Program Analysis with Local Policy Iteration

George Karpenkov

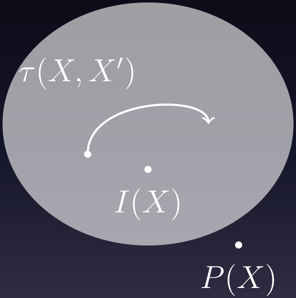
VERIMAG

September 28, 2015

Inductive Invariant

Motivation

- Task: verify programs, unreachability of error state
- Prove: by induction
- Finding separating inductive invariant
  - Includes initial state
  - Closed under transition
  - Separates bad state $P(X)$



Abstract Interpretation Limitations

- Usual tool: abstract interpretation
- Interpret the program in the abstract domain
- Use widening to enforce convergence

Abstract Interpretation Limitations

- Usual tool: abstract interpretation
- Interpret the program in the abstract domain
- Use widening to enforce convergence
  - $i \leq 1$
  - $i \leq 2$
  - $i \leq 3$
  - $\ldots$
  - $i < \infty$
- Loss of precision

Abstract Interpretation Limitations

- Usual tool: abstract interpretation
- Interpret the program in the abstract domain
- Use widening to enforce convergence
  - $i \leq 1$
  - $i \leq 2$
  - $i \leq 3$
  - $\ldots$
  - $i < \infty$
- Loss of precision
- Narrowing: very brittle

Policy Iteration

- Finds least inductive invariant in the given abstract domain
- Solves the equation an inductive invariant has to satisfy

Policy Iteration

- Finds least inductive invariant in the given abstract domain
- Solves the equation an inductive invariant has to satisfy
- Works only with certain domains
  - Template Constraints Domain
  - Upper bound on a fixed in advance set of linear expression
  - E.g. $\{x, y, x + y\}$

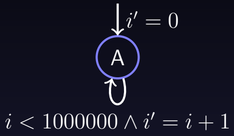
Policy Iteration

- Finds least inductive invariant in the given abstract domain
- Solves the equation an inductive invariant has to satisfy

- Works only with certain domains
  - Template Constraints Domain
  - Upper bound on a fixed in advance set of linear expression
  - E.g. $\{x, y, x + y\}$
  - Abstract semantics given using convex optimization
    $\max x'$ s.t. $x' = x + 1 \wedge x \leq 5$

Policy Iteration

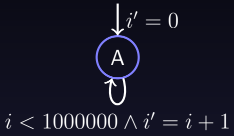
Simple Example



- Template constraints domain $\{i\}$
- Aim: find smallest d, s.t. $i \leq d$ is an inductive invariant
- Necessary and sufficient condition for d to be an inductive invariant:

Policy Iteration

Simple Example



- Template constraints domain $\{i\}$
- Aim: find smallest d, s.t. $i \leq d$ is an inductive invariant
- Necessary and sufficient condition for d to be an inductive invariant:
- $d = \sup i'$ s.t.
  $i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$
  - Disjunctions come from multiple edges
  - $\bot$ represents an unreachable state

Policy Iteration

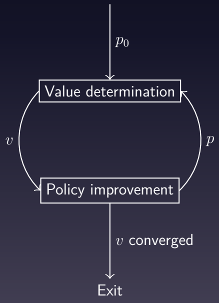
Algorithm Overview

- Solve by iterating over policies: choice of an argument per disjunction
- Find a value for policy using convex maximization

Policy Iteration

Algorithm Overview

- Solve by iterating over policies: choice of an argument per disjunction
- Find a value for policy using convex maximization



Policy $p \leftarrow p_0$
repeat
  $v \leftarrow$ value determination based on p
  $p \leftarrow$ policy based on v
until v converges

Policy Iteration Example

Algorithm Run

- $d = \sup i'$ s.t.
  $$i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$$

Policy Iteration Example

Algorithm Run

- $d = \sup i'$ s.t.
  $$i' = i + 1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \bot$$
  1. Equation $d = \sup i'$ s.t. $\bot$ evaluates to $d = -\infty$

Policy Iteration Example

Algorithm Run

- $d = \sup i'$ s.t.
  $$i' = i + 1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \bot$$
  1. Equation $d = \sup i'$ s.t. $\bot$ evaluates to $d = -\infty$
  2. Substitute the value, not inductive:
     $-\infty = \sup i'$ s.t. $i' = i + 1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \bot$

Policy Iteration Example

Algorithm Run

- $d = \sup i'$ s.t.
  $$i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$$

1. Equation $d = \sup i'$ s.t. $\bot$ evaluates to $d = -\infty$
2. Substitute the value, not inductive:
   $-\infty = \sup i'$ s.t. $i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$
3. Increase the value to 0 using policy $d = \sup i'$ s.t. $i' = 0$

Policy Iteration Example

Algorithm Run

- $d = \sup i'$ s.t.
  $$i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$$

1. Equation $d = \sup i'$ s.t. $\bot$ evaluates to $d = -\infty$
2. Substitute the value, not inductive:
   $-\infty = \sup i'$ s.t. $i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$
3. Increase the value to 0 using policy $d = \sup i'$ s.t. $i' = 0$
4. Substituting, not inductive:
   $0 = \sup i'$ s.t. $i' = i + 1 \wedge i < 1000000 \bigvee i' = 0 \bigvee \bot$

Policy Iteration Example

Algorithm Run

- $d = \sup i'$ s.t.
  $$i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$$

1. Equation $d = \sup i'$ s.t. $\bot$ evaluates to $d = -\infty$
2. Substitute the value, not inductive:
   $-\infty = \sup i'$ s.t. $i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \bot$
3. Increase the value to 0 using policy $d = \sup i'$ s.t. $i' = 0$
4. Substituting, not inductive:
   $0 = \sup i'$ s.t. $i' = i + 1 \wedge i < 1000000 \bigvee i' = 0 \bigvee \bot$
5. Increase to 1000000 using
   $d = \sup i'$ s.t. $i' = i + 1 \wedge i < 1000000 \wedge i \leq d$

Policy Iteration Example

Algorithm Run

- $d = \sup i'$ s.t.
  $i' = i + 1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \bot$

1. Equation $d = \sup i'$ s.t. $\bot$ evaluates to $d = -\infty$
2. Substitute the value, not inductive:
   $-\infty = \sup i'$ s.t. $i' = i + 1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \bot$
3. Increase the value to 0 using policy $d = \sup i'$ s.t. $i' = 0$
4. Substituting, not inductive:
   $0 = \sup i'$ s.t. $i' = i + 1 \land i < 1000000 \bigvee i' = 0 \bigvee \bot$
5. Increase to 1000000 using
   $d = \sup i'$ s.t. $i' = i + 1 \land i < 1000000 \land i \leq d$
6. Substitute, finally inductive!
   $1000000 = \sup i'$ s.t. $i' = i+1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \bot$

Contribution

- Policy Iteration: solving global system of equations
- Contribution: local algorithm for Policy Iteration
  - At every step search for local candidate invariant
- Improved Scalability and Precision
- Ability to cooperate with other analyses

Results

- Evaluated on SV-Comp “Loops” category