Efficiently intertwining widening and narrowing

Kalmer Apinis    Helmut Seidl    Vesal Vojdani
Gianluca Amato    Francesca Scozzari
The Plan

- Static analysis à la Bourdoncle\(^1\)
- *Localized* Widening & Narrowing\(^2\)
- Static analysis à la Goblint
- Adaptation of Localized Widening & Narrowing
- Conclusion

\(^1\) Efficient chaotic iteration strategies with widenings, Bourdoncle
\(^2\) Localizing widening and narrowing, Amato&Scozzari
Bourdonacle

1. AST → dependency graph + equation system

```
① x := 0;
② while x < 10 do begin
③ y := 0;
④ while y < x do
⑤ y := y + 1
⑥ end;
⑦ end;
```
**Bourdoncle**

1. AST → dependency graph + equation system

\[
\begin{align*}
\text{x} & := 0;
\text{while } \text{x} < 10 \text{ do begin} \\
\text{y} & := 0;
\text{while } \text{y} < \text{x} \text{ do} \\
\text{y} & := \text{y} + 1
\text{end;}
\end{align*}
\]

\[
\begin{align*}
\begin{cases}
\text{x}_1 = \text{start} \\
\text{x}_2 = (\text{x} := 0)^\# \text{x}_1 \\
\vdots \\
\text{x}_8 = (\text{x} \geq 10)^\# \text{x}_2 \sqcup (\text{x} \geq 10)^\# \text{x}_7
\end{cases}
\end{align*}
\]

- control points → equation system variables
- transitions → right-hand sides
Bourdoncle (cont.)

2. dependency graph → w.t.o. → iteration strategy

**hierarchical ordering**

A hierarchical ordering of a set is a well-parenthesized permutation of this set without two consecutive ‘(’.

Example: $1 \ 2 \ (3 \ 4 \ (5 \ 6) \ 7) \ 8$, $\omega(6) = \{5, 3\}$

**weak topological ordering**

A weak topological ordering of a directed graph (w.t.o. for short) is a hierarchical ordering of its vertices such that for every edge $u \rightarrow v$:

$$(u < v \land v \notin \omega(u)) \lor (v \preceq u \land v \in \omega(u))$$
Recursively iterate based on the w.t.o.

- **State:**
  - variable assignment
  - set of stable variables

- **Example:** 1 2 [3 4 [5 6]* 7]* 8

\[
\begin{align*}
    x_1 &= \text{start} \\
    x_2 &= \lceil x := 0 \rceil^\# x_1 \\
    \vdots \\
    x_8 &= (\lceil x \geq 10 \rceil^\# x_2) \sqcup (\lceil x \geq 10 \rceil^\# x_7)
\end{align*}
\]
Interval Domain Example

\[
x = 0;
\]

1. while \((x \leq 100)\)
2. \(x++;\)
3. \(x++;\)
Interval Domain Example

\begin{align*}
x &= 0; \\
\text{1. } &\textbf{while } (x \leq 100) \\
\text{2. } &x++; \\
\text{3. } &
\end{align*}

\begin{align*}
x_1 &= [0, 0] \cup (x_2 + [1, 1]) \\
x_2 &= x_1 \cap [-\infty, 100] \\
x_3 &= x_1 \cap [101, \infty]
\end{align*}
Interval Domain Example

\[ x = 0; \]
\[ \text{① while} \ (x \leq 100) \]
\[ \text{②} \ x++; \]
③

Iteration strategy: \[ [1 \ 2]^* \ 3 \quad \rightarrow \quad x_1 = [0, 101] \]

\[
\begin{align*}
  \chi_1 &= [0, 0] \sqcup (\chi_2 + [1, 1]) \\
  \chi_2 &= \chi_1 \cap [\infty, 100] \\
  \chi_3 &= \chi_1 \cap [101, \infty]
\end{align*}
\]
Interval Domain Example

\[
x = 0;
\]

1. `while (x <= 100)`
2. `x++;

Iteration strategy: \([1 \ 2]^* \ 3\)  
\[
\begin{align*}
\chi_1 &= [0, 0] \sqcup (\chi_2 + [1, 1]) \\
\chi_2 &= \chi_1 \cap [-\infty, 100] \\
\chi_3 &= \chi_1 \cap [101, \infty]
\end{align*}
\]

\[
\chi_1 = [0, 101]
\]

Takes too many iterations!
Interval Domain Example

\[ x = 0; \]

1. \textbf{while} (x <= 100)
2. \textbf{x++};

Iteration strategy: \([1 \ 2]^* \ 3 \rightarrow x_1 = [0, 101]\]

Takes too many iterations!

Solution: make component heads widening points!
Widening intervals

\[
\begin{align*}
\text{x} & = 0; \\
\text{① while} & \ (x \leq 100) \\
\text{② } x & ++;
\end{align*}
\]

Iteration strategy: \([1, 2]^* \ 3 \rightarrow x_1 = [0, \infty]\)

**Widening:**

\(\sqcup\) — makes increasing chains stabilize in finite steps.

E.g., \([0, 0] \sqcup [0, 1] = [0, \infty]\)
Widening intervals

\[
x = 0;\\n\textbf{1. while} (x \leq 100)\\n\quad \textbf{2. } x++;\\n\textbf{3. }
\]

Iteration strategy: \([1 \ 2]^* \ 3 \rightarrow x_1 = [0, \infty]\)

Widening:

\[\sqcup \text{ — makes increasing chains stabilize in finite steps.}\]

E.g., \([0, 0] \sqcup [0, 1] = [0, \infty]\)

Bourdoncle: “\ldots, narrowing operators can be used to improve the post-fixed points \ldots”. But how?”
Intertwined widening and narrowing.

- Examples
  - \([1 \ 2]_w^\ast \ [1 \ 2]_n^\ast \ 3\)
  - \([3 \ 4 \ [5 \ 6]_w^\ast \ [5 \ 6]_n^\ast \ 7]_w^\ast \ [3 \ 4 \ [5 \ 6]_w^\ast \ [5 \ 6]_n^\ast \ 7]_n^\ast \ 8\)

- Iterate widening until stabilization.
- Iterate narrowing “a few times”.

Termination for monotonic right-hand sides proven!
Intertwining W/N Example

\[ x = 0; \]

1. while (x <= 100)
   2. x++;
3. 

\[
\begin{align*}
\chi_1 &= \chi_1 \square ([0, 0] \sqcup (\chi_2 + [1, 1])) \\
\chi_2 &= \chi_1 \sqcap [-\infty, 100] \\
\chi_3 &= \chi_1 \sqcap [101, \infty]
\end{align*}
\]

Iteration strategy: \([1 \ 2]^*_w \ [1 \ 2]^*_n \ 3 \rightarrow \chi_1 = [0, 101]\)
Localized Widening:

- Replace

\[ x = x \sqcup (\text{in } \sqcap \text{back}) \]

with

\[ x = \text{in } \sqcup (x \sqcup \text{back}) \]
Localized Widening Example

Example

```plaintext
i = 0;
while (i < 10) {
    j = 0;
    while (j < 10) {
        // 0 ≤ i < 10
        j = j + 1;
    }
    i = i + 1;
}
```
Amato & Scozzari: Idea 2

Localized Narrowing

- Reset the loop body after (each) update to loop head.

Example: $[1 \ 2]_w^*$, $[1 \ R_2 \ 2]_n^* \ 3$
Amato&Scozzari: Conclusion

- First classical *concrete* description on “intertwining widening and narrowing”.
- Interesting optimizations: first — easy, second — general.
Amato & Scozzari: Conclusion

- (First) (classical) *concrete* description on “intertwining widening and narrowing”.

- Interesting optimizations: first — easy, second — general.
Goblint

Differences:

- Infinite systems — cannot (pre)compute everything.
- Dynamic deps. — do not want to over-approximate
- Uses demand-driven solving

Generalize the ideas —

- similar effect for examples, and
- correctness generally.
The problem in detail

\[ x = f(a, b, c) \]

Questions that need answers:

- How to find component heads?
- How to find back edges?
- How to find loop nodes?
Loop detection

- Label nodes with increasing numbers (from the back).
- Edge to a bigger number — loop.
  - Starting node is the loop head.

Problem: detection at the wrong edge.
Back-edge detection

By example:

- Mark 2 for widening any time 4 is updated
- Remove mark after recomputing 2
Loop body detection

Dynamic loop body detection:
Nodes with larger label that influence the loop head. (Loop head has the smallest label in the loop)
Conclusion

• Not solved — fine control on when to restart.
• Small examples work as precisely as Amato&Scozzari
• Works with dynamic deps. & infinite eq. systems.
• Restarting is computationally expensive.
  (also in Amato&Scozzari)