Proving Termination by Policy Iteration

Damien Massé

LabSTICC
Université de Bretagne Occidentale
Brest, France

NSAD 2012

Motivation

Using policy iteration to prove termination.

1. Why?

   (termination and fixpoint approximation)

2. How?

   (a simple application)

Termination

We are looking for sufficient conditions for definite termination.

Definite termination

Given a program represented by a transition system (Σ, τ), initial states $I \subseteq \Sigma$, the program definitely terminates from an initial state i if every computation from i terminates.

We want $\mathcal{T}_I \subseteq I$ such that the program definitely terminates from all elements of $\mathcal{T}_I$

Sufficient conditions $\rightarrow$ needed to prove termination.

Proving termination

Common method: ranking function $r \in \Sigma \to O$:

$$\forall \sigma, \sigma' \in \mathrm{Reach}(\mathcal{T}_I), \sigma \xrightarrow{\tau} \sigma' \implies r(\sigma') < r(\sigma)$$

Termination can also be expressed using fixpoint semantics (here state-based):

- with the set of (definitely) terminating states

$$\mathcal{T}_I \subseteq \mathrm{lfp}\,\widetilde{\mathrm{pre}} \text{ where } \widetilde{\mathrm{pre}}(X) = \{\sigma | \forall \sigma \xrightarrow{\tau} \sigma', \sigma' \in X\}$$

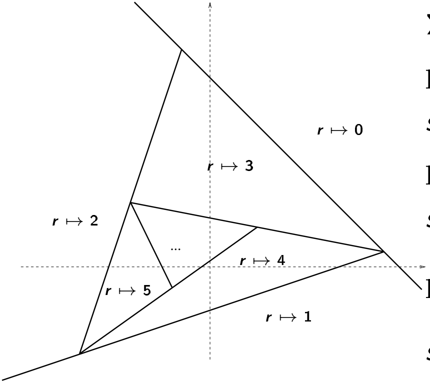
- with the set of (potentially) non-terminating states

$$\mathcal{T}_I \cap \mathrm{gfp}\,\mathrm{pre} = \emptyset \text{ where } \mathrm{pre}(X) = \{\sigma | \exists \sigma' \in X, \sigma \xrightarrow{\tau} \sigma'\}$$

The iterates of these fixpoints give a ranking function.

Example

real x,y;
while (x+y<=10) { x=-2y // y=x-y+3; }



$\Sigma = \mathbb{R}^2$

$\mathrm{pre}(\Sigma) : x + y \leq 10$

$s \notin \mathrm{pre}(\Sigma) \Rightarrow r(s) = 0$

$\mathrm{pre}^2(\Sigma) : x + y \leq 10 \wedge -3y + x \leq 7$

$s \in \mathrm{pre}(\Sigma) \setminus \mathrm{pre}^2(\Sigma) \Rightarrow r(s) = 1$

$$\mathrm{pre}^3(\Sigma) : \begin{array}{l} x + y \leq 10 \wedge -3y + x \leq 7 \\ \wedge -3x + y \leq 16 \end{array}$$

$s \in \mathrm{pre}^2(\Sigma) \setminus \mathrm{pre}^3(\Sigma) \Rightarrow r(s) = 2$

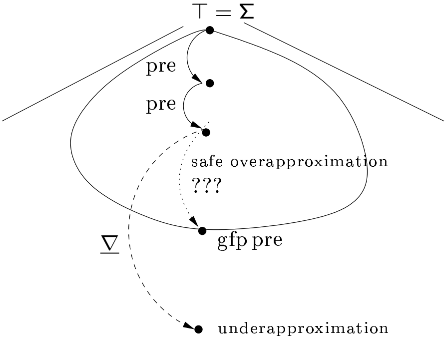
...

Abstract fixpoint

To get *sufficient conditions*, you need either:

- to *underapproximate* the least fixpoint;
- to *overapproximate* the greatest fixpoint.

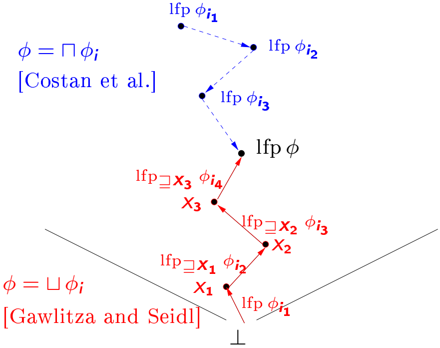
We want to use abstractions ⇒ choose the gfp.



We cannot use widenings.

Policy iteration

Policy/strategy iteration techniques have been used to compute *exact* (abstract) fixpoints.
Two approaches, from Costan et al [CAV'05], or Gawlitza and Siedl [CSL'07].



The approach from below is more appropriate:

- It guarantees to reach the least fixpoint.
- And any intermediate result is correct.

Algorithm (for gfp)

Suppose $\phi = \sqcap\{\phi_i\}$, ϕ_i are the *strategies* such that $\forall x, \exists i, \phi(x) = \phi_i(x)$.
The algorithm has two steps, given an initial postsolution $x = \top$:

1. Strategy selection: select ϕ_i such that $\phi_i(x) = \phi(x)$.
2. Strategy solving: compute $x = \mathrm{gfp}_{\sqsubseteq x}\,\phi_i$. Stop if $x = \phi(x)$.

Two questions:

1. does the algorithm terminate (and returns $\mathrm{gfp}\,\phi$)?
   Yes, under some conditions (e.g. every strategy is selected at most once).
2. can we compute $\mathrm{gfp}_{\sqsubseteq x}\phi_i$?
   Yes, under some conditions (e.g. x is *consistent* w.r.t. ϕ_i).

We can only use this method on specific classes of programs and abstract domains.

Affine programs

An affine program is defined by $(N, E, \mathbf{st})$ where

- N is the finite set of program points;
- $E \subseteq N \times \mathbf{Stmt} \times N$ transitions labeled by *statements*;
- $\mathbf{st}$ initial program point.

Statements are pairs of the form $(g; a)$ such that:

- g is an affine guard $A\mathbf{x} + b \geq 0$ on the program variables $\mathbf{x}$
- a is an affine assignment $\mathbf{x} := A\mathbf{x} + b$.

Template polyhedral domain

Abstraction of $\wp(\mathbb{R}^n)$ relative to a template constraint matrix $T \in \mathbb{R}^{m\times n}$:

$$\wp(\mathbb{R}^n) \xrightleftharpoons[\alpha_T]{\gamma_T} (\mathbb{R} \cup \{-\infty, +\infty\})^m$$

with $\gamma_T(\rho) = \{x \in \mathbb{R}^n \mid Tx \leq \rho\}$.

Example: octagons with two variables: $T = \begin{pmatrix} 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \\ 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{pmatrix}$

$\rightarrow$ 8 “abstract” variables (C_y, C_{-y}, $\ldots$).

Abstract (forward) semantics

The abstract semantics of an affine program can be expressed as the least solution of a system of equations of the form $C_v := e$ with:

$$e \ ::= \ a \mid C_w \mid e + e \mid b \cdot e \mid e \vee e \mid e \wedge e \mid \mathrm{LP}_{A,b}(e, \ldots, e)$$

$\mathrm{LP}_{A,b}$ denotes a linear program:

$$\mathrm{LP}_{A,b}(x_1, \ldots, x_m) = \max\{b^T y | y \in \mathbb{R}^n, Ay \leq x\}$$

This is a system of rational equations with linear programs.

Strategy selection and solving

A strategy associates each $\vee$-formula to one of its subformula.
The application of a strategy gives a system of conjunctive equations with linear programs:

$$e ::= a \mid C_w \mid e + e \mid b \cdot e \mid e \wedge e \mid \mathrm{LP}_{A,b}(e, \ldots, e)$$

Although LPs can be treated as the minimum of several linear expressions, they are dealt with by adding new variables and constraints.

Results

- Once the strategy is selected, the fixpoint can be computed by solving two linear programs.
- Each strategy is selected at most once, the algorithm terminates.

Abstract backward semantics

Abstract backward semantics: $\mathrm{gfp}\, \alpha_T \circ \mathrm{pre} \circ \gamma_T$.

Proposition

The abstract backward semantics of the affine program is the greatest solution of a system of equations on $\mathbf{C}$ of the form:

$$C_v := U_1 \vee U_2 \vee \ldots \vee U_k \text{ with } U_i := \phi_i \wedge \psi_i$$

where

- ϕ_i is of the form (`if` $\{\mathbf{y} | A\mathbf{y} + b \leq \mathbf{C}\} \neq \emptyset$ `then` ∞ `else` $-\infty$)
- ψ_i is a linear program, which can be expressed as:

$$\psi_i = \bigwedge \{\lambda^T \cdot (\mathbf{C} - b) | \lambda \geq 0 \wedge A^T \lambda = V\}$$

Example

real x,y;
while (x+y<=10) { x=-2y // y=x-y+3; }

$C_x = \phi \wedge \psi$ with

- $\phi = -\infty$ iff the set of constraints $\{\ x+y-10 \leq 0,\ x-y+3 \leq C_y,\ -x+y-3 \leq C_{-y},\ -2y \leq C_x,\ 2y \leq C_{-x},\ x-3y+3 \leq C_{x+y},\ -x-y-3 \leq C_{x-y},\ x+y+3 \leq C_{-x+y},\ -x+3y-3 \leq C_{-x-y}\}$ is unsatisfiable.
- $$\begin{aligned}\psi = \min\{&10\lambda_0 + \lambda_1(C_y-3) + \lambda_2(C_{-y}+3) + \lambda_3 C_x + \lambda_4 C_{-x} + \lambda_5(C_{x+y}-3)\\ &+\lambda_6(C_{x-y}+3) + \lambda_7(C_{-x+y}-3) + \lambda_8(C_{-x-y}+3)\\ &|\boldsymbol{\lambda} \geq 0 \wedge \lambda_0 + \lambda_1 - \lambda_2 + \lambda_5 - \lambda_6 + \lambda_7 - \lambda_8 = 1\\ &\wedge \lambda_0 - \lambda_1 + \lambda_2 - 2\lambda_3 + 2\lambda_4 - 3\lambda_5 - \lambda_6 + \lambda_7 + 3\lambda_8 = 0\}\end{aligned}$$

Strategy construction

Vertex principle of linear programming

ψ_i is the minimum of a finite set of affine expressions, each one being related to an optimal solution of the linear program.

1. Select between ϕ_i and ψ_i.
   - if ϕ_i evaluates to ∞, select ϕ_i
   - otherwise, replace the expression by $-\infty$.
2. Extract an affine expression from ϕ_i.
   - Computing at once all the affine expressions is costly.
   - So we can compute the affine expressions lazily.

Example

$$\psi = \min\{10\lambda_0 + \lambda_1(C_y - 3) + \lambda_2(C_{-y} + 3) + \lambda_3 C_x + \lambda_4 C_{-x} + \lambda_5(C_{x+y} - 3)$$
$$+\lambda_6(C_{x-y} + 3) + \lambda_7(C_{-x+y} - 3) + \lambda_8(C_{-x-y} + 3)$$
$$|\boldsymbol{\lambda} \geq 0 \wedge \lambda_0 + \lambda_1 - \lambda_2 + \lambda_5 - \lambda_6 + \lambda_7 - \lambda_8 = 1$$
$$\wedge \lambda_0 - \lambda_1 + \lambda_2 - 2\lambda_3 + 2\lambda_4 - 3\lambda_5 - \lambda_6 + \lambda_7 + 3\lambda_8 = 0\}$$

With $C_{x+y} = 10$ and $C_x = C_{-x} = \ldots = C_{-x-y} = +\infty$, the optimal solution is:

$$\lambda_5 = 0.25 \quad \lambda_0 = 0.75 \quad \lambda_i = 0 \text{ for } i \notin \{0, 5\}$$

which gives the affine expression:

$$6.75 + 0.25\, C_{x+y}$$

We replace ψ by this expression.

Strategy

The strategy selection step gives a system of disjunctive equations.

Results

Strategy solving

Once the strategy is constructed, its solution ($\leq$ a consistent postsolution) can be computed by solving to linear programs extractable from the system in linear time.

Strategy improvement

The strategy improvement operator preserves the consistency of the postsolution.

Final result

The algorithm terminates and returns the abstract semantics $\mathrm{gfp}\ \mathit{pre}^\sharp$.

The number of iterations may be exponential (we expect it to remain low in practice). However, any intermediate result is a safe overapproximation.

Example

real x,y;
while (x+y<=10) { x=-2y // y=x-y+3; }

#	Strategy	Solution
1	$C_{x+y} = 10$	$x + y \leq 10$
2	$C_x = 6.75 + 0.25 C_{x+y}$, $C_{x+y} = 10$, $C_{x-y} = 3.5 + C_{x+y}/2$	$x \leq 9.25$, $x + y \leq 10$ $x - y \leq 8.5$
3	$C_x = 6.75 + 0.25 C_{x+y}$, $C_{x+y} = 10$, $C_{x-y} = 3.5 + C_{x+y}/2$, $C_{-y} = 0.5 C_x$, $C_{-x-y} = 3 + C_{x-y}$	$x \leq 9.25$, $-4.625 \leq y$ $-11.5 \leq x + y \leq 10$ $x - y \leq 8.5$
4	$C_x = 6.75 + 0.25 C_{x+y}$, $C_{x+y} = 10$, $C_{x-y} = 3.5 + C_{x+y}/2$, $C_{-y} = 0.5 C_x$, $C_{-x-y} = 3 + C_{x-y}$, $C_y = 3.25 + 0.25 C_{-x-y}$ $C_{y-x} = 3 + C_{-y}$, $C_{-x} = 3 + 0.5 C_{-x-y} + 0.5 C_{-y}$	$-9.5625 \leq x \leq 9.25$ $-4.625 \leq y \leq 6.125$ $-11.5 \leq x + y \leq 10$ $-7.625 \leq x - y \leq 8.5$
5	$C_x = -3 + 0.5 C_{-x-y} + 0.5 C_y$, $C_{x+y} = -3 + C_{-x+y}$, $C_{x-y} = -3 + C_y$ $C_{-y} = 0.5 C_x$, $C_{-x-y} = 3 + C_{x-y}$, $C_y = 0.5 C_{-x}$, $C_{y-x} = 3 + C_{-y}$, $C_{-x} = 3 + 0.5 C_{-x-y} + 0.5 C_{-y}$	$x = -1.5$, $y = 0.75$

The program terminates from any state $\neq (-1.5, 0.75)$.

Discussion on ranking functions

Our method computes exactly the abstract semantics, i.e.:

$$S = \mathrm{gfp}\,(\rho_T \circ \mathrm{pre}) \text{ where } \rho_T = \gamma_T \circ \alpha_T$$

The iterates give a ranking function r on $\Sigma \setminus S$, where $S \cup r(n\uparrow) \in \mathrm{Im}(\rho_T)$. Conversely, if a ranking function of this form exists, our method proves the termination.

Theorem

Our approach proves the termination on $\Sigma \setminus Z$ with the template matrix T if and only if there exists a ranking function r such that $\{r(n\uparrow) \cup Z\} \subseteq \mathrm{Im}(\gamma_T)$.

Hence, if the program admits a linear ranking function $x \mapsto Vx$, we can prove the termination if $-V$ is a row of T.

Conclusion

First attempt to use policy iteration for termination properties.

Improvements

- Non-determinism.
- Incremental construction of the template matrix.
- Other weakly relational domains (previous work).

Future work

- Comparison with other methods.
- Mixing with other methods.