical Transition System

States

$$\begin{aligned}x &\in \text{int} \\pc &\in \{\text{inc}, \text{dec}\}\end{aligned}$$

Initial Predicate

$$\text{Init}([x, pc]) := (pc = \text{inc} \wedge x = c)$$

Next state relation

$$\text{Next}([x, pc], [x', pc']) := \left(\begin{array}{l} pc = \text{inc} \\ \wedge x' = x + 2 \\ \wedge pc' = \text{dec} \end{array} \right) \vee \left(\begin{array}{l} pc = \text{dec} \\ \wedge x > c \\ \wedge x' = x - 2 \end{array} \right)$$

Invariants

Strongest postcondition

$$SP(\phi)(\vec{y}) \quad := \quad \exists \vec{x}. \text{Next}(\vec{x}, \vec{y}) \wedge \phi(\vec{x})$$

Reachable States

$$\text{Reach}(\vec{x}) \quad := \quad \mu \phi. \text{Init}(\vec{x}) \vee SP(\phi)(\vec{x})$$

Invariants

$$\text{Reach}(\vec{x}) \quad \Longrightarrow \quad \phi(\vec{x})$$

Inductive Invariants

$$(\text{Init}(\vec{x}) \vee SP(\phi)(\vec{x})) \quad \Longrightarrow \quad \phi(\vec{x})$$

Upward Iteration

- Twos example

$$\psi_0([x, pc]) = \text{false}$$

$$\begin{aligned}\psi_1([x, pc]) &= \text{Init}([x, pc]) \vee SP(\psi_0)([x, pc]) \\ &= pc = inc \wedge x = c\end{aligned}$$

$$\begin{aligned}\psi_2([x, pc]) &= \text{Init}([x, pc]) \vee SP(\psi_1)([x, pc]) \\ &= \text{Init}([x, pc]) \vee \exists \bar{x}, \bar{pc}. \bar{pc} = inc \wedge \bar{x} = c \wedge \text{Next}([\bar{x}, \bar{pc}], [x, pc]) \\ &= \text{Init}([x, pc]) \vee \text{Next}([c, inc], [x, pc]) \\ &= (pc = inc \wedge x = c) \vee (pc = dec \wedge x = c + 2)\end{aligned}$$

$$\begin{aligned}\psi_3([x, pc]) &= \text{Init}([x, pc]) \vee SP(\psi_2)([x, pc]) \\ &= \text{Init}([x, pc]) \vee \text{Next}([c, inc], [x, pc]) \vee \text{Next}([c + 2, dec], [x, pc]) \\ &= \psi_2([x, pc]) \vee (pc = dec \wedge x = c)\end{aligned}$$

- Terminates, since $\psi_4 \implies \psi_3$.
- But usually nonterminating.

Downward Iteration

Iteration.

$$\begin{aligned}\phi_0 &:= true \\ \phi_{i+1} &:= Init \vee SP(\phi_i)\end{aligned}$$

Lemma. Every formula ϕ_i is an inductive invariant.

Two example. $x > c - 2$

Experience. Usually terminates but with (very) weak invariants.

Decomposition. Every ϕ_i can be decomposed as $\psi_i \vee \chi_i$ s.t.

$$\begin{aligned}\psi_0 &= false & \chi_0 &= true \\ \psi_{i+1} &= Init \vee SP(\psi_i) & \chi_{i+1} &= SP(\chi_i)\end{aligned}$$

Approximations. ψ_i under-approxs and ϕ_i over-approxs. of *Reach*

Widening and Narrowing

- Upward iteration with widening

$$\begin{aligned} \psi_0 &:= \textit{Init} \\ \psi_{i+1} &:= \begin{cases} \psi_i \vee SP(\psi_i) \\ \text{or} \\ \psi_i \vee \alpha_i \end{cases} \end{aligned}$$

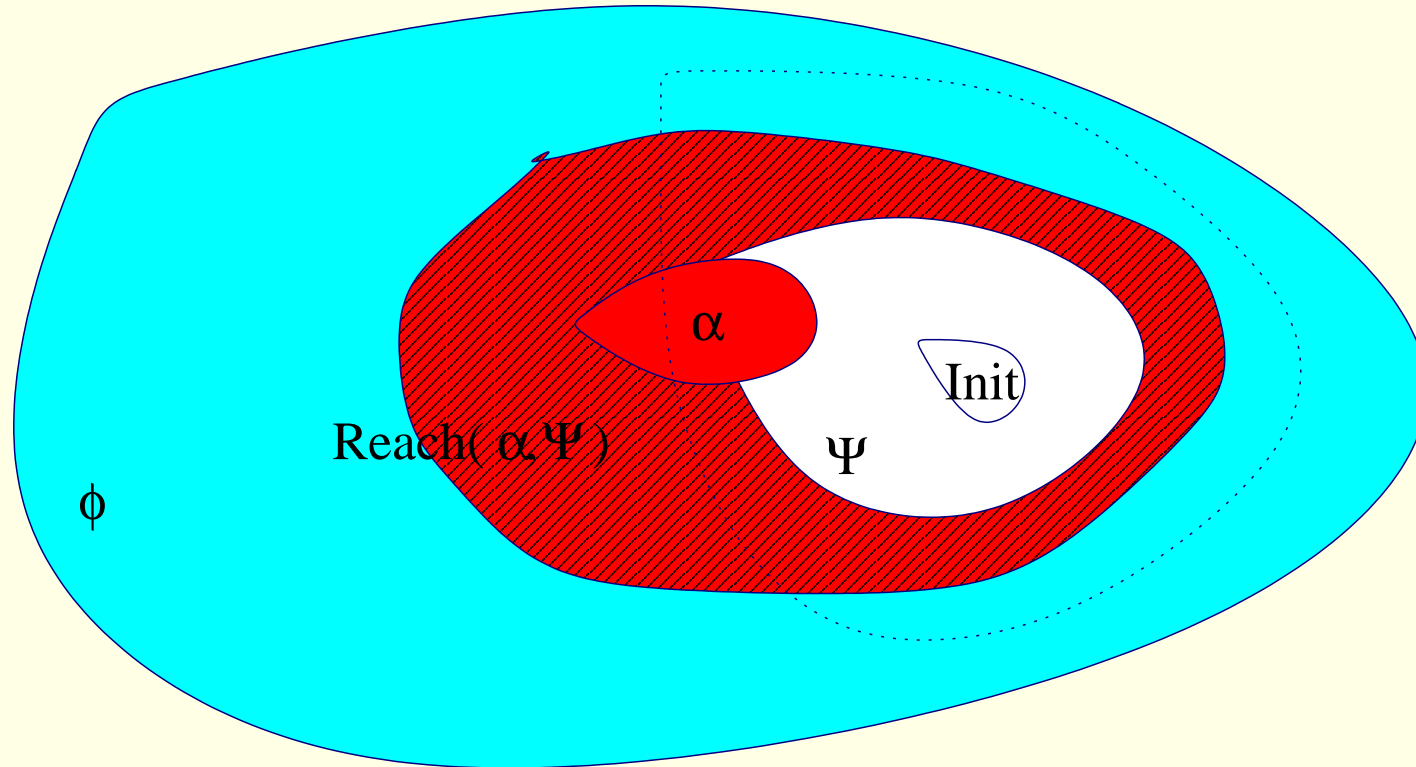
- A ψ_n obtained by propagation is inductive if $\psi_n \implies \psi_{n-1}$.
- Downward iteration with narrowing.

$$\begin{aligned} \phi_0 &:= \textit{true} \\ \phi_{i+1} &:= \begin{cases} \textit{Init} \vee SP(\phi_i) \\ \text{or} \\ \phi_i \wedge \beta_i \quad \text{where } \beta_i \text{ is inductive.} \end{cases} \end{aligned}$$

- Every ϕ_i is inductive.

Inductive Invariants for Narrowing (I)

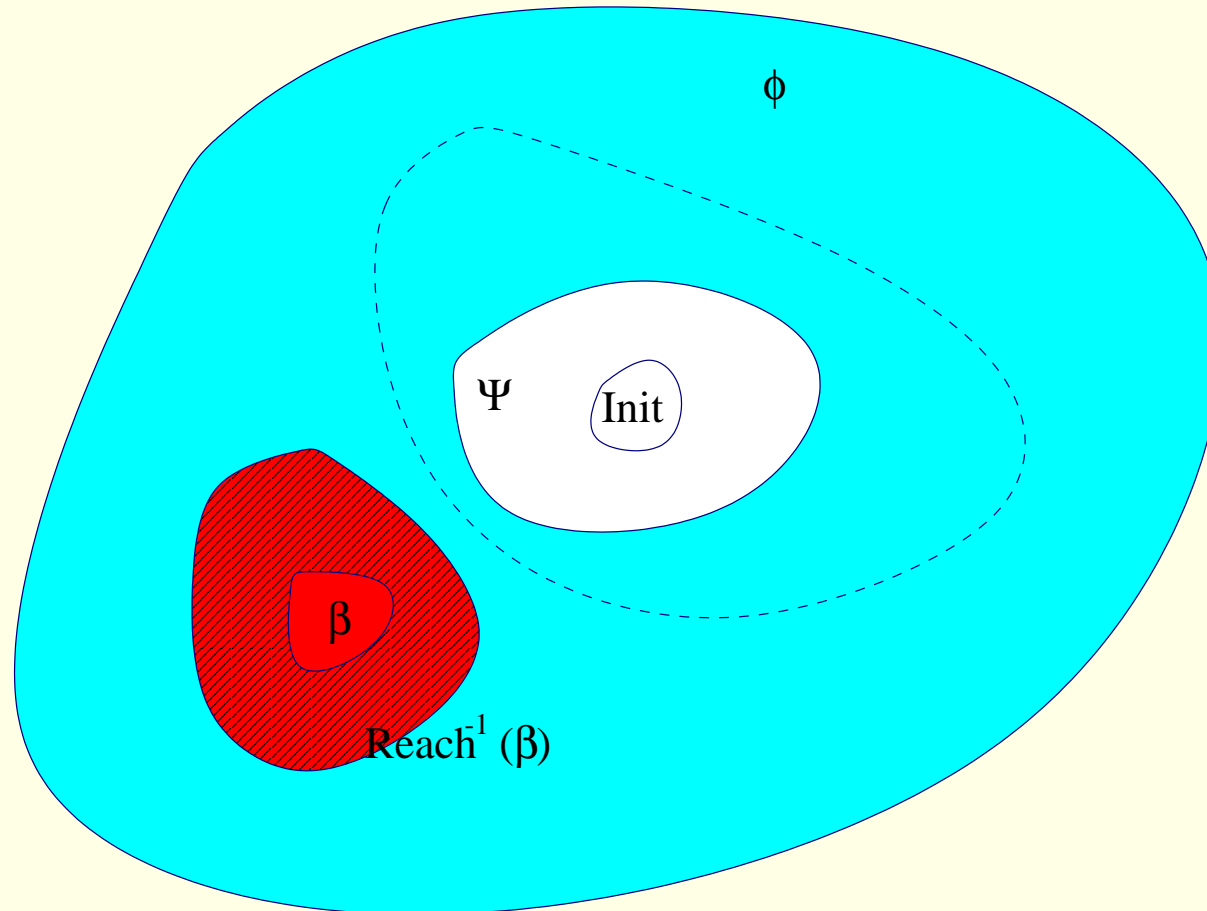
Choose α in ϕ .



In case forward propagation terminates, new invariant has been found (may not necessarily be contained in *Reach*).

Inductive Invariants for Narrowing (II)

Choose β in ϕ and outside of ψ .



If $Reach^{-1}(\beta) \wedge Init$ is unsatisfiable, then $\neg Reach^{-1}(\beta)$ is inductive.

Outline of Algorithm

Keep an underapproximation ψ and an overapproximation ϕ of the reachable state set.

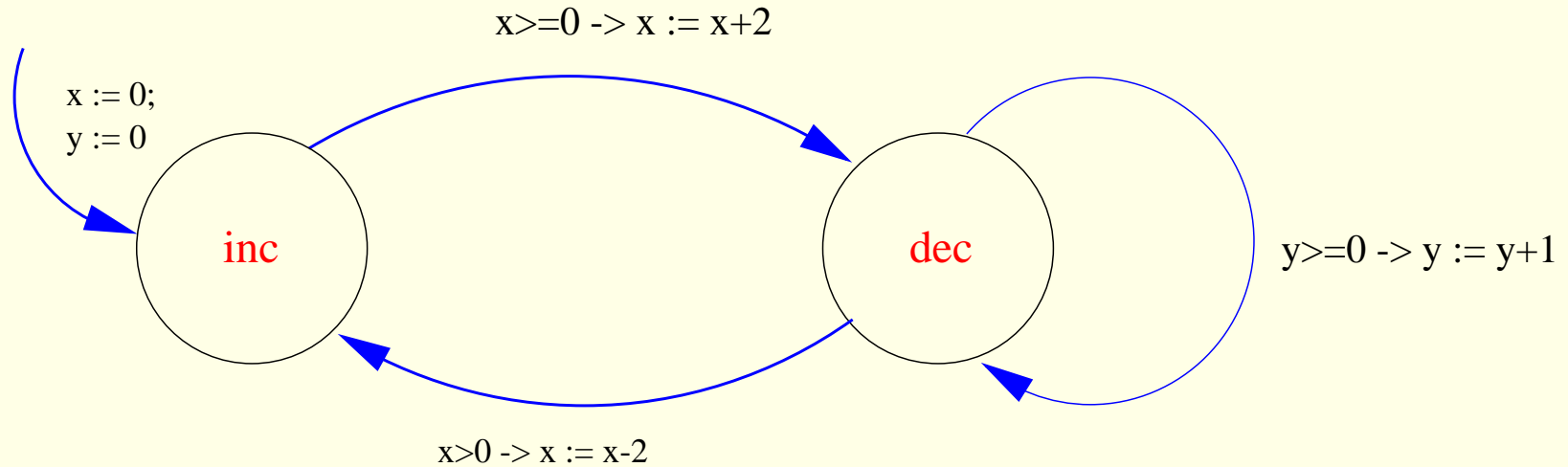
Iteratively improve these approximations by

- Upward iteration of ψ .
- Downward iteration of ϕ .
- Narrowing of upper bound with inductive invariants. These invariants are obtained by
 - Forward simulation of ψ widened with some disjunct from ϕ .
 - Backward simulation of a disjunct of ϕ , which is considered to be unreachable.

When upward iteration of ψ terminates, this limit is returned as the invariant.

Algorithm can be stopped at any time with the upper bound as the currently best invariant.

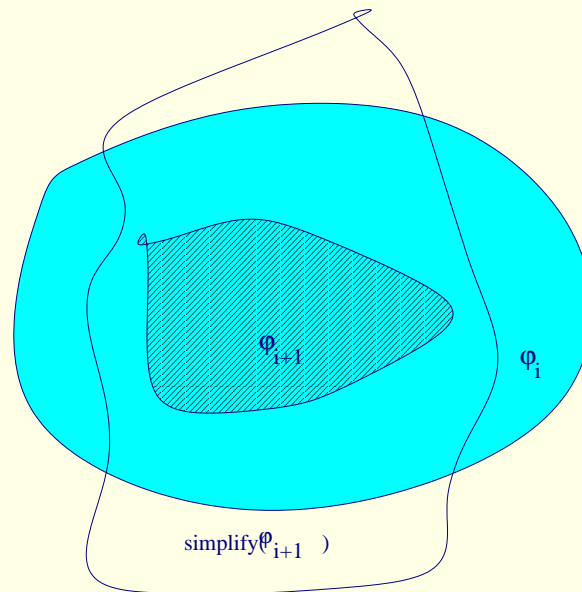
Another Simple Program



	$pc = inc$		$pc = dec$	
	ψ_i	χ_i	ψ_i	χ_i
P	$false$	$true$	$false$	$true$
P	$x = 0 \wedge y = 0$	$x > -2$	$false$	$x \geq 2 \wedge y \geq 1$
P	$x = 0 \wedge y = 0$	$x \geq 0 \vee (x > -2 \wedge y \geq 1)$	$x = 2 \wedge y = 0$	$x = 2 \vee x > 2 \vee (x \geq 2 \vee y \geq 1) \vee y \geq 2$
N	$x = 0 \wedge y = 0$	$x = 0 \vee (-2 < x \leq 0 \wedge y \geq 1)$	$x = 2 \wedge y = 0$	$x = 2 \vee x = 2 \wedge y \geq 1 \vee (x \geq 2 \wedge y \geq 2)$
W	$x = 0 \wedge y = 0$	$x = 0 \wedge y \geq 1$	$x = 2 \wedge y = 0$	$x = 2 \wedge y \geq 1$

Simplification and Quantifier Elimination

- Assume a weakening simplifier: $\phi \implies \text{simplify}(\phi)$
- Upward iteration: Widening ψ_i to $\psi_i \vee \text{simplify}(\psi_{i+1})$.
- Downward iteration: Narrowing ϕ_i to $\phi_i \wedge \text{simplify}(\phi_{i+1})$



$\phi_i \wedge \text{simplify}(\phi_{i+1})$ is an inductive invariant.

- Example: weaken $\exists x. p(x) \wedge q(x)$ as $\exists x. p(x) \wedge \exists x. q(x)$.
- Failure-tolerant theorem proving.

Conclusions

- Automated invariant generation technique for infinite-state systems.
- Simultaneous computation of under- and overapproximations of reachable state set.
- Narrowing for refining over-approximation of the reachable state set.
- Anytime algorithm.
- Failure-tolerant theorem proving.