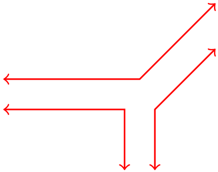
Using Dynamic Analysis to Generate Disjunctive Invariants

ThanhVu (Vu) Nguyen[*],
Deepak Kapur[*], Westley Weimer[†], Stephanie Forrest[*]

[*]University of New Mexico, [†]University of Virginia

ICSE 2014



Program Invariants

- *"Invariants are asserted properties, such as relations among variables, at certain locations in a program"*
  - Assertions
  - Pre/Post conditions
  - Loop invariants

Program Invariants

- *"Invariants are asserted properties, such as relations among variables, at certain locations in a program"*
    - Assertions
    - Pre/Post conditions
    - Loop invariants
- Uses of Invariants
    - Understand and verify programs
    - Debug (locate errors)
    - Formal proofs
    - Documentations

Program Invariants

- *"Invariants are asserted properties, such as relations among variables, at certain locations in a program"*
  - Assertions
  - Pre/Post conditions
  - Loop invariants
- Uses of Invariants
  - Understand and verify programs
  - Debug (locate errors)
  - Formal proofs
  - Documentations
- Invariants can generated using static or dynamic analysis
  - Static analysis examines program source code
  - Dynamic learns from program execution traces

Polynomial Invariants

- A polynomial invariant is a relation among numerical program variables, e.g., $x = 2y + 3, |arr| \geq x \geq 0$

Polynomial Invariants

- A polynomial invariant is a relation among numerical program variables, e.g., $x = 2y + 3, |arr| \geq x \geq 0$
- A conjunctive polynomial invariant is a set (logical and) of polynomial inariants, e.g., $x = 2y + 3 \ \wedge \ |arr| \geq x \geq 0$

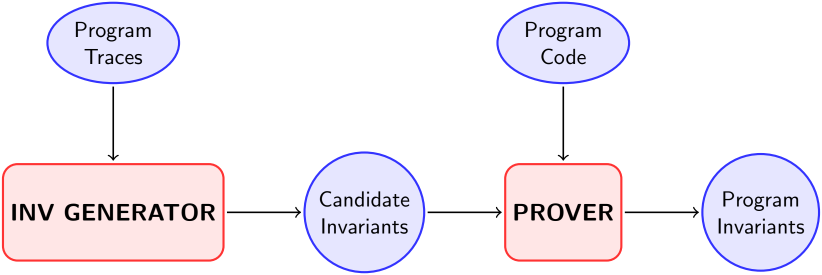
Polynomial Invariants

- A polynomial invariant is a relation among numerical program variables, e.g., $x = 2y + 3, |arr| \geq x \geq 0$
- A conjunctive polynomial invariant is a set (logical and) of polynomial inariants, e.g., $x = 2y + 3 \ \wedge \ |arr| \geq x \geq 0$
- A disjunctive polynomial invariant is a disjunct (logical or) of polynomial invariants, e.g., $x = 2y + 3 \ \vee \ |arr| \geq x \geq 0$
  - Represent semantics of branching
    The invariant after `if (p) {a=1;} else {a=2;}` is
    $(p \wedge a = 1) \vee (\neg p \wedge a = 2)$
  - Existing approaches focus mostly on *conjunctive* relations

Polynomial Invariants

- A polynomial invariant is a relation among numerical program variables, e.g., $x = 2y + 3, |arr| \geq x \geq 0$
- A conjunctive polynomial invariant is a set (logical and) of polynomial inariants, e.g., $x = 2y + 3 \ \wedge \ |arr| \geq x \geq 0$
- A disjunctive polynomial invariant is a disjunct (logical or) of polynomial invariants, e.g., $x = 2y + 3 \ \vee \ |arr| \geq x \geq 0$
  - Represent semantics of branching
    The invariant after `if (p) {a=1;} else {a=2;}` is
    $(p \wedge a = 1) \vee (\neg p \wedge a = 2)$
  - Existing approaches focus mostly on *conjunctive* relations
- Existing approaches have *trade-offs* among soundness, efficiency, and expressive power
  - Static analyzers, e.g., Interproc, Astrée, support conjunctive relations
  - Dynamic techniques, e.g., Daikon, also have limited support for disjunctive invariants and can produce spurious results

Hybrid Invariant Generation



- **DIG** (**D**ynamic **I**nvariant **G**enerator)
  - Find invariants directly from program traces
  - Build nonconvex polyhedra over trace points and extract facets representing disjunctive relations
- **KIP** (**K**-**I**nductive **P**rover)
  - Verify candidate invariants statically from program code
  - Based on k-induction and SMT solving

Geometric Invariant Inference

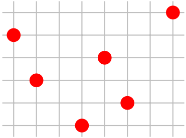
- Treat trace values as points in multi-dimensional space
- Build a convex hull (polyhedron) over the points
- Representation of a polyhedron: a conjunction of inequalities

Geometric Invariant Inference

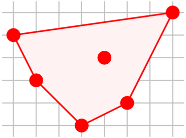
- Treat trace values as points in multi-dimensional space
- Build a convex hull (polyhedron) over the points
- Representation of a polyhedron: a conjunction of inequalities

x	y
-2	1
-1	-1
1	-3
2	0
3	-2
5	2

program traces



trace pts
in 2D



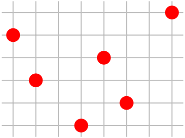
polygon
$c_1x + c_2y \geq c$

Geometric Invariant Inference

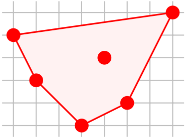
- Treat trace values as points in multi-dimensional space
- Build a convex hull (polyhedron) over the points
- Representation of a polyhedron: a conjunction of inequalities

x	y
-2	1
-1	-1
1	-3
2	0
3	-2
5	2

program traces

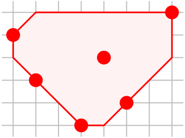


trace pts
in 2D

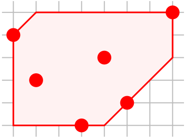


polygon
$c_1 x + c_2 y \geq c$

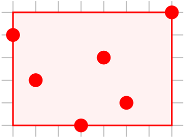
- Consider simpler shapes (decreasing precision, increasing efficiency)



octagon
$\pm x \pm y \geq c$



zone
$x - y \geq c$



box
$\pm x, y \geq c$

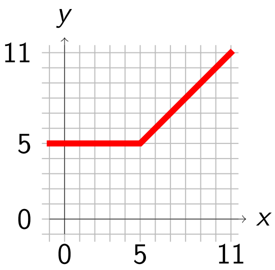
Outline

Example

```
def ex(x):
  y = 5
  if x > y: x = y
  while[L] x ≤ 10:
    if x ≥ 5:
      y = y + 1
    x = x + 1

  assert y ≡ 11
```

x	y
-1	5
⋮	⋮
5	5
6	6
⋮	⋮
11	11

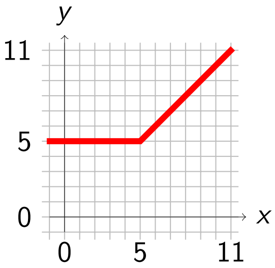


Example

```
def ex(x):
  y = 5
  if x > y: x = y
  while[L] x ≤ 10:
    if x ≥ 5:
      y = y + 1
    x = x + 1

  assert y ≡ 11
```

x	y
-1	5
⋮	⋮
5	5
6	6
⋮	⋮
11	11



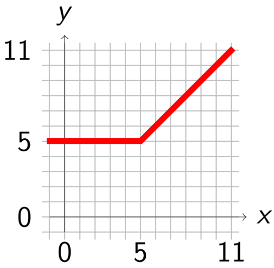
$$L : (x < 5 \wedge 5 = y) \ \vee \ (x \geq 5 \wedge x = y), 11 \geq x$$

Example

```
def ex(x):
  y = 5
  if x > y: x = y
  while[L] x ≤ 10:
    if x ≥ 5:
      y = y + 1
    x = x + 1

  assert y ≡ 11
```

x	y
-1	5
⋮	⋮
5	5
6	6
⋮	⋮
11	11



$$L : (x < 5 \wedge 5 = y) \ \vee \ (x \geq 5 \wedge x = y), 11 \geq x$$

Disjunction of 2 cases:

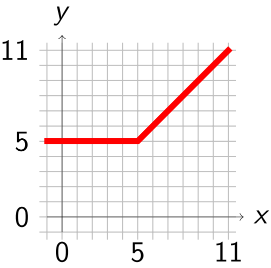
1. if $x < 5$ then $y = 5$
2. if $x \geq 5$ then $x = y$

Example

```
def ex(x):
  y = 5
  if x > y: x = y
  while[L] x ≤ 10:
    if x ≥ 5:
      y = y + 1
    x = x + 1

  assert y ≡ 11
```

x	y
-1	5
⋮	⋮
5	5
6	6
⋮	⋮
11	11



$$L : (x < 5 \land 5 = y) \;\lor\; (x \geq 5 \land x = y), 11 \geq x$$

Disjunction of 2 cases:

1. if $x < 5$ then $y = 5$
2. if $x \geq 5$ then $x = y$

$\longleftrightarrow$ if $0 > x - 5$ then $0 = y - 5$ else $x - 5 = y - 5$

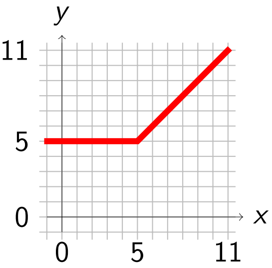
$\max(0, x - 5) = y - 5$

Example

```
def ex(x):
  y = 5
  if x > y: x = y
  while[L] x ≤ 10:
    if x ≥ 5:
      y = y + 1
    x = x + 1

  assert y ≡ 11
```

x	y
-1	5
$\vdots$	$\vdots$
5	5
6	6
$\vdots$	$\vdots$
11	11



$$L : (x < 5 \wedge 5 = y) \;\vee\; (x \geq 5 \wedge x = y), 11 \geq x$$

Disjunction of 2 cases:

1. if $x < 5$ then $y = 5$
2. if $x \geq 5$ then $x = y$

$\longleftrightarrow$ if $0 > x - 5$ then $0 = y - 5$ else $x - 5 = y - 5$

$\max(0, x - 5) = y - 5$

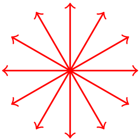
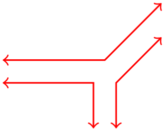
a linear relation .. in max-plus algebra

Max-plus Algebra

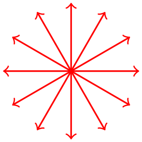
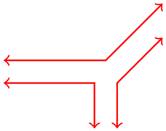
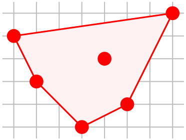
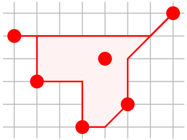
Max-plus Algebra

	Linear	Max-plus
Domain	$\mathbb{R}$	$\mathbb{R} \cup \{-\infty\}$
Addition	$+$	max
Multiplication	$\times$	$+$
Zero elem	0	$-\infty$
Unit elem	1	0
Relation form	$c_0 + c_1 t_1 + \cdots + c_n t_n \geq 0$	$\max(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \max(d_0, d_1 + t_1, \ldots, d_n + t_n)$

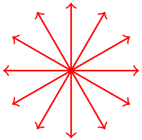
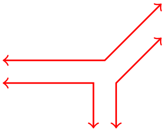
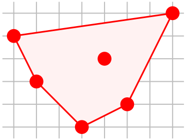
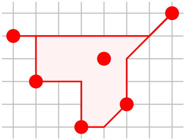
Max-plus Algebra

	Linear	**Max-plus**
Domain	$\mathbb{R}$	$\mathbb{R} \cup \{-\infty\}$
Addition	$+$	max
Multiplication	$\times$	$+$
Zero elem	0	$-\infty$
Unit elem	1	0
Relation form	$c_0 + c_1 t_1 + \cdots + c_n t_n \geq 0$	$\max(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \max(d_0, d_1 + t_1, \ldots, d_n + t_n)$
Line shapes		

Max-plus Algebra

	Linear	**Max-plus**
Domain	$\mathbb{R}$	$\mathbb{R} \cup \{-\infty\}$
Addition	$+$	max
Multiplication	$\times$	$+$
Zero elem	0	$-\infty$
Unit elem	1	0
Relation form	$c_0 + c_1 t_1 + \cdots + c_n t_n \geq 0$	$\max(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \max(d_0, d_1 + t_1, \ldots, d_n + t_n)$
Line shapes		
Convex hull		

Max-plus Algebra

	Linear	**Max-plus**
Domain	$\mathbb{R}$	$\mathbb{R} \cup \{-\infty\}$
Addition	$+$	max
Multiplication	$\times$	$+$
Zero elem	0	$-\infty$
Unit elem	1	0
Relation form	$c_0 + c_1 t_1 + \cdots + c_n t_n \geq 0$	$\max(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \max(d_0, d_1 + t_1, \ldots, d_n + t_n)$
Line shapes		
Convex hull		

Not convex in classical sense!

Dynamically Inferring Max-plus Invariants

Examples

$$z = \max(x, y) \quad \equiv \quad (x < y \wedge z = x) \quad \vee \quad (x \geq y \wedge z = y)$$

Dynamically Inferring Max-plus Invariants

Examples

$$
\begin{array}{rcccc}
z = \max(x, y) & \equiv & (x < y \wedge z = x) & \vee & (x \geq y \wedge z = y) \\
\mathrm{strncpy}(s, d, n) & \equiv & (n \geq |s| \wedge |d| = |s|) & \vee & (n < |s| \wedge |d| \geq n)
\end{array}
$$

Dynamically Inferring Max-plus Invariants

Examples

$$
\begin{aligned}
z = \max(x, y) \quad &\equiv \quad (x < y \wedge z = x) \quad \vee \quad (x \geq y \wedge z = y) \\
\mathrm{strncpy}(s, d, n) \quad &\equiv \quad (n \geq |s| \wedge |d| = |s|) \quad \vee \quad (n < |s| \wedge |d| \geq n)
\end{aligned}
$$

DIG discovers disjunctive relations of the max-plus form

$$\max(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \max(d_0, d_1 + t_1, \ldots, d_n + t_n)$$

Dynamically Inferring Max-plus Invariants

Examples

$$
\begin{array}{rcccc}
z = \max(x, y) & \equiv & (x < y \wedge z = x) & \vee & (x \geq y \wedge z = y) \\
\mathrm{strncpy}(s, d, n) & \equiv & (n \geq |s| \wedge |d| = |s|) & \vee & (n < |s| \wedge |d| \geq n)
\end{array}
$$

DIG discovers disjunctive relations of the max-plus form

$$\max(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \max(d_0, d_1 + t_1, \ldots, d_n + t_n)$$

Method

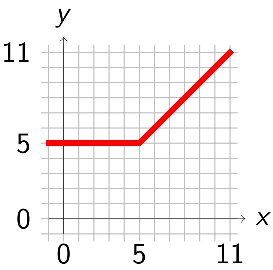
- Represent trace values as points
- Build a max-plus convex polyhedron
- Extract facets represented by max-plus relations

Example

x	y
-1	5
$\vdots$	$\vdots$
5	5
6	6
$\vdots$	$\vdots$
11	11

Example

x	y
-1	5
$\vdots$	$\vdots$
5	5
6	6
$\vdots$	$\vdots$
11	11



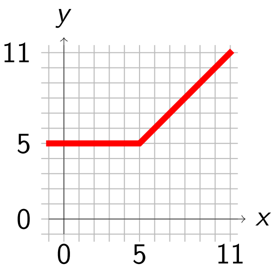
- DIG builds a max-plus polygon and finds candidate invs:

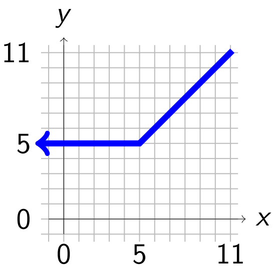
$$11 \geq x \geq -1, \quad 11 \geq y \geq 5, \quad 0 \geq x - y \geq -6$$

$$\max(0, x-5) \geq y - 5 \;\equiv\; (x < 5 \;\wedge\; 5 \geq y) \;\vee\; (x \geq 5 \;\wedge\; y \leq x)$$

Example

x	y
-1	5
$\vdots$	$\vdots$
5	5
6	6
$\vdots$	$\vdots$
11	11





- DIG builds a max-plus polygon and finds candidate invs:

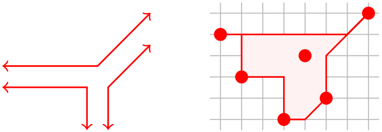
$$11 \geq x \geq -1, \quad 11 \geq y \geq 5, \quad 0 \geq x - y \geq -6$$

$$\max(0, x-5) \geq y - 5 \;\equiv\; (x < 5 \;\wedge\; 5 \geq y) \;\vee\; (x \geq 5 \;\wedge\; y \leq x)$$

- KIP removes the spurious invariants $x \geq -1$ and $x - y \geq -6$
- Remaining invariants are true and equivalent to

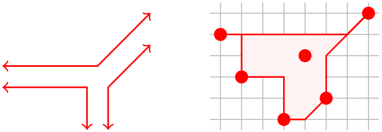
$$(x < 5 \;\wedge\; 5 = y) \;\vee\; (11 \geq x \geq 5 \;\wedge\; x = y)$$

Weak Max-plus Invariants

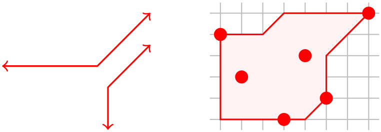


General Max-plus

Weak Max-plus Invariants

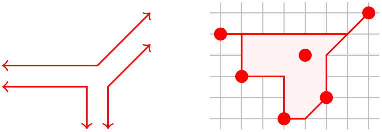


General Max-plus

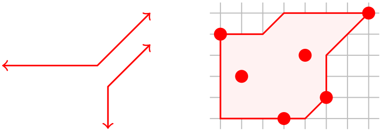


Weak Max-plus

Weak Max-plus Invariants



General Max-plus



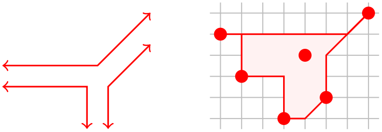
Weak Max-plus

- DIG introduces weak max-plus relations of the form

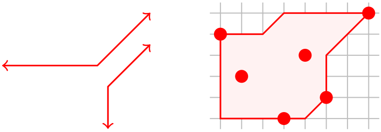
$$\max(c_0, c_1 + t_1, \ldots, c_k + t_k) \geq t_j + d,$$
$$t_j + d \geq \max(c_0, c_1 + t_1, \ldots, c_k + t_k),$$

where $d \in \mathbb{R}$ and $t_j \in \{t_1, \ldots, t_k\}$

Weak Max-plus Invariants



General Max-plus



Weak Max-plus

- DIG introduces weak max-plus relations of the form

$$\max(c_0, c_1 + t_1, \ldots, c_k + t_k) \geq t_j + d,$$
$$t_j + d \geq \max(c_0, c_1 + t_1, \ldots, c_k + t_k),$$

where $d \in \mathbb{R}$ and $t_j \in \{t_1, \ldots, t_k\}$

  - Restrict values of coefs c_i to $\{0, -\infty\}$
  - Fix the number k of variables t_i, e.g., $k = 2$
  - Allow only one unknown param d

Algorithm for Finding Weak Max-plus Invs

Given 2D points $\{(x_1, y_1), \dots, (x_n, y_n)\}$, build a weak max-plus polygon, i.e., a conjunction of inequalities of the form

$$\begin{aligned} \max(c_0, c_1 + x, c_2 + y) &\geq x + d, \quad x + d \geq \max(c_0, c_1 + x, c_k + y), \\ \max(c_0, c_1 + x, c_2 + y) &\geq y + d, \quad y + d \geq \max(c_0, c_1 + x, c_k + y) \end{aligned}$$

Algorithm for Finding Weak Max-plus Invs

Given 2D points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, build a weak max-plus polygon, i.e., a conjunction of inequalities of the form

$$\begin{array}{ll} \max(c_0, c_1 + x, c_2 + y) \geq x + d, & x + d \geq \max(c_0, c_1 + x, c_k + y), \\ \max(c_0, c_1 + x, c_2 + y) \geq y + d, & y + d \geq \max(c_0, c_1 + x, c_k + y) \end{array}$$

1. Enumerate weak relations by instantiating c_i over $\{0, -\infty\}$
   - Each weak max-plus form yields at most 8 relations, e.g., $\max(c_0, c_1 + x, c_2 + y) \geq x + d$ produces

     $$\begin{array}{ll} \max(0, x, y) \geq x + d, & \max(0, x) \geq x + d, \\ \max(0, y) \geq x + d, & 0 \geq x + d, \ldots \end{array}$$

   - Obtain at most 32 relations from 4 weak max-plus forms

Algorithm for Finding Weak Max-plus Invs

Given 2D points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, build a weak max-plus polygon, i.e., a conjunction of inequalities of the form

$$\begin{aligned} \max(c_0, c_1 + x, c_2 + y) &\geq x + d, \quad x + d \geq \max(c_0, c_1 + x, c_k + y), \\ \max(c_0, c_1 + x, c_2 + y) &\geq y + d, \quad y + d \geq \max(c_0, c_1 + x, c_k + y) \end{aligned}$$

1. Enumerate weak relations by instantiating c_i over $\{0, -\infty\}$
   - Each weak max-plus form yields at most 8 relations, e.g., $\max(c_0, c_1 + x, c_2 + y) \geq x + d$ produces

     $$\begin{aligned} &\max(0, x, y) \geq x + d, \quad \max(0, x) \geq x + d, \\ &\max(0, y) \geq x + d, \quad 0 \geq x + d, \ldots \end{aligned}$$

   - Obtain at most 32 relations from 4 weak max-plus forms
2. Solve for d in the obtained relations using given points, e.g.,

$$\begin{aligned} \max(0, y) \geq x + d \quad &\rightarrow \quad d = \min(\max(0, y_i) - x_i) \\ x + d \geq \max(0, y) \quad &\rightarrow \quad d = \max(\max(0, y_i) - x_i) \end{aligned}$$

Min-plus Invariants

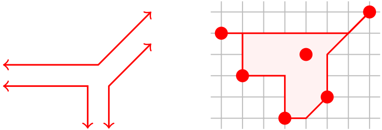
- DIG discovers disjunctive relations of the min-plus form

$$\min(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \min(d_0, d_1 + t_1, \ldots, d_n + t_n)$$

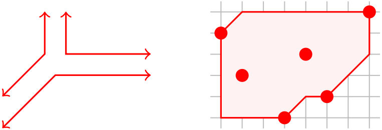
Min-plus Invariants

- DIG discovers disjunctive relations of the min-plus form

$$\min(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \min(d_0, d_1 + t_1, \ldots, d_n + t_n)$$



Max-plus

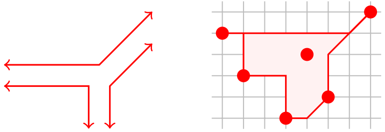


Min-plus

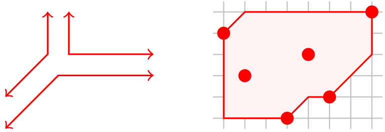
Min-plus Invariants

- DIG discovers disjunctive relations of the min-plus form

$$\min(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \min(d_0, d_1 + t_1, \ldots, d_n + t_n)$$



Max-plus



Min-plus

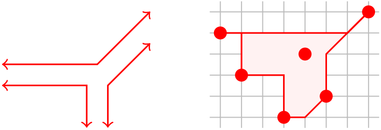
- Also support weak min-plus relations of the form

$$\min(c_0, c_1 + t_1, \ldots, c_k + t_k) \geq t_j + d_i,$$
$$t_j + d_i \geq \min(c_0, c_1 + t_1, \ldots, c_k + t_k)$$

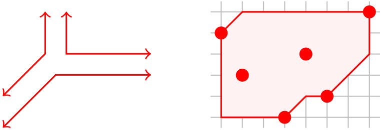
Min-plus Invariants

- DIG discovers disjunctive relations of the min-plus form

$$\min(c_0, c_1 + t_1, \ldots, c_n + t_n) \geq \min(d_0, d_1 + t_1, \ldots, d_n + t_n)$$



Max-plus



Min-plus

- Also support weak min-plus relations of the form

$$\min(c_0, c_1 + t_1, \ldots, c_k + t_k) \geq t_j + d_i,$$
$$t_j + d_i \geq \min(c_0, c_1 + t_1, \ldots, c_k + t_k)$$

- Combine max and min-plus invariants for more experessive power, e.g., can capture `iff` behavior

Algorithmic Analysis

	Linear	**Max/Min-plus**	**Weak Max/Min-plus**
Complexity	$O(p^{\frac{n}{2}})$	$O(pn^2(p+n)^n)$	$O(p2^k)$

p = # of trace pts, n = # of variables, k = a predefined constant, e.g., k = 2

Algorithmic Analysis

	Linear	**Max/Min-plus**	**Weak Max/Min-plus**
Complexity	$O(p^{\frac{n}{2}})$	$O(pn^2(p+n)^n)$	$O(p2^k)$
Facets	unbounded	unbounded	$k2^{k+2}$

p = # of trace pts, n = # of variables, k = a predefined constant, e.g., k = 2

Algorithmic Analysis

	Linear	**Max/Min-plus**	**Weak Max/Min-plus**
Complexity	$O(p^{\frac{n}{2}})$	$O(pn^2(p+n)^n)$	$O(p2^k)$
Facets	unbounded	unbounded	$k2^{k+2}$

p = # of trace pts, n = # of variables, k = a predefined constant, e.g., k = 2

Underapproximation Property

- *Theorem*: If the form of the *true* invariant f is supported, then DIG generates only the candidate invariant f' such that $f' \Rightarrow f$
- *Application*: useful for debugging; if f' *strictly* overapproximates f, then there is a trace representing a counterexample violating f

Outline

Proving Program Invariants Using k-Induction

- Represent a program execution as a state transition system

$$M = (I, T),$$

with the initial state I and the transition relation T

Proving Program Invariants Using k-Induction

- Represent a program execution as a state transition system

$$M = (I, T),$$

  with the initial state I and the transition relation T

- Use k-induction to prove that p is an invariant of M

$$\begin{array}{rcl} I \wedge T_1 \wedge \cdots \wedge T_k & \Rightarrow & p_0 \wedge \cdots \wedge p_k \\ p_n \wedge T_{n+1} \wedge \cdots \wedge p_{n+k} \wedge T_{n+k+1} & \Rightarrow & p_{n+k+1} \end{array}$$

  - More powerful than standard ($k = 0$) induction
  - Help prove properties cannot be proved using standard induction

Example

```
def sqrt(x):
  assert(x ≥ 0);
  a = 0; s = 1; t = 1
  while[L] s ≤ x:
    a += 1; t += 2; s += t
  return a
```

DIG

find candidate invariants at L

$$4s = t^2 + 2t + 1$$
$$t = 2a + 1$$
$$s = (a + 1)^2$$
$$s \geq t$$
$$9989 \geq x$$
$$\vdots$$

Example

```
def sqrt(x):
  assert(x ≥ 0);
  a = 0; s = 1; t = 1
  while[L] s ≤ x:
    a += 1; t += 2; s += t
  return a
```

DIG find candidate invariants at L	**KIP** distinguish true and spurious invs
$4s = t^2 + 2t + 1$	
$t = 2a + 1$	
$s = (a + 1)^2$	
$s \geq t$	
$9989 \geq x$	
$\vdots$	

Example

```
def sqrt(x):
  assert(x ≥ 0);
  a = 0; s = 1; t = 1
  while[L] s ≤ x:
    a += 1; t += 2; s += t
  return a
```

DIG find candidate invariants at L	**KIP** distinguish true and spurious invs
$4s = t^2 + 2t + 1$	inductive
$t = 2a + 1$	inductive
$s = (a + 1)^2$	1-inductive
$s \geq t$	
$9989 \geq x$	
$\vdots$	

Example

```
def sqrt(x):
  assert(x ≥ 0);
  a = 0; s = 1; t = 1
  while[L] s ≤ x:
    a += 1; t += 2; s += t
  return a
```

DIG find candidate invariants at L	**KIP** distinguish true and spurious invs
$4s = t^2 + 2t + 1$	inductive
$t = 2a + 1$	inductive
$s = (a + 1)^2$	1-inductive
$s \geq t$	potentially non-inductive
$9989 \geq x$	
$\vdots$	

Example

```
def sqrt(x):
  assert(x ≥ 0);
  a = 0; s = 1; t = 1
  while[L] s ≤ x:
    a += 1; t += 2; s += t
  return a
```

DIG find candidate invariants at L	**KIP** distinguish true and spurious invs
$4s = t^2 + 2t + 1$	inductive
$t = 2a + 1$	inductive
$s = (a + 1)^2$	1-inductive
$s \geq t$	potentially non-inductive
$9989 \geq x$	spurious
$\vdots$	$\vdots$

Features of KIP

- Iterative k-induction
- Employ the Z3 SMT solver
- Learn lemmas
- Eliminate redundancy
- Parallelism

Outline

Evaluation

Benchmarks

- Disjunctive testsuite: 14 programs require disjunctive invariants
- Nonlinear test suite: 27 programs require nonlinear invariants

Setup

- Implemented in SAGE/Python (with Z3 backend solver)
- Test machine: 64-core 2.6GHZ CPU, 128GB RAM, Linux OS
- Traces obtained at loop entrances and program exits

Evaluation

Benchmarks

- Disjunctive testsuite: 14 programs require disjunctive invariants
- Nonlinear test suite: 27 programs require nonlinear invariants

Setup

- Implemented in SAGE/Python (with Z3 backend solver)
- Test machine: 64-core 2.6GHZ CPU, 128GB RAM, Linux OS
- Traces obtained at loop entrances and program exits

Results

- All generated equalities are valid and most inequalities are spurious (and removed by KIP)
- Identified invariants are sufficiently strong to explain program behavior
- Current dynamic analysis cannot find any of these invariants

Results for Disjunctive Invariants

Prog	Loc	Var
ex	1	2
strncpy	1	3
oddeven3	1	6
oddeven4	1	8
oddeven5	1	10
bubble3	1	6
bubble4	1	8
bubble5	1	10
partd3	4	5
partd4	5	6
partd5	6	7
parti3	4	5
parti4	5	6
parti5	6	7
total		

Results for Disjunctive Invariants

Prog	Loc	Var	Gen	T_{Gen} (secs)
ex	1	2	15	0.2
strncpy	1	3	69	1.1
oddeven3	1	6	286	3.7
oddeven4	1	8	867	12.7
oddeven5	1	10	2334	56.8
bubble3	1	6	249	4.1
bubble4	1	8	832	11.7
bubble5	1	10	2198	53.9
partd3	4	5	479	10.5
partd4	5	6	1217	23.3
partd5	6	7	2943	53.3
parti3	4	5	464	10.3
parti4	5	6	1148	22.4
parti5	6	7	2954	53.6
total			16055	317.6

Results for Disjunctive Invariants

Prog	**Loc**	**Var**	**Gen**	**T_{Gen} (secs)**	**Val**	**T_{Val} (secs)**
ex	1	2	15	0.2	4	1.5
strncpy	1	3	69	1.1	4	7.7
oddeven3	1	6	286	3.7	8	16.0
oddeven4	1	8	867	12.7	22	46.0
oddeven5	1	10	2334	56.8	52	1319.4
bubble3	1	6	249	4.1	8	4.9
bubble4	1	8	832	11.7	22	47.6
bubble5	1	10	2198	53.9	52	938.2
partd3	4	5	479	10.5	10	50.8
partd4	5	6	1217	23.3	15	181.1
partd5	6	7	2943	53.3	21	418.1
parti3	4	5	464	10.3	10	45.5
parti4	5	6	1148	22.4	15	165.1
parti5	6	7	2954	53.6	21	405.6
total			16055	317.6	264	3647.5

Results for Disjunctive Invariants

Prog	**Loc**	**Var**	**Gen**	$\mathbf{T_{Gen}}$ **(secs)**	**Val**	$\mathbf{T_{Val}}$ (secs)	**Strength**
ex	1	2	15	0.2	4	1.5	✓
strncpy	1	3	69	1.1	4	7.7	✓
oddeven3	1	6	286	3.7	8	16.0	✓
oddeven4	1	8	867	12.7	22	46.0	✓
oddeven5	1	10	2334	56.8	52	1319.4	✓
bubble3	1	6	249	4.1	8	4.9	✓
bubble4	1	8	832	11.7	22	47.6	✓
bubble5	1	10	2198	53.9	52	938.2	✓
partd3	4	5	479	10.5	10	50.8	✓
partd4	5	6	1217	23.3	15	181.1	✓
partd5	6	7	2943	53.3	21	418.1	✓
parti3	4	5	464	10.3	10	45.5	✓
parti4	5	6	1148	22.4	15	165.1	✓
parti5	6	7	2954	53.6	21	405.6	✓
total			16055	317.6	264	3647.5	**14/14**

Results for Complex Invariants

Prog	Loc	Var	Gen	T_{Gen} (secs)	Val	T_{Val} (secs)	kI	Strength
cohendv	2	6	152	26.2	7	8.2	14	✓
divbin	2	5	96	37.7	8	8.7	15	–
manna	1	5	49	19.2	3	5.6	2	✓
hard	2	6	107	14.2	11	9.2	4	–
sqrt1	1	4	27	25.3	3	4.3	1	✓
dijkstra	2	5	61	30.7	8	10.9	6	–
freire1	1	3	25	22.5	2	2.2	0	✓
freire2	1	4	35	26.0	3	5.1	1	✓
cohencb	1	5	31	23.6	4	4.2	1	✓
egcd1	1	8	108	43.1	1	12.8	8	–
egcd2	2	10	209	60.8	8	14.6	12	✓
egcd3	3	12	475	67.0	14	23.4	25	✓
lcm1	3	6	203	38.9	12	14.2	0	✓
lcm2	1	6	52	14.9	1	0.9	10	✓
prodbin	1	5	61	28.3	3	1.1	10	–
prod4br	1	6	42	9.6	4	8.6	7	✓
fermat1	3	5	217	75.7	6	6.2	1	✓
fermat2	1	5	70	25.8	2	5.2	0	✓
knuth	1	8	113	57.1	4	24.6	6	✓
geo1	1	4	25	16.7	2	1.5	4	✓
geo2	1	4	45	24.1	1	2.1	10	✓
geo3	1	5	65	22.1	1	2.7	12	✓
ps2	1	3	25	21.1	2	4.0	0	✓
ps3	1	3	25	21.9	2	4.2	0	✓
ps4	1	3	25	23.5	2	4.9	0	✓
ps5	1	3	24	24.9	2	7.4	0	✓
ps6	1	3	25	25.0	2	69.5	0	✓
total			2392	825.9	118	149	266.3	22/27

Results for Complex Invariants

Prog	Loc	Var	Gen	T_{Gen} (secs)	Val	T_{Val} (secs)	*k*I	Strength
cohendv	2	6	152	26.2	7	8.2	14	✓
divbin	2	5	96	37.7	8	8.7	15	–
manna	1	5	49	19.2	3	5.6	2	✓
hard	2	6	107	14.2	11	9.2	4	–
sqrt1	1	4	27	25.3	3	4.3	1	✓
dijkstra	2	5	61	30.7	8	10.9	6	–
freire1	1	3	25	22.5	2	2.2	0	✓
freire2	1	4	35	26.0	3	5.1	1	✓
cohencb	1	5	31	23.6	4	4.2	1	✓
egcd1	1	8	108	43.1	1	12.8	8	–
egcd2	2	10	209	60.8	8	14.6	12	✓
egcd3	3	12	475	67.0	14	23.4	25	✓
lcm1	3	6	203	38.9	12	14.2	0	✓
lcm2	1	6	52	14.9	1	0.9	10	✓
prodbin	1	5	61	28.3	3	1.1	10	–
prod4br	1	6	42	9.6	4	8.6	7	✓
fermat1	3	5	217	75.7	6	6.2	1	✓
fermat2	1	5	70	25.8	2	5.2	0	✓
knuth	1	8	113	57.1	4	24.6	6	✓
geo1	1	4	25	16.7	2	1.5	4	✓
geo2	1	4	45	24.1	1	2.1	10	✓
geo3	1	5	65	22.1	1	2.7	12	✓
ps2	1	3	25	21.1	2	4.0	0	✓
ps3	1	3	25	21.9	2	4.2	0	✓
ps4	1	3	25	23.5	2	4.9	0	✓
ps5	1	3	24	24.9	2	7.4	0	✓
ps6	1	3	25	25.0	2	69.5	0	✓
total			2392	825.9	118	149	266.3	**22/27**

Summary

Hybrid invariant generation:

- **DIG** employs geometric concepts for dynamic invariant inference
  - Build *max-plus polyhedra* to find disjunctive polynomial invariants
  - Combine max and min relations for expressive power
  - Introduce new classes of *weak* max/min invariants that retain expressiveness and have polynomial time complexity
- **KIP** verifies invariants using program code (k-induction, SMT solving, lemmas learning, redundancy elimination, parallelism)

Results:

- Identify strong enough invariants to prove correctness of *36/41* programs
- Produce *no spurious* results
- Take *2 minutes* per program, on avg, to find and prove invs

Thank you for your attention !

Project (DIG + KIP) is open source and available at

http://cs.unm.edu/~tnguyen

Ask me about (or check my webpage for details)

- Dynamic analysis on complex array invariants (identified 60% of array relations in AES)
- Static analysis on octagonal and max-plus invariants using quantifier elimination
- Automatic program repair using test-input generation (equivalence theorem: program reachability $\equiv$ program synthesis)

Combining Max and Min-plus Invariants

Combining Max and Min-plus Invariants

```
def ex2(x):
  if x ≥ 0:
    y = x + 1
  else:
    y = x - 1
  b = y > 10
  [L]
  return b
```

x	y	b
-50	-51	0
-33	-34	0
9	10	0
10	11	1
12	13	1
40	41	1

Combining Max and Min-plus Invariants

```
def ex2(x):
  if x ≥ 0:
    y = x + 1
  else:
    y = x - 1
  b = y > 10
  [L]
  return b
```

x	y	b
-50	-51	0
-33	-34	0
9	10	0
10	11	1
12	13	1
40	41	1

- By building max and min-plus polyhedra in 3D, DIG obtains

$$1 \geq b \geq 0, \;\; \max(y-10,0) \geq b, \;\; b+10 \geq \min(y,11),$$

i.e.,

$$\begin{aligned} 1 \geq b \geq 0 \;\wedge\; \max(y-10,0) \geq b &\quad\Rightarrow\quad b = 0 \Rightarrow y \leq 10 \\ 1 \geq b \geq 0 \;\wedge\; b+10 \geq \min(y,11) &\quad\Rightarrow\quad b \neq 0 \Rightarrow y > 10 \end{aligned}$$

Combining Max and Min-plus Invariants

```
def ex2(x):
  if x ≥ 0:
    y = x + 1
  else:
    y = x - 1
  b = y > 10
  [L]
  return b
```

x	y	b
-50	-51	0
-33	-34	0
9	10	0
10	11	1
12	13	1
40	41	1

- By building max and min-plus polyhedra in 3D, DIG obtains

$$1 \geq b \geq 0, \ \ \max(y - 10, 0) \geq b, \ \ b + 10 \geq \min(y, 11),$$

i.e.,

$$\begin{array}{lcl} 1 \geq b \geq 0 \ \wedge \ \max(y - 10, 0) \geq b & \Rightarrow & b = 0 \Rightarrow y \leq 10 \\ 1 \geq b \geq 0 \ \wedge \ b + 10 \geq \min(y, 11) & \Rightarrow & b \neq 0 \Rightarrow y > 10 \end{array}$$

- Logically equivalent to the `iff` condition

$$b = 0 \ \Leftrightarrow \ y \leq 10$$