Localizing Widening and Narrowing

Gianluca Amato

Università di Chieti–Pescara

20th Static Analysis Symposium
SAS 2013

(joint work with Francesca Scozzari)

Overview

We will show two techniques to improve precision in AI-based static analysis on nested loops:

Localized widening restricts the scope of application of widening

Localized narrowing intertwines ascending and descending phases for nested loops

These techniques are:

- domain independent
- compatible with other precision-improving techniques
- effective (at least for localized widening)
- new (?)

Overview

We will show two techniques to improve precision in AI-based static analysis on nested loops:

Localized widening restricts the scope of application of widening

Localized narrowing intertwines ascending and descending phases for nested loops

These techniques are:

- domain independent
- compatible with other precision-improving techniques
- effective (at least for localized widening)
- new (?)

Overview

We will show two techniques to improve precision in AI-based static analysis on nested loops:

Localized widening restricts the scope of application of widening

Localized narrowing intertwines ascending and descending phases for nested loops

These techniques are:

- domain independent
- compatible with other precision-improving techniques
- effective (at least for localized widening)
- new (?)

Overview

We will show two techniques to improve precision in AI-based static analysis on nested loops:

Localized widening restricts the scope of application of widening

Localized narrowing intertwines ascending and descending phases for nested loops

These techniques are:

- domain independent
- compatible with other precision-improving techniques
- effective (at least for localized widening)
- new (?)

Overview

We will show two techniques to improve precision in AI-based static analysis on nested loops:

Localized widening restricts the scope of application of widening

Localized narrowing intertwines ascending and descending phases for nested loops

These techniques are:

- domain independent
- compatible with other precision-improving techniques
- effective (at least for localized widening)
- new (?)

Overview

We will show two techniques to improve precision in AI-based static analysis on nested loops:

Localized widening restricts the scope of application of widening

Localized narrowing intertwines ascending and descending phases for nested loops

These techniques are:

- domain independent
- compatible with other precision-improving techniques
- effective (at least for localized widening)
- new (?)

Overview

We will show two techniques to improve precision in AI-based static analysis on nested loops:

Localized widening restricts the scope of application of widening

Localized narrowing intertwines ascending and descending phases for nested loops

These techniques are:

- domain independent
- compatible with other precision-improving techniques
- effective (at least for localized widening)
- new (?)

Disclaimer

An idea similar to *localized narrowing* has been presented yesterday, at PLDI:

K. Apinis, H. Seidl, V. Vojdani
How to Combine Widening and Narrowing for Non-monotonic Systems of Equation

Abstract Interpretation and Equations

In abstract interpretation-based static analysis, the program to analyze is typically translated into a set of equations:

$$\begin{cases} x_1 = \Phi_1(x_1, \ldots, x_n) \\ \quad \vdots \\ x_n = \Phi_n(x_1, \ldots, x_n) \end{cases}$$

- each x_i
  - is the member of an abstract domain
  - corresponds to a program point
- each Φ_i
  - is an abstract transformer
  - corresponds to a node in the control flow graph
- we want a post-fixpoint (possibly the least fixpoint) of this system

How to get non-trivial post-fixpoints

Several techniques:

- standard Kleene iteration
  - may not terminate if the domain is infinite
  - may be slow for finite domains
- policy/strategy iteration
  - powerful technique but only for template domains
- acceleration
  - powerful technique but only for restricted linear assignments
- widening/narrowing

Widening is generally applicable:

- ascending phase using a widening operator which ensures convergence
- an optional descending phase to improve the result

How to get non-trivial post-fixpoints

Several techniques:

- standard Kleene iteration
  - may not terminate if the domain is infinite
  - may be slow for finite domains
- policy/strategy iteration
  - powerful technique but only for template domains
- acceleration
  - powerful technique but only for restricted linear assignments
- widening/narrowing

Widening is generally applicable:

- ascending phase using a widening operator which ensures convergence
- an optional descending phase to improve the result

How to get non-trivial post-fixpoints

Several techniques:

- standard Kleene iteration
  - may not terminate if the domain is infinite
  - may be slow for finite domains
- policy/strategy iteration
  - powerful technique but only for template domains
- acceleration
  - powerful technique but only for restricted linear assignments
- widening/narrowing

Widening is generally applicable:

- ascending phase using a widening operator which ensures convergence
- an optional descending phase to improve the result

How to get non-trivial post-fixpoints

Several techniques:

- standard Kleene iteration
  - may not terminate if the domain is infinite
  - may be slow for finite domains
- policy/strategy iteration
  - powerful technique but only for template domains
- acceleration
  - powerful technique but only for restricted linear assignments
- widening/narrowing

Widening is generally applicable:

- ascending phase using a widening operator which ensures convergence
- an optional descending phase to improve the result

How to get non-trivial post-fixpoints

Several techniques:

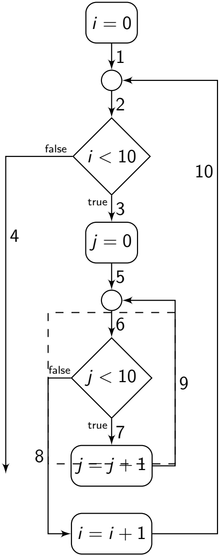
- standard Kleene iteration
  - may not terminate if the domain is infinite
  - may be slow for finite domains
- policy/strategy iteration
  - powerful technique but only for template domains
- acceleration
  - powerful technique but only for restricted linear assignments
- widening/narrowing

Widening is generally applicable:

- ascending phase using a widening operator which ensures convergence
- an optional descending phase to improve the result

From programs to equations

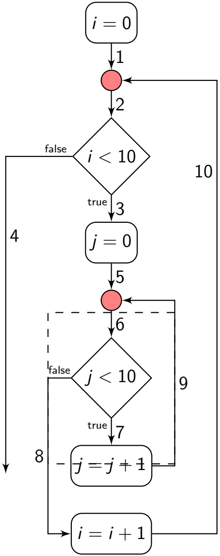
```
i = 0
while (i<10) {
  j = 0
  while (j<10)
    j = j+1
  i = i+1
}
```



$$
\begin{aligned}
x_1 &= [0,0] \times [-\infty, \infty] \\
x_2 &= x_1 \vee x_{10} \\
x_3 &= x_2 \wedge ([-\infty, 9] \times [-\infty, \infty]) \\
x_4 &= x_2 \wedge ([10, \infty] \times [-\infty, \infty]) \\
x_5 &= \mathit{first}(x_3) \times [0,0] \\
x_6 &= x_5 \vee x_9 \\
x_7 &= x_6 \wedge ([-\infty, \infty] \times [-\infty, 9]) \\
x_8 &= x_6 \wedge ([-\infty, \infty] \times [10, \infty]) \\
x_9 &= x_7 + ([0,0] \times [1,1]) \\
x_{10} &= x_8 + ([1,1] \times [0,0])
\end{aligned}
$$

Introducing widening

```
i = 0
while (i<10) {
  j = 0
  while (j<10)
    j = j+1
  i = i+1
}
```



$$x_1 = [0,0] \times [-\infty, \infty]$$

$$x_2 = x_1 \vee x_{10}$$

$$x_3 = x_2 \wedge ([-\infty, 9] \times [-\infty, \infty])$$

$$x_4 = x_2 \wedge ([10, \infty] \times [-\infty, \infty])$$

$$x_5 = \mathit{first}(x_3) \times [0,0]$$

$$x_6 = x_5 \vee x_9$$

$$x_7 = x_6 \wedge ([-\infty, \infty] \times [-\infty, 9])$$

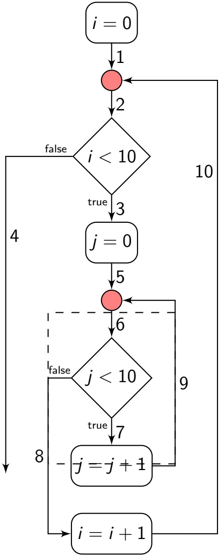
$$x_8 = x_6 \wedge ([-\infty, \infty] \times [10, \infty])$$

$$x_9 = x_7 + ([0,0] \times [1,1])$$

$$x_{10} = x_8 + ([1,1] \times [0,0])$$

Introducing widening

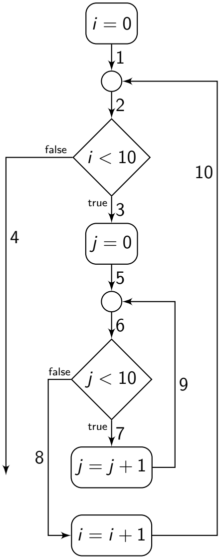
```
i = 0
while (i<10) {
  j = 0
  while (j<10)
    j = j+1
  i = i+1
}
```



$$x_1 = [0,0] \times [-\infty, \infty]$$
$$x_2 = x_2 \,\nabla\, (x_1 \vee x_{10})$$
$$x_3 = x_2 \wedge ([-\infty, 9] \times [-\infty, \infty])$$
$$x_4 = x_2 \wedge ([10, \infty] \times [-\infty, \infty])$$
$$x_5 = \mathit{first}(x_3) \times [0,0]$$
$$x_6 = x_6 \,\nabla\, (x_5 \vee x_9)$$
$$x_7 = x_6 \wedge ([-\infty, \infty] \times [-\infty, 9])$$
$$x_8 = x_6 \wedge ([-\infty, \infty] \times [10, \infty])$$
$$x_9 = x_7 + ([0,0] \times [1,1])$$
$$x_{10} = x_8 + ([1,1] \times [0,0])$$

Problems with nested loops

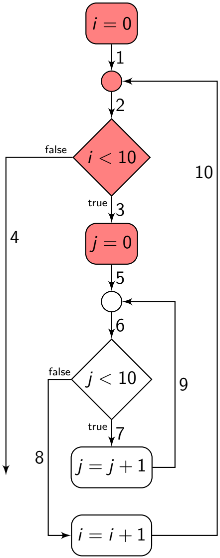
Ascending chain



$$
\begin{aligned}
x_1 &= \bot \\
x_2 &= \bot \\
x_3 &= \bot \\
x_4 &= \bot \\
x_5 &= \bot \\
x_6 &= \bot \\
x_7 &= \bot \\
x_8 &= \bot \\
x_9 &= \bot \\
x_{10} &= \bot
\end{aligned}
$$

Problems with nested loops

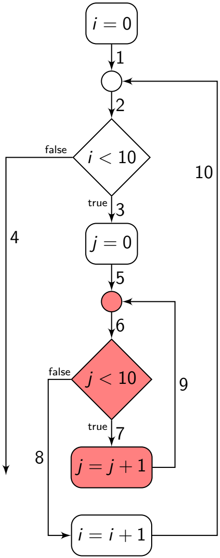
Ascending chain



$$
\begin{aligned}
{\color{red}x_1} &= \{i = 0\} \\
{\color{red}x_2} &= \{i = 0\} \\
{\color{red}x_3} &= \{i = 0\} \\
{\color{red}x_4} &= \bot \\
{\color{red}x_5} &= \{i = 0, j = 0\} \\
x_6 &= \bot \\
x_7 &= \bot \\
x_8 &= \bot \\
x_9 &= \bot \\
x_{10} &= \bot
\end{aligned}
$$

Problems with nested loops

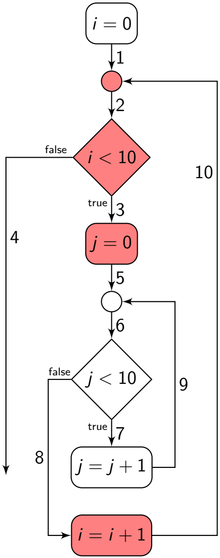
Ascending chain



$$x_1 = \{i = 0\}$$
$$x_2 = \{i = 0\}$$
$$x_3 = \{i = 0\}$$
$$x_4 = \bot$$
$$x_5 = \{i = 0, j = 0\}$$
$$x_6 = \{i = 0, 0 \leq j\}$$
$$x_7 = \{i = 0, 0 \leq j \leq 9\}$$
$$x_8 = \{i = 0, 10 \leq j\}$$
$$x_9 = \{i = 0, 1 \leq j \leq 10\}$$
$$x_{10} = \bot$$

Problems with nested loops

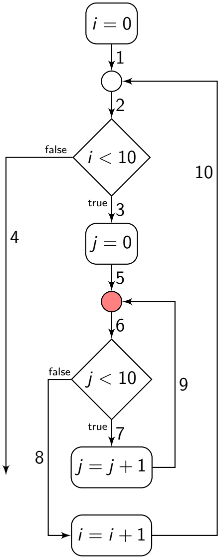
Ascending chain



$$
\begin{aligned}
x_1 &= \{i = 0\} \\
{\color{red}x_2} &= \{0 \leq i\} \\
{\color{red}x_3} &= \{0 \leq i \leq 9\} \\
{\color{red}x_4} &= \{10 \leq i\} \\
{\color{red}x_5} &= \{0 \leq i \leq 9, j = 0\} \\
x_6 &= \{i = 0, 0 \leq j\} \\
x_7 &= \{i = 0, 0 \leq j \leq 9\} \\
x_8 &= \{i = 0, 10 \leq j\} \\
x_9 &= \{i = 0, 1 \leq j \leq 10\} \\
{\color{red}x_{10}} &= \{i = 1, 10 \leq j\}
\end{aligned}
$$

Problems with nested loops

Ascending chain

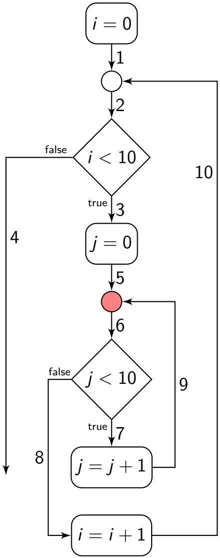


$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \leq i\} \\
x_3 &= \{0 \leq i \leq 9\} \\
x_4 &= \{10 \leq i\} \\
x_5 &= \{0 \leq i \leq 9, j = 0\} \\
x_6 &= \{i = 0, 0 \leq j\} \\
x_7 &= \{i = 0, 0 \leq j \leq 9\} \\
x_8 &= \{i = 0, 10 \leq j\} \\
x_9 &= \{i = 0, 1 \leq j \leq 10\} \\
x_{10} &= \{i = 1, 10 \leq j\}
\end{aligned}
$$

$$\boxed{x_6 = x_6 \triangledown (x_5 \cup x_9)}$$

Problems with nested loops

Ascending chain

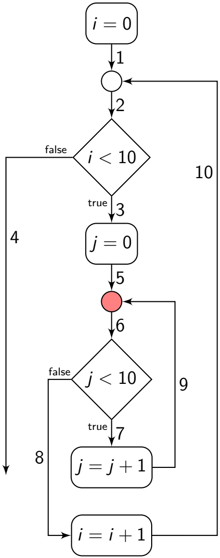


$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \leq i\} \\
x_3 &= \{0 \leq i \leq 9\} \\
x_4 &= \{10 \leq i\} \\
x_5 &= \{0 \leq i \leq 9, j = 0\} \\
x_6 &= \{i = 0, 0 \leq j\} \\
x_7 &= \{i = 0, 0 \leq j \leq 9\} \\
x_8 &= \{i = 0, 10 \leq j\} \\
x_9 &= \{i = 0, 1 \leq j \leq 10\} \\
x_{10} &= \{i = 1, 10 \leq j\}
\end{aligned}
$$

$$\boxed{x_6 = \{i = 0, 0 \leq j\} \triangledown \{0 \leq i \leq 9, 0 \leq j \leq 10\}}$$

Problems with nested loops

Ascending chain

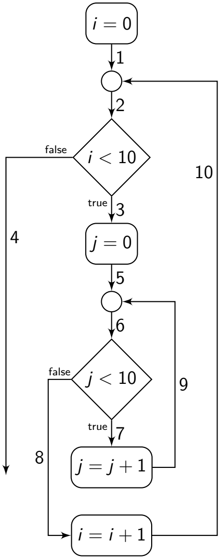


$$x_1 = \{i = 0\}$$
$$x_2 = \{0 \le i\}$$
$$x_3 = \{0 \le i \le 9\}$$
$$x_4 = \{10 \le i\}$$
$$x_5 = \{0 \le i \le 9, j = 0\}$$
$$x_6 = \{i = 0, 0 \le j\}$$
$$x_7 = \{i = 0, 0 \le j \le 9\}$$
$$x_8 = \{i = 0, 10 \le j\}$$
$$x_9 = \{i = 0, 1 \le j \le 10\}$$
$$x_{10} = \{i = 1, 10 \le j\}$$

$$\boxed{x_6 = \{0 \le i, 0 \le j\}}$$

Problems with nested loops

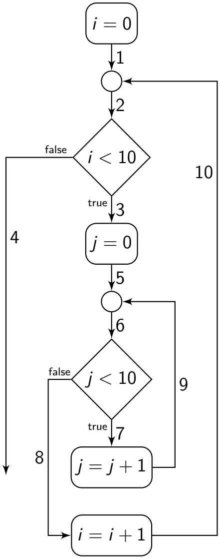
Result of ascending chain



$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \leq i\} \\
x_3 &= \{0 \leq i \leq 9\} \\
x_4 &= \{10 \leq i\} \\
x_5 &= \{0 \leq i \leq 9, j = 0\} \\
x_6 &= \{0 \leq i, 0 \leq j\} \\
x_7 &= \{0 \leq i, 0 \leq j \leq 9\} \\
x_8 &= \{0 \leq i, 10 \leq j\} \\
x_9 &= \{0 \leq i, 1 \leq j \leq 10\} \\
x_{10} &= \{1 \leq i, 10 \leq j\}
\end{aligned}
$$

Problems with nested loops

Result of descending chain

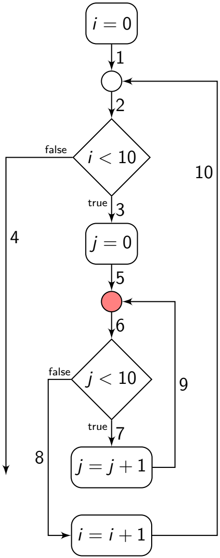


$$x_1 = \{i = 0\}$$
$$x_2 = \{0 \le i\}$$
$$x_3 = \{0 \le i \le 9\}$$
$$x_4 = \{10 \le i\}$$
$$x_5 = \{0 \le i \le 9, j = 0\}$$
$$x_6 = \{0 \le i, 0 \le j \le 10\}$$
$$x_7 = \{0 \le i, 0 \le j \le 9\}$$
$$x_8 = \{0 \le i, 10 \le j \le 10\}$$
$$x_9 = \{0 \le i, 1 \le j \le 10\}$$
$$x_{10} = \{1 \le i, 10 \le j \le 10\}$$

We cannot prove $i = 10$ in x_4

Problems with nested loops

A step back

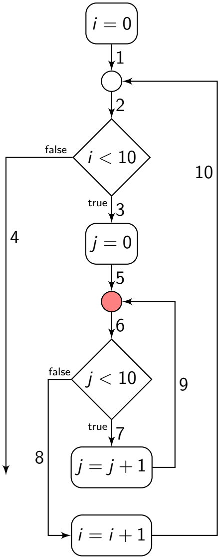


$$x_1 = \{i = 0\}$$
$$x_2 = \{0 \leq i\}$$
$$x_3 = \{0 \leq i \leq 9\}$$
$$x_4 = \{10 \leq i\}$$
$$x_5 = \{0 \leq i \leq 9, j = 0\}$$
$$x_6 = \{i = 0, 0 \leq j\}$$
$$x_7 = \{i = 0, 0 \leq j \leq 9\}$$
$$x_8 = \{i = 0, 10 \leq j\}$$
$$x_9 = \{i = 0, 1 \leq j \leq 10\}$$
$$x_{10} = \{i = 1, 10 \leq j\}$$

$$\boxed{x_6 = x_6 \triangledown (x_5 \cup x_9)}$$

Localized widening

What is localized widening?

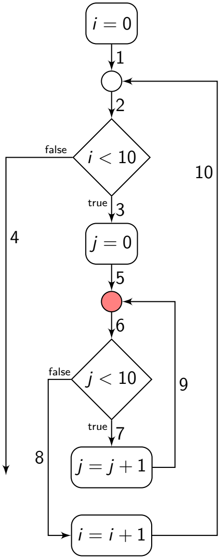


$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \leq i\} \\
x_3 &= \{0 \leq i \leq 9\} \\
x_4 &= \{10 \leq i\} \\
x_5 &= \{0 \leq i \leq 9, j = 0\} \\
x_6 &= \{i = 0, 0 \leq j\} \\
x_7 &= \{i = 0, 0 \leq j \leq 9\} \\
x_8 &= \{i = 0, 10 \leq j\} \\
x_9 &= \{i = 0, 1 \leq j \leq 10\} \\
x_{10} &= \{i = 1, 10 \leq j\}
\end{aligned}
$$

$$\boxed{x_6 = x_6 \nabla (x_5 \cup x_9)} \Longrightarrow \boxed{x_6 = x_5 \cup (x_6 \nabla x_9)}$$

Localized widening

What is localized widening?

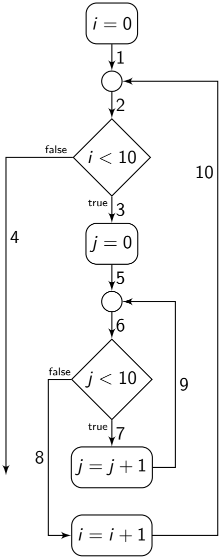


$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \le i\} \\
x_3 &= \{0 \le i \le 9\} \\
x_4 &= \{10 \le i\} \\
x_5 &= \{0 \le i \le 9, j = 0\} \\
x_6 &= \{i = 0, 0 \le j\} \\
x_7 &= \{i = 0, 0 \le j \le 9\} \\
x_8 &= \{i = 0, 10 \le j\} \\
x_9 &= \{i = 0, 1 \le j \le 10\} \\
x_{10} &= \{i = 1, 10 \le j\}
\end{aligned}
$$

$$\boxed{x_6 = \{0 \le i \le 9, 0 \le j\}}$$

Localized widening

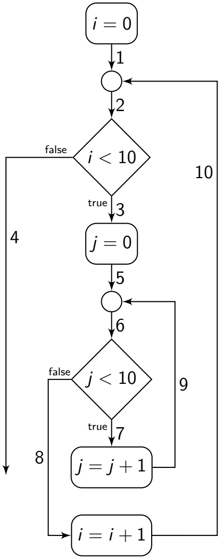
Result of ascending chain with localized widening



$$x_1 = \{i = 0\}$$
$$x_2 = \{0 \leq i\}$$
$$x_3 = \{0 \leq i \leq 9\}$$
$$x_4 = \{10 \leq i\}$$
$$x_5 = \{0 \leq i \leq 9, j = 0\}$$
$$x_6 = \{0 \leq i \leq 9, 0 \leq j\}$$
$$x_7 = \{0 \leq i \leq 9, 0 \leq j \leq 9\}$$
$$x_8 = \{0 \leq i \leq 9, 10 \leq j\}$$
$$x_9 = \{0 \leq i \leq 9, 1 \leq j \leq 10\}$$
$$x_{10} = \{1 \leq i \leq 10, 10 \leq j\}$$

Localized widening

Result of descending chain with localized widening



$$x_1 = \{i = 0\}$$
$$x_2 = \{0 \leq i \leq 10\}$$
$$x_3 = \{0 \leq i \leq 9\}$$
$$x_4 = \{10 \leq i \leq 10\}$$
$$x_5 = \{0 \leq i \leq 9, j = 0\}$$
$$x_6 = \{0 \leq i \leq 9, 0 \leq j \leq 10\}$$
$$x_7 = \{0 \leq i \leq 9, 0 \leq j \leq 9\}$$
$$x_8 = \{0 \leq i \leq 9, 10 \leq j \leq 10\}$$
$$x_9 = \{0 \leq i \leq 9, 1 \leq j \leq 10\}$$
$$x_{10} = \{1 \leq i \leq 10, 10 \leq j \leq 10\}$$

We can prove $i = 10$ in x_4

Localized Widening

Definition (informal)

We call *localized widening* the use of a widening in the form

$$x_i = x_{in} \vee (x_i \triangledown x_{back})$$

instead of

$$x_i = x_i \triangledown (x_{in} \vee x_{back})$$

where

- x_{in} is the value of the edge(s) coming from the outer component;
- x_{back} is the value of the back edge(s) of the loop.

Theorem

Using localized widening, every chaotic iteration sequence terminates on a post-fixpoint (if every loop head is a widening point).

Localized Widening

Definition (informal)

We call *localized widening* the use of a widening in the form

$$x_i = x_{in} \vee (x_i \nabla x_{back})$$

instead of

$$x_i = x_i \nabla (x_{in} \vee x_{back})$$

where

- x_{in} is the value of the edge(s) coming from the outer component;
- x_{back} is the value of the back edge(s) of the loop.

Theorem

Using localized widening, every chaotic iteration sequence terminates on a post-fixpoint (if every loop head is a widening point).

Is it really new?

The optimization given by localized widening is very very simple. Is it really new?

We cannot be sure it has not been implemented before, but we think it has not been described before.

We are sure both INTERPROC and PAGAI do not implement localized widening.

We implemented it in RANDOM long before this paper.

Is it really new?

The optimization given by localized widening is very very simple. Is it really new?

We cannot be sure it has not been implemented before, but we think it has not been described before.

We are sure both INTERPROC and PAGAI do not implement localized widening.

We implemented it in RANDOM long before this paper.

Is it really new?

The optimization given by localized widening is very very simple. Is it really new?

We cannot be sure it has not been implemented before, but we think it has not been described before.

We are sure both INTERPROC and PAGAI do not implement localized widening.

We implemented it in RANDOM long before this paper.

Is it really new?

The optimization given by localized widening is very very simple. Is it really new?

We cannot be sure it has not been implemented before, but we think it has not been described before.

We are sure both Interproc and Pagai do not implement localized widening.

We implemented it in Random long before this paper.

Is it really worth

Is it really worth to implement it? Definitively yes.

- It is very simple.
  Our implementation in PAGAI, apart from stuff related to configuration and user interface, only requires to change a single line of code.
- Nested loops are common.
  Most programs benefits from the improved precision.
- Computational overhead should be negligible.
  Hard to say something more precise, due to non-monotonicity of widening.

Is it really worth

Is it really worth to implement it? Definitively yes.

- It is very simple.
  Our implementation in PAGAI, apart from stuff related to configuration and user interface, only requires to change a single line of code.
- Nested loops are common.
  Most programs benefits from the improved precision.
- Computational overhead should be negligible.
  Hard to say something more precise, due to non-monotonicity of widening.

Is it really worth

Is it really worth to implement it? Definitively yes.

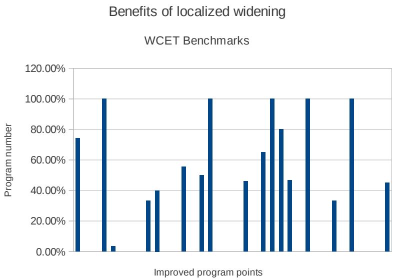
- It is very simple.
  Our implementation in PAGAI, apart from stuff related to configuration and user interface, only requires to change a single line of code.
- Nested loops are common.
  Most programs benefits from the improved precision.
- Computational overhead should be negligible.
  Hard to say something more precise, due to non-monotonicity of widening.

Is it really worth

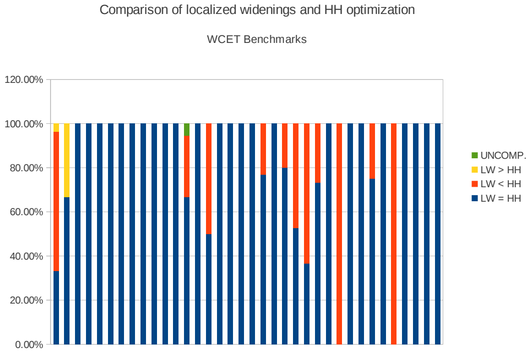
Is it really worth to implement it? Definitively yes.

- It is very simple.
  Our implementation in PAGAI, apart from stuff related to configuration and user interface, only requires to change a single line of code.
- Nested loops are common.
  Most programs benefits from the improved precision.
- Computational overhead should be negligible.
  Hard to say something more precise, due to non-monotonicity of widening.

We implemented localized widening in `Pagai` and compared it with standard widening.

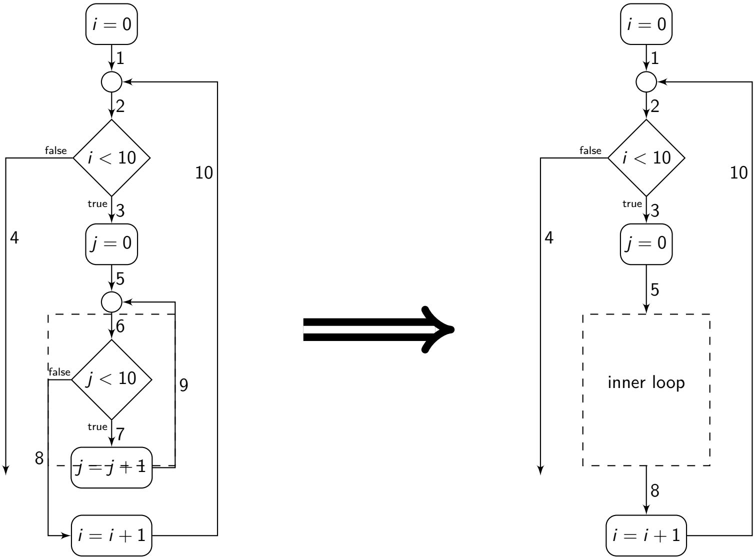


- total: 379 nodes, improvements in 164 nodes
- standard widening: 4.48 secs, 19522 asc. steps and 23415 desc. steps
- localized widening: 5.22 secs, 19363 asc. steps and 22528 desc. steps

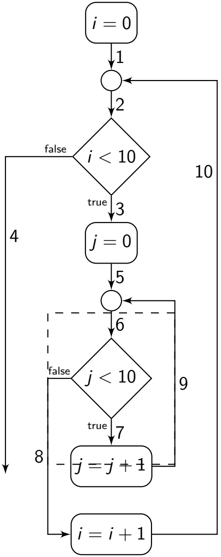


- 91 improved head nodes, 2 regressed, 2 incomparable
- localized widening: 5.26 secs, 19363 asc. steps and 22528 desc. steps
- hh narrowing: 5.88 secs, 22665 asc. steps and 27134 desc. steps

Loops as black boxes

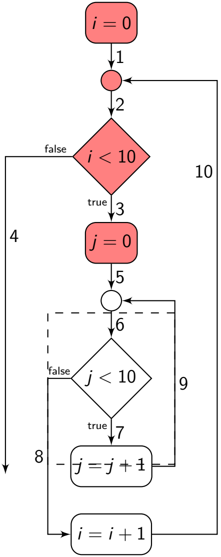


Iteration with localized narrowing



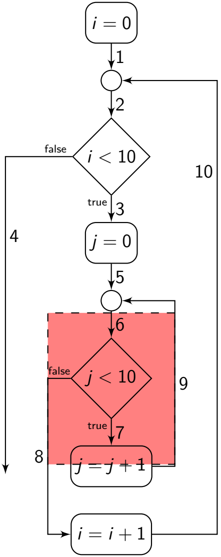
$$
\begin{aligned}
x_1 &= \bot \\
x_2 &= \bot \\
x_3 &= \bot \\
x_4 &= \bot \\
x_5 &= \bot \\
x_6 &= \bot \\
x_7 &= \bot \\
x_8 &= \bot \\
x_9 &= \bot \\
x_{10} &= \bot
\end{aligned}
$$

Iteration with localized narrowing



$$x_1 = \{i = 0\}$$
$$x_2 = \{i = 0\}$$
$$x_3 = \{i = 0\}$$
$$x_4 = \perp$$
$$x_5 = \{i = 0, j = 0\}$$
$$x_6 = \perp$$
$$x_7 = \perp$$
$$x_8 = \perp$$
$$x_9 = \perp$$
$$x_{10} = \perp$$

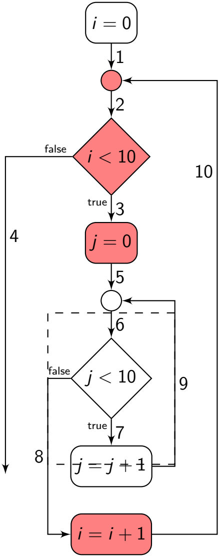
Iteration with localized narrowing



$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{i = 0\} \\
x_3 &= \{i = 0\} \\
x_4 &= \bot \\
x_5 &= \{i = 0, j = 0\} \\
{\color{red}x_6} &= \{i = 0, 0 \leq j \leq 10\} \\
{\color{red}x_7} &= \{i = 0, 0 \leq j \leq 9\} \\
{\color{red}x_8} &= \{i = 0, j = 10\} \\
{\color{red}x_9} &= \{i = 0, 1 \leq j \leq 10\} \\
x_{10} &= \bot
\end{aligned}
$$

Ascending and descending chain

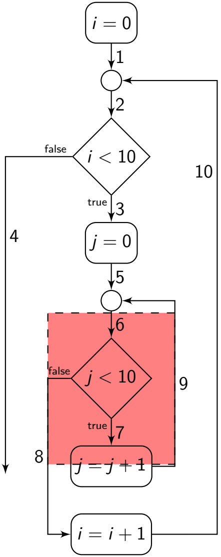
Iteration with localized narrowing



$$x_1 = \{i = 0\}$$
$$x_2 = \{0 \le i\}$$
$$x_3 = \{0 \le i \le 9\}$$
$$x_4 = \{10 \le i\}$$
$$x_5 = \{0 \le i \le 9, j = 0\}$$
$$x_6 = \{i = 0, 0 \le j \le 10\}$$
$$x_7 = \{i = 0, 0 \le j \le 9\}$$
$$x_8 = \{i = 0, j = 10\}$$
$$x_9 = \{i = 0, 1 \le j \le 10\}$$
$$x_{10} = \{i = 1, j = 10\}$$

What to do with $x_6 - x_9$?

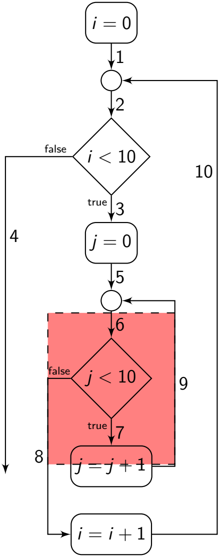
Iteration with localized narrowing



$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \le i\} \\
x_3 &= \{0 \le i \le 9\} \\
x_4 &= \{10 \le i\} \\
x_5 &= \{0 \le i \le 9, j = 0\} \\
x_6 &= \bot \\
x_7 &= \bot \\
x_8 &= \bot \\
x_9 &= \bot \\
x_{10} &= \{i = 1, j = 10\}
\end{aligned}
$$

Start with $x_6 = x_7 = x_8 = x_9 = \bot \ldots$

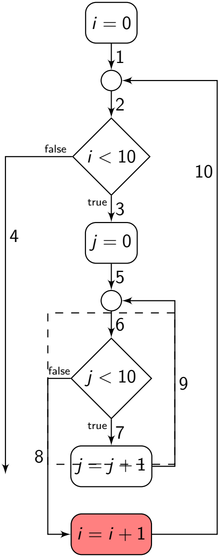
Iteration with localized narrowing



$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \leq i\} \\
x_3 &= \{0 \leq i \leq 9\} \\
x_4 &= \{10 \leq i\} \\
x_5 &= \{0 \leq i \leq 9, j = 0\} \\
{\color{red}x_6} &= \{0 \leq i \leq 9, 0 \leq j \leq 10\} \\
{\color{red}x_7} &= \{0 \leq i \leq 9, 0 \leq j \leq 9\} \\
{\color{red}x_8} &= \{0 \leq i \leq 9, j = 10\} \\
{\color{red}x_9} &= \{0 \leq i \leq 9, 1 \leq j \leq 10\} \\
x_{10} &= \{i = 1, j = 10\}
\end{aligned}
$$

. . . and re-analyze

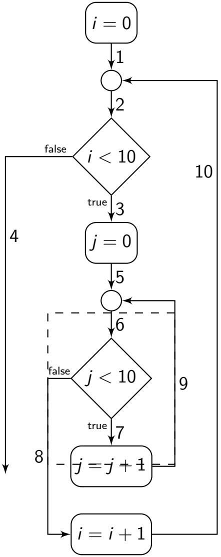
Iteration with localized narrowing



$$
\begin{aligned}
x_1 &= \{i = 0\} \\
x_2 &= \{0 \le i\} \\
x_3 &= \{0 \le i \le 9\} \\
x_4 &= \{10 \le i\} \\
x_5 &= \{0 \le i \le 9, j = 0\} \\
x_6 &= \{0 \le i \le 9, 0 \le j \le 10\} \\
x_7 &= \{0 \le i \le 9, 0 \le j \le 9\} \\
x_8 &= \{0 \le i \le 9, j = 10\} \\
x_9 &= \{0 \le i \le 9, 1 \le j \le 10\} \\
x_{10} &= \{1 \le i \le 10, j = 10\}
\end{aligned}
$$

End of external ascending sequence

Iteration with localized narrowing



$$x_1 = \{i = 0\}$$
$$x_2 = \{0 \leq i \leq 10\}$$
$$x_3 = \{0 \leq i \leq 9\}$$
$$x_4 = \{i = 10\}$$
$$x_5 = \{0 \leq i \leq 9, j = 0\}$$
$$x_6 = \{0 \leq i \leq 9, 0 \leq j \leq 10\}$$
$$x_7 = \{0 \leq i \leq 9, 0 \leq j \leq 9\}$$
$$x_8 = \{0 \leq i \leq 9, j = 10\}$$
$$x_9 = \{0 \leq i \leq 9, 1 \leq j \leq 10\}$$
$$x_{10} = \{1 \leq i \leq 10, j = 10\}$$

End of external descending sequence

Localized widening vs localized narrowing

In this case, both techniques give the same result, but in general?
Due to non-monotonicity, it is difficult to actually prove something, but in general, localized narrowing is more precise.

```
i = 0
while (TRUE) {
  i = i+1
  j = 0
  while (j<10) {
    // Inv: 0 ≤ i ≤ 10
    j = j+1
  }
  if (i>9) i = 0
}
```

Localized narrowing proves the required invariant

Localized widening doesn't because the check for the upper bound of i is after the inner loop.

Initialization policies strategies

When we restart the analysis of an inner loop, which values to choose for the data flow variables ?

- `RESTART` policy: start from $\perp$ (the one we used in the example)
- `CONTINUE` policy: start from last values
- `HYBRID` policy:
  - behave as `CONTINUE` if outer loop is in ascending phase;
  - behave as `RESTART` if outer loop is in descending phase

Problems with CONTINUE policy

The `CONTINUE` policy is faster, but less precise.

```
i = 0
while (TRUE) {
  i = i+1
  j = 0
  while (j<10) {
    // Inv: 0 ≤ i ≤ 10
    j = j+1
  }
  if (i>9) i = 0
}
```

RESTART policy proves the required invariant

CONTINUE policy doesn't because the check for the upper bound of i is after the inner loop.

Initialization policies strategies

When we restart the analysis of an inner loop, which values to choose for the data flow variables ?

- `RESTART` policy: start from $\perp$ (the one we used in the example)
- `CONTINUE` policy: start from last values
- `HYBRID` policy:
  - behave as `CONTINUE` if outer loop is in ascending phase;
  - behave as `RESTART` if outer loop is in descending phase

Localized narrowing and termination

Localized widening works for any iteration strategy.

Localized narrowing is based on a recursive iteration strategy as defined in Burdoncle '93 but:

- different loops may be in different phases
- for two nested loops, all possible combinations of ascending and descending phases are possible

Experiments

- We have implement localized narrowing in our analyzer JANDOM
  - JANDOM is an abstract interpreter for Java bytecode and other languages (`https://github.com/jandom-devel/Jandom`)
  - ... but it is not yet ready to analyze real code
- Localized narrowing has not been implemented in PAGAI
- ... hence, we have only limited qualitative comparison of precision with other techniques

Four programs

nested

```
i = 0
while (i <10) {
   j = 0
   while (j <10)
      j = j+1
   i = i+1
}
// Inv: i = 10
```

hybrid

```
i = 0
while (TRUE) {
   i = i+1
   j = 0
   while (j <10)
      // Inv: 0 ≤ i ≤ 10
      j = j+1
   if (i >9) i = 0
}
```

hh (Halbwachs & Henry)

```
i = 0
while (i <4)  {
   j = 0
   while (j <4)  {
      // Inv:  i ≤ j + 3
      i = i+1
      j = j+1
   }
   i = i−j+1
}
```

nested2

```
i = 0
while (TRUE) {
   // Inv:  i ≥ 0
   j = 0
   while (j <10) {
      j = j+1
   }
   i = i+11−j
}
```

Results on the four programs

program	loc widening	continue	hybrid	guided	lookahead	HH
`nested`	yes	yes	yes	no	no	yes
`nested2`	no	yes	yes	no	yes	no
`hybrid`	no	no	yes	no	no	yes
`hh`	yes	no	yes	no	no	yes

loc widening: localized widening

continue: localized narrowing with CONTINUE policy

hybrid: localized narrowing with HYBRID policy

guided: guided static analysis

lookahead: lookahead widening

HH: optimized narrowing in Halbwachs & Henry 2012

Conclusions

1. *localized widening*: do not see any reason not to implement it
   - easy to implement
   - good precision improvement
   - low overhead
2. *localized narrowing*: more work needed to establish its usefulness
   - implementation from easy (for AST-based analyzers) to medium difficulty
   - better than localized widening, but how much better?
   - what overhead?

Results in PLDI '13 paper by Apinis, Seidl, Vojdani are encouraging.

Future work

1. Evaluate benefits of localized narrowing
   - Implement localized narrowing in PAGAI
   - Finish implementation of Java bytecode analysis in JANDOM
2. Merge ideas from this paper and the PLDI '13 paper by Apinis, Seidl, Vojdani.