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 $(\Sigma, \tau)$, 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 $r : \Sigma \rightarrow O$:

$$\forall \sigma, \sigma' \in \text{Reach}(T_I), \sigma \xrightarrow{\tau} \sigma' \implies r(\sigma') < r(\sigma)$$

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

- with the set of (definitely) terminating states

  $$T_I \subseteq \text{lfp } \tilde{\text{pre}} \text{ where } \tilde{\text{pre}}(X) = \{ \sigma | \forall \sigma \xrightarrow{\tau} \sigma', \sigma' \in X \}$$

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

  $$T_I \cap \text{gfp } \text{pre} = \emptyset \text{ where } \text{pre}(X) = \{ \sigma | \exists \sigma' \in X, \sigma \xrightarrow{\tau} \sigma' \}$$

The iterates of these fixpoints give a ranking function.
Example

\[
\begin{align*}
  \text{real } x, y; \\
  \text{while } (x+y \leq 10) \{ \ &x=-2y \ // \ y=x-y+3; \} \\
\end{align*}
\]

\[
\Sigma = \mathbb{R}^2 \\
\text{pre} (\Sigma) : x + y \leq 10 \\
\text{pre}^2 (\Sigma) : x + y \leq 10 \land -3y + x \leq 7 \\
\text{pre}^3 (\Sigma) : x + y \leq 10 \land -3y + x \leq 7 \land -3x + y \leq 16
\]

\[
\forall s \in \text{pre} (\Sigma) \setminus \text{pre}^2 (\Sigma) \Rightarrow r(s) = 1 \\
\forall s \in \text{pre}^2 (\Sigma) \setminus \text{pre}^3 (\Sigma) \Rightarrow r(s) = 2
\]

\[
\ldots
\]
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].

φ = \bigcap φ_i

[Costan et al.]

φ = \bigcup φ_i

[Gawlitza and Seidl]

The approach from below is more appropriate:

- It guarantees to reach the least fixpoint.
- And any intermediate result is correct.
Algorithm (for gfp)

Suppose $\phi = \bigcap\{\phi_i\}$, $\phi_i$ 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 $\phi_i$ such that $\phi_i(x) = \phi(x)$.
2. **Strategy solving**: compute $x = \text{gfp}_{\subseteq x} \phi_i$. Stop if $x = \phi(x)$.

Two questions:

1. does the algorithm terminate (and returns gfp $\phi$)?
   
   Yes, under some conditions (e.g. every strategy is selected at most once).

2. can we compute $\text{gfp}_{\subseteq x} \phi_i$?
   
   Yes, under some conditions (e.g. $x$ is consistent w.r.t. $\phi_i$).

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

An affine program is defined by \((N, E, st)\) where

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

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

- \(g\) is an affine guard \(Ax + b \geq 0\) on the program variables \(x\)
- \(a\) is an affine assignment \(x := Ax + b\).
Template polyhedral domain

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

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

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

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:

$$e := a | C_w | e + e | b \cdot e | e \lor e | e \land e | \text{LP}_{A,b}(e, \ldots, e)$$

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

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

This is a system of rational equations with linear programs.
Strategy selection and solving

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

\[
e ::= a \mid C_w \mid e + e \mid b \cdot e \mid e \land e \mid LP_{A,b}(e, \ldots, e)
\]

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: \( \text{gfp } \alpha_T \circ \text{pre} \circ \gamma_T \).

Proposition

The abstract backward semantics of the affine program is the greatest solution of a system of equations on \( C \) of the form:

\[
C_v := U_1 \lor U_2 \lor \ldots \lor U_k \text{ with } U_i := \phi_i \land \psi_i
\]

where

- \( \phi_i \) is of the form (if \( \{y | Ay + b \leq C\} \neq \emptyset \) then \( \infty \) else \( -\infty \))
- \( \psi_i \) is a linear program, which can be expressed as:

\[
\psi_i = \bigwedge \{\lambda^T \cdot (C - b) | \lambda \geq 0 \land A^T\lambda = V\}
\]
Example

\[\text{real } x, y;\]
\[\text{while } (x+y \leq 10) \{ x = -2y // y = x - y + 3; \}\]

\[C_x = \phi \land \psi \text{ with}\]

- \(\phi = -\infty\) 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.

- \(\psi = \min\{10\lambda_0 + \lambda_1(C_y - 3) + \lambda_2(C_{-y} + 3) + \lambda_3C_x + \lambda_4C_{-x} + \lambda_5(C_{x+y} - 3) + \lambda_6(C_{x-y} + 3) + \lambda_7(C_{-x+y} - 3) + \lambda_8(C_{-x-y} + 3) | \lambda \geq 0 \land \lambda_0 + \lambda_1 - \lambda_2 + \lambda_5 - \lambda_6 + \lambda_7 - \lambda_8 = 1 \land \lambda_0 - \lambda_1 + \lambda_2 - 2\lambda_3 + 2\lambda_4 - 3\lambda_5 - \lambda_6 + \lambda_7 + 3\lambda_8 = 0\}\)
Strategy construction

Vertex principle of linear programming

\[ \psi_i \] 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 \( \phi_i \) and \( \psi_i \).
   - if \( \phi_i \) evaluates to \( \infty \), select \( \phi_i \)
   - otherwise, replace the expression by \(-\infty\).

2. Extract an affine expression from \( \phi_i \).
   - Computing at once all the affine expressions is costly.
   - So we can compute the affine expressions lazily.
Example

\[ \psi = \min \{ 10\lambda_0 + \lambda_1 (C_y - 3) + \lambda_2 (C_{-y} + 3) + \lambda_3 C_x + \lambda_4 C_{-x} + \lambda_5 (C_{x+y} - 3) \]
\[ + \lambda_6 (C_{x-y} + 3) + \lambda_7 (C_{-x+y} - 3) + \lambda_8 (C_{-x-y} + 3) \]
\[ |\lambda| \geq 0 \land \lambda_0 + \lambda_1 - \lambda_2 + \lambda_5 - \lambda_6 + \lambda_7 - \lambda_8 = 1 \]
\[ \land \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:

\[ 6.75 + 0.25C_{x+y} \]

We replace \( \psi \) by this expression.

Strategy

The strategy selection step gives a system of disjunctive equations.
Results

Strategy solving
Once the strategy is constructed, its solution (≤ 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 \( \text{gfp} \ pre^\# \).

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

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

<table>
<thead>
<tr>
<th>#</th>
<th>Strategy</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(C_{x+y} = 10)</td>
<td>(x + y \leq 10)</td>
</tr>
<tr>
<td>2</td>
<td>(C_x = 6.75 + 0.25C_{x+y}, C_{x+y} = 10, C_{x-y} = 3.5 + C_{x+y}/2)</td>
<td>(x \leq 9.25, x + y \leq 10) (x - y \leq 8.5)</td>
</tr>
<tr>
<td>3</td>
<td>(C_x = 6.75 + 0.25C_{x+y}, C_{x+y} = 10, C_{x-y} = 3.5 + C_{x+y}/2, C_{-y} = 0.5C_x, C_{-x-y} = 3 + C_{x-y})</td>
<td>(x \leq 9.25, -4.625 \leq y) (-11.5 \leq x + y \leq 10) (x - y \leq 8.5)</td>
</tr>
<tr>
<td>4</td>
<td>(C_x = 6.75 + 0.25C_{x+y}, C_{x+y} = 10, C_{x-y} = 3.5 + C_{x+y}/2, C_{-y} = 0.5C_x, C_{-x-y} = 3 + C_{x-y}, C_y = 3.25 + 0.25C_{-x-y})</td>
<td>(-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)</td>
</tr>
<tr>
<td>5</td>
<td>(C_x = -3 + 0.5C_{-x-y} + 0.5C_y, C_{x+y} = -3 + C_{-x+y}, C_{x-y} = -3 + C_y, C_{-y} = 0.5C_x, C_{-x-y} = 3 + C_{x-y}, C_y = 0.5C_{-x}, C_{y-x} = 3 + C_{-y}, C_{-x} = 3 + 0.5C_{-x-y} + 0.5C_{-y})</td>
<td>(x = -1.5, y = 0.75)</td>
</tr>
</tbody>
</table>

The program terminates from any state \(\neq (-1.5, 0.75)\).
Discussion on ranking functions

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

\[ S = \text{gfp} (\rho_T \circ \text{pre}) \text{ where } \rho_T = \gamma_T \circ \alpha_T \]

The iterates give a ranking function \( r \) on \( \Sigma \setminus S \), where \( S \cup r(n \uparrow) \in \text{Im}(\rho_T) \).

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 \( \{ r(n \uparrow) \cup Z \} \subseteq \text{Im}(\gamma_T) \).*

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.