Program Analysis with Local Policy Iteration

George Karpenkov

VERIMAG

May 6, 2015

Outline

Motivation

- Program verification

Motivation

- Program verification
- Finding inductive invariants

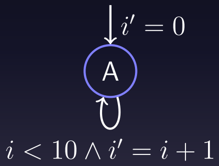
Motivation

- Program verification
- Finding inductive invariants
- LPI
  - Scalable algorithm for policy iteration
  - Sent to FMCAD'15

Program Modeling

- Control Flow Automaton (CFA)

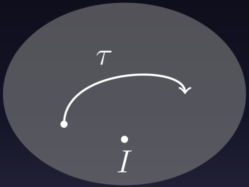
```
int i=0;
while (i<10) {
    i++;
}
```



Inductive Invariant

Motivation

- Task: verify program properties
- Prove: by induction
- Aim: find inductive invariant
  - Includes initial state
  - Closed under transition



Inductive Invariant

Abstract Interpretation Limitations

- Usual tool: abstract interpretation
- Relies on widenings/narrowings to enforce convergence
- Can be very brittle

Policy Iteration

Historical Perspective

- Game-theoretique technique
- Solving markov processes

Policy Iteration

Historical Perspective

- Game-theoretique technique
- Solving markov processes
- Used for poker AI

Policy Iteration

Introduction - 2

- Finds least inductive invariant in the given abstract domain
- Considers the program as a set of equations
- Game-theoretic algorithm adapted to find inductive invariant
- Requires abstract semantics to be monotone & concave

Policy Iteration

Introduction - 2

- Finds least inductive invariant in the given abstract domain
- Considers the program as a set of equations
- Game-theoretic algorithm adapted to find inductive invariant
- Requires abstract semantics to be monotone & concave

Guarantees

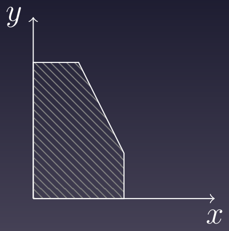
Least inductive invariant, not least invariant in general!

Outline

Template Constraints Domain

Domain used in our work

- Choose linear inequalities to be tracked before the analysis
- E.g. $x, y, x + y$ (templates)
- We want to find inductive invariant $x \leq d_1 \wedge y \leq d_2 \wedge x + y \leq d_3$ for all control states
- An element of the domain above is a vector $(3, 2, 4)$ which corresponds to $x \leq 3 \wedge y \leq 2 \wedge x + y \leq 4$



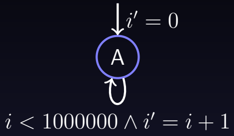
Template Constraints Domain

Abstract Semantics

- Abstract Semantics: transition relation in the abstract domain
- Convex optimization:
  - Template x, transition $x' = x + 1$, previous element $x \leq 5$
  - New element given by $\max x'$ s. t. $x' = x + 1 \land x \leq 5$

Policy Iteration

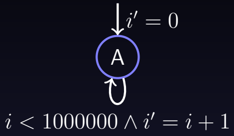
Simple Example



- Template constraints domain $\{i\}$
- Aim: find smallest d, s.t. $i \leq d$ is an inductive invariant
- Use semantical equations for d

Policy Iteration

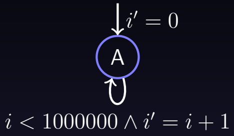
Simple Example



- Template constraints domain $\{i\}$
- Aim: find smallest d, s.t. $i \leq d$ is an inductive invariant
- Use semantical equations for d
- Necessary and sufficient condition:
- $d = \sup i'$ s.t.
  $$i' = i + 1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \perp$$
  - Disjunctions come from multiple edges

Policy Iteration

Simple Example



- Template constraints domain $\{i\}$
- Aim: find smallest d, s.t. $i \leq d$ is an inductive invariant
- Use semantical equations for d
- Necessary and sufficient condition:
- $d = \sup i'$ s.t. $i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \perp$
  - Disjunctions come from multiple edges
  - $\perp$ represents unreachable state
  - We take supremum as the answer can be ∞ (unbounded) or $-\infty$ (unreachable)

Policy Iteration

Explanation By Example

- We have a min-max equation:
  $d = \min\left(\sup i' \text{ s.t. } i' = i + 1 \land i < 1000000 \land i \leq d \bigvee i' = 0 \bigvee \bot\right)$

Policy Iteration

Explanation By Example

- We have a min-max equation:
  $d = \min\left(\sup i' \text{ s.t. } i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \perp\right)$
- We consider separate cases for disjunctions
- Replacing each disjunction with one argument
  - $d = \sup i' \text{ s.t. } i' = 0$
  - Referred to as a policy

Policy Iteration

Explanation By Example - 2

- $d = \sup i'$ s.t. $i' = 0$
- Simplified system (with no disjunctions):
  - Monotone and concave
  - ≤ 2 fixpoints
  - Can be solved using LP

Policy Iteration Example

Algorithm Run

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

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, does not hold:
     $-\infty = \sup i'$ 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.
  $$i' = i + 1 \wedge i < 1000000 \wedge i \leq d \bigvee i' = 0 \bigvee \perp$$

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

Policy Iteration Example

Algorithm Run

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

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

Policy Iteration Example

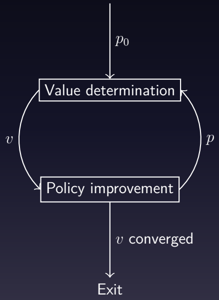
Algorithm Run

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

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

Policy Iteration

Algorithm Overview



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

Policy Iteration

Algorithm Continued

- Policy Improvement: SMT call
  - Policy which can improve current value?

- Value Determination: LP call
  - Maximum value for current policy?

Path Focusing

Similar to Large Block Encoding

- Unknown per node per template

Path Focusing

Similar to Large Block Encoding

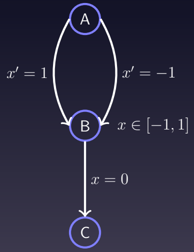
- Unknown per node per template
- Over-approximates invariant in the abstract domain
- Loss of precision

Path Focusing

Similar to Large Block Encoding

- Unknown per node per template
- Over-approximates invariant in the abstract domain
- Loss of precision

```
if (unknown()) {
    x = -1;
} else {
    x = 1;
}
assert(x != 0);
```



Path Focusing

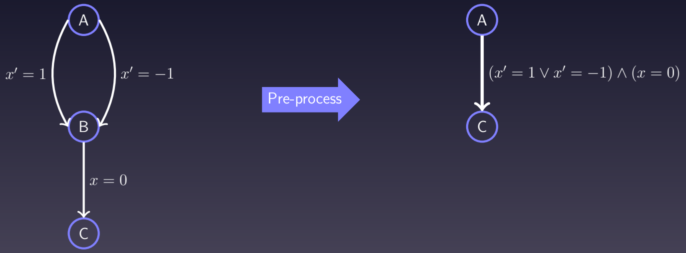
Adding disjunctions

- Solution: remove nodes!
- CFG Compaction:
  - Edges (A, τ_1, B), (B, τ_2, C), with no other incoming to B
    - Converted to $(A, \tau_1 \wedge \tau_2, C)$, B removed
  - Edges (A, τ_1, B), (A, τ_2, B), no other incoming to B
    - Converted to $(A, \tau_1 \vee \tau_2, B)$

Path Focusing

Adding disjunctions

- Solution: remove nodes!
- CFG Compaction:
  - Edges (A, τ_1, B), (B, τ_2, C), with no other incoming to B
    - Converted to $(A, \tau_1 \wedge \tau_2, C)$, B removed
  - Edges (A, τ_1, B), (A, τ_2, B), no other incoming to B
    - Converted to $(A, \tau_1 \vee \tau_2, B)$



Path Focusing

Properties

- For a well-structured graph: only loop-heads remain

Path Focusing

Properties

- For a well-structured graph: only loop-heads remain
- Disjunctions create new policies

Path Focusing

Properties

- For a well-structured graph: only loop-heads remain
- Disjunctions create new policies
- Possible improvement: cut-set
  - Set of nodes which cut all the cycles in the graph
- Disadvantage: requires pre-processing

Outline

Problems of Policy Iteration

Motivation for our work

- Problems with the approach above:
  - Scalability: at each step, we update each policy, and at each step, we solve the global equation system (of the size of the entire program)
  - Cooperability: policy iterations don't fit into any existing framework, pre-processing makes it worse

Problems of Policy Iteration

Our Contribution

- Our work: LPI (Local Policy Iteration)
  - Exploits the locality to avoid redundant computation
  - Avoids solving the global equation at each point
  - Unifies policy iteration with other approaches using CPA (Configurable Program Analysis Framework)
  - No pre-processing is involved

LPI as a Configurable Program Analysis

$$x \leq 4 \xrightarrow{x' = x + 1} ?$$

- Transfer Relation:
  - Similar to abstract interpretation
  - Record the bound along with the policy

LPI as a Configurable Program Analysis

$$x \leq 4 \xrightarrow{x' = x + 1} \boxed{?}$$

- Transfer Relation:
  - Similar to abstract interpretation
  - Record the bound along with the policy
- Global map M : location $\rightarrow$ abstract state

LPI as a Configurable Program Analysis

$$x \leq 4 \xrightarrow{x' = x + 1} ?$$

- Transfer Relation:
  - Similar to abstract interpretation
  - Record the bound along with the policy
- Global map M : location $\rightarrow$ abstract state

- When two states for the same node, merge
  - When merge closes the loop, perform value determination
  - Follow backpointers to re-create global problem

Abstract Domain

- Two lattices:
  - Abstracted State (element of template constraints domain)
  - Intermediate State (formula)
- Idea: avoid pre-processing

Abstract Domain

- Two lattices:
  - Abstracted State (element of template constraints domain)
  - Intermediate State (formula)
- Idea: avoid pre-processing
- Propagate intermediate states, convert to abstracted at loop-heads

LPI Abstract Domain

Abstracted States

Abstracted State

Set of tuples (bound, policy, backpointer) for each template.
Current bound, policy and the previous value used to derive that bound.

- Abstracted state example: $\{i : (0, i' = 0, A)\}$

LPI Abstract Domain

Abstracted States

Abstracted State

Set of tuples (bound, policy, backpointer) for each template.
Current bound, policy and the previous value used to derive that bound.

- Abstracted state example: $\{i : (0, i' = 0, A)\}$
- Partial order given by component-wise comparison on bounds

LPI Abstract Domain

Abstracted States

Abstracted State

Set of tuples (bound, policy, backpointer) for each template.
Current bound, policy and the previous value used to derive that bound.

- Abstracted state example: $\{i : (0, i' = 0, A)\}$
- Partial order given by component-wise comparison on bounds
- On merge:
  - Pick the upper bound for each template
  - Keep the corresponding policy and backpointer

Intermediate State

Formula $\phi(X')$ representing set of reachable states

Ω meta-variables instead of backpointers

- Intermediate State example:
    - $x' = 1 \land \Omega = A \lor x' = 0 \land \Omega = B$

LPI Abstract Domain

Intermediate States

Intermediate State

Formula $\phi(X')$ representing set of reachable states

Ω meta-variables instead of backpointers

- Intermediate State example:
  - $x' = 1 \land \Omega = A \lor x' = 0 \land \Omega = B$
- Propagation: symbolic execution
- Can be converted to abstracted state using abstraction
  - Maximizing for every template
  - Recording policy and backpointer

LPI

Propagation Overview

- Start with abstracted state at node A: $\{x : (0, x' = 0, I)\}$

LPI

Propagation Overview

- Start with abstracted state at node A: $\{x : (0, x' = 0, I)\}$
- Successor under edge $x' = x + 5$

LPI

Propagation Overview

- Start with abstracted state at node A: $\{x : (0, x' = 0, I)\}$
- Successor under edge $x' = x + 5$
- Intermediate state: $x' = x + 5 \wedge x \leq 0 \wedge \Omega = A$

LPI

Propagation Overview

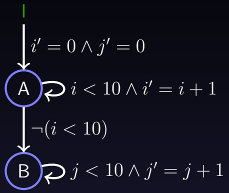
- Start with abstracted state at node A: $\{x : (0, x' = 0, I)\}$
- Successor under edge $x' = x + 5$
- Intermediate state: $x' = x + 5 \land x \leq 0 \land \Omega = A$
- If we need to perform abstraction, we get $\{x : (5, x' = x + 5, A)\}$

Local Value Determination

- On closing the loop (at abstracted state):
  - Follow backpointers, keep adding constraints
  - Create value determination problem
    - Potentially size of the largest loop

Local Policy Iteration

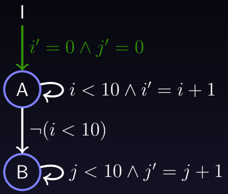
Algorithm Example



1. Start with abstracted state $\top$

Local Policy Iteration

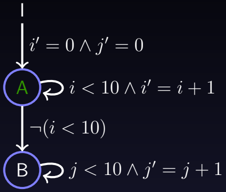
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \wedge j' = 0 \wedge \Omega = I$

Local Policy Iteration

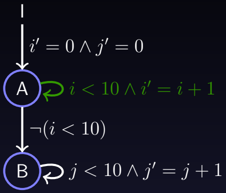
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \wedge j' = 0 \wedge \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$

Local Policy Iteration

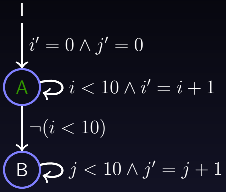
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \wedge j' = 0 \wedge \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \wedge j \leq 0 \wedge \Omega = A$

Local Policy Iteration

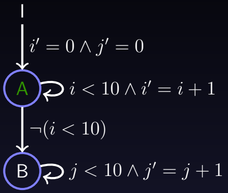
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \land j' = 0 \land \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \land j \leq 0 \land \Omega = A$
5. Abstracted to $\{i : (1, A), j : (0, A)\}$

Local Policy Iteration

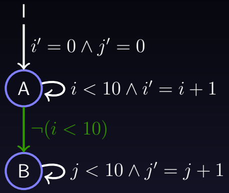
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \wedge j' = 0 \wedge \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \wedge j \leq 0 \wedge \Omega = A$
5. Abstracted to $\{i : (1, A), j : (0, A)\}$
6. Merge A, val. det.: $\{i : (10, A), j : (0, I)\}$

Local Policy Iteration

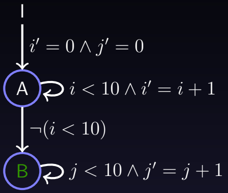
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \wedge j' = 0 \wedge \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \wedge j \leq 0 \wedge \Omega = A$
5. Abstracted to $\{i : (1, A), j : (0, A)\}$
6. Merge A, val. det.: $\{i : (10, A), j : (0, I)\}$
7. Intermediate $i \leq 10 \wedge j \leq 0 \wedge \neg(i < 10) \wedge \Omega = A$

Local Policy Iteration

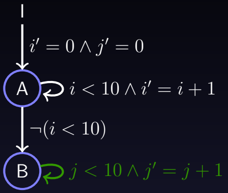
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \land j' = 0 \land \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \land j \leq 0 \land \Omega = A$
5. Abstracted to $\{i : (1, A), j : (0, A)\}$
6. Merge A, val. det.: $\{i : (10, A), j : (0, I)\}$
7. Intermediate $i \leq 10 \land j \leq 0 \land \neg(i < 10) \land \Omega = A$
8. Abstracted: $\{i : (10, A), j : (0, A)\}$

Local Policy Iteration

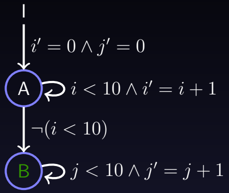
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \wedge j' = 0 \wedge \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \wedge j \leq 0 \wedge \Omega = A$
5. Abstracted to $\{i : (1, A), j : (0, A)\}$
6. Merge A, val. det.: $\{i : (10, A), j : (0, I)\}$
7. Intermediate $i \leq 10 \wedge j \leq 0 \wedge \neg(i < 10) \wedge \Omega = A$
8. Abstracted: $\{i : (10, A), j : (0, A)\}$
9. Intermediate:
   $i = 10 \wedge j \leq 0 \wedge j' = j + 1 \wedge \Omega = B$

Local Policy Iteration

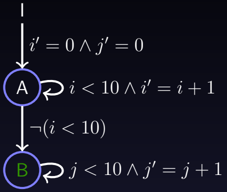
Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \land j' = 0 \land \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \land j \leq 0 \land \Omega = A$
5. Abstracted to $\{i : (1, A), j : (0, A)\}$
6. Merge A, val. det.: $\{i : (10, A), j : (0, I)\}$
7. Intermediate $i \leq 10 \land j \leq 0 \land \neg(i < 10) \land \Omega = A$
8. Abstracted: $\{i : (10, A), j : (0, A)\}$
9. Intermediate:
   $i = 10 \land j \leq 0 \land j' = j + 1 \land \Omega = B$
10. Abstracted: $\{i : (10, B), j : (1, B)\}$

Local Policy Iteration

Algorithm Example



1. Start with abstracted state $\top$
2. Intermediate state $i' = 0 \land j' = 0 \land \Omega = I$
3. Abstracted to $\{i : (0, I), j : (0, I)\}$
4. Intermediate state $i \leq 0 \land j \leq 0 \land \Omega = A$
5. Abstracted to $\{i : (1, A), j : (0, A)\}$
6. Merge A, val. det.: $\{i : (10, A), j : (0, I)\}$
7. Intermediate $i \leq 10 \land j \leq 0 \land \neg(i < 10) \land \Omega = A$
8. Abstracted: $\{i : (10, A), j : (0, A)\}$
9. Intermediate:
   $i = 10 \land j \leq 0 \land j' = j + 1 \land \Omega = B$
10. Abstracted: $\{i : (10, B), j : (1, B)\}$
11. Merge B, val. det.: $\{i : (10, B), j : (10, A)\}$

Reachability of Bad States

- Whether we are safe:
  - $\phi \wedge E$ is unsat
  - Example: $(x \leq 10) \wedge (x = 11)$

Reachability of Bad States

- Whether we are safe:
  - $\phi \land E$ is unsat
  - Example: $(x \leq 10) \land (x = 11)$
- Whether we are unsafe:
  - $\phi \land \neg E$ is unsat
  - Example: $(x = 0) \land (x = 0)$

Algorithm Properties

- Soundness
  - Only terminate when inductive

Algorithm Properties

- Soundness
  - Only terminate when inductive
- Termination
  - Bounds can only grow
  - Each bound corresponds to some policy
  - Finite number of policies

Algorithm Properties

- Soundness
  - Only terminate when inductive
- Termination
  - Bounds can only grow
  - Each bound corresponds to some policy
  - Finite number of policies
- Least invariant property
  - Only select feasible policies

LPI Configurations

- Configurations
  - Intervals ($\pm x$)
  - Octagons (above and $\pm x, \pm x \pm y$)
  - Rich Templates (above and $\pm 2x \pm y, \pm x \pm y \pm z, \pm 2x \pm y \pm z$)
  - Unrolling
  - Simple Congruence Analysis

LPI Configurations

- Configurations
  - Intervals ($\pm x$)
  - Octagons (above and $\pm x, \pm x \pm y$)
  - Rich Templates (above and $\pm 2x \pm y, \pm x \pm y \pm z, \pm 2x \pm y \pm z$)
  - Unrolling
  - Simple Congruence Analysis
- Refinement: progressively switch to more expensive config

Contrast with Classical Policy Iteration

- Only update policies which need updating
- Run value determination on a reduced program section
- Stated in CPA framework
- (Unguided) refinement of template precision
- Local value determination optimizations
  Not in the presentation

Outline

Local Policy Iteration

Code analysis tool

Tool

Included in CPAchecker trunk.

https://github.com/dbeyer/cpachecker

- Configurations:
  - `-policy-intervals`
  - `-policy`
  - `-policy-ensemble`
  - `-policy-counterexample-checking`

LPI Evaluation

- Evaluated on SV-Comp "Loops" category
- Compared with
  - BLAST(2014)
  - CPAchecker-SVcomp15
  - PAGAI
- Across true benchmarks

Results

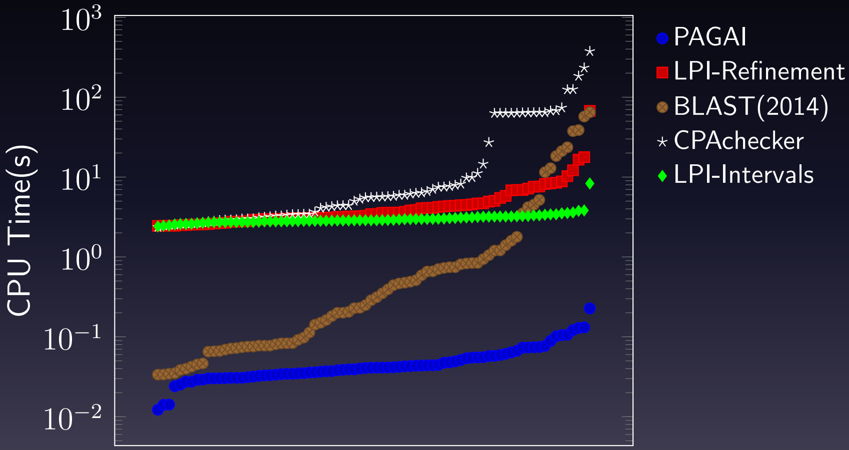
Comparison of Approaches

	vs. LPI	PAGAI	BLAST	CPAchecker	Unique	**Verified**	**Incorrect**
LPI		13	21	22	8	60	1
PAGAI	5		14	15	0	52	1
BLAST	4	5		7	0	43	1
CPAchecker	19	20	21		12	57	2

- Difference between approaches
- Reads: A vs. B

Results

Timing Results



Contributions

Re-iterating

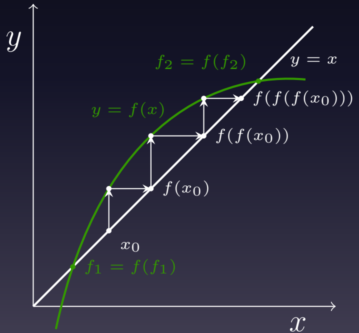
- New scalable algorithm for policy iteration
- Tool for program analysis (using CPAchecker framework)
  - The only policy-iteration based tool capable of dealing with C

Questions?

Policy Iteration

Fixpoints and Concavity

- Concavity and monotonicity limits the number of fixpoints
- Can solve for $x = f(x)$



Policy Iteration

Concavity of Abstract Semantics

- Linear Semantics: $x' = Tx \wedge G(x)$
- Let $t' = t(Tx)$