Combining Widening and Narrowing for Non-monotonic Systems of Equations

Helmut Seidl
Vesal Vojdani
Kalmer Apinis

PUMA Lecture, May 31, 2013
Observation 1  Cousot, Cousot 1977

- Program invariants are post-solutions of systems of equations over suitable lattices.
Observation 1

- Program invariants are \textit{post-solutions} of systems of equations over suitable lattices.
- Non-trivial program invariants require lattices with \textit{infinite} ascending chains.
Observation 1

Cousot, Cousot 1977

- Program invariants are post-solutions of systems of equations over suitable lattices.
- Non-trivial program invariants require lattices with infinite ascending chains.
- Kleene fixpoint iteration will often not terminate.
Observation 1

• Program invariants are post-solutions of systems of equations over suitable lattices.

• Non-trivial program invariants require lattices with infinite ascending chains.

• Kleene fixpoint iteration will often not terminate.

• Widening allows to enforce termination—at the price of giving up precision.
Observation 1  Cousot, Cousot 1977

- Program invariants are post-solutions of systems of equations over suitable lattices.
- Non-trivial program invariants require lattices with infinite ascending chains.
- Kleene fixpoint iteration will often not terminate.
- **Widening** allows to enforce termination—at the price of giving up precision.
- Some of the precision subsequently can may be recovered through **narrowing** ...
Example

Example:

\[ i < 20 \rightarrow i = 0; \]
\[ i < 10 \rightarrow x++; \]
\[ i \geq 10 \rightarrow i = 0; x = 0; \]
\[ i \geq 20 \rightarrow i++; \]
\[ i \geq 20 \rightarrow i++; \]
Example Analysis

reduced product of

• intervals
• linear equalities

⇒ infinite ascending chains
⇒ widening/narrowing required
Widening

Equations

\[ x_i = f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]
Widening

Equations

\[ x_i = x_i \sqcup f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]
Widening

Equations

\[ x_i = x_i \sqcup f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]

Widening operator

\[ a \sqcup b \subseteq a \uplus b \]
Widening

0 \leq i, \quad 0 \leq x
x - i = 0

\begin{align*}
i \geq 10 \\
i = 0; \\
i \leq 10 \\
i < 10 \\
&& x++; \\
i \geq 20 \\
i < 20 \\
i \geq 20 \\
i = 0; \\
i < 20 \\
i \geq 20 \\
i = 0; \\
&x++; \quad i++;
Widening

\[
0 \leq i, \quad 0 \leq x
\]
\[
x - i = 0
\]
Narrowing

Equations

\[ x_i = x_i \sqcap f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]

Narrowing operator

\[ b \sqsubseteq a \quad \implies \quad b \sqsubseteq a \sqcap b \sqsubseteq a \]
Narrowing

\[0 \leq i \leq 10, \ 0 \leq x \leq 10\]
\[x - i = 0\]

\[i = 0; \ x = 0;\]

\[i \geq 10\]

\[i = 0;\]

\[i < 10\]

\[x++;\]

\[i++;\]

\[i \geq 20\]

\[i < 20\]

\[x++;\]

\[i++;\]

\[20 \leq i\]

\[10 \leq x\]

\[0 \leq i\]

\[10 \leq x\]
Narrowing

1. \( i < 20 \)
   - \( i = 0; \)
   - \( x = 0 \)
2. \( i = 20 \)
   - \( 10 \leq x \)
   - \( i = 20; \)
   - \( 10 \leq x \)
3. \( i > 20 \)
   - \( 0 \leq i \leq 10, 0 \leq x \leq 10 \)
   - \( x - i = 0 \)
   - \( i = 0; x = 0 \)

\( i + 1 \)
Problem

- Precision, once abandoned, is difficult to recover.
- Narrowing is only partially defined and requires:
  ... equations
  ... monotonic right-hand sides
Problem

- Precision, once abandoned, is difficult to recover.
- Narrowing is only partially defined and requires:
  
  ... equations
  
  ... **monotonic** right-hand sides

  ➞ not met by interprocedural analysis ...
Our Analyzer Goblint

Certify absence of concurrency bugs in C!
Our Analyzer Goblint

Certify absence of concurrency bugs in C!

Automotive
Avionics

Challenges

Control-flow may depend on data.
Call-graph may depend on pointer analysis.
Multi-threading.
Decent scalability.
Architecture:

Program

Frontend

Analysis 1
Solver 1

Analysis 2
Solver 2

Analysis n
Solver n

Answer
Goblint:

Program → Frontend → Analysis 1 → Analysis 2 → Analysis n → Solver → Answer
# Observation 2

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Constraint system</th>
</tr>
</thead>
<tbody>
<tr>
<td>no context</td>
<td>finite monotonic</td>
</tr>
<tr>
<td></td>
<td>finite dependences</td>
</tr>
<tr>
<td>context</td>
<td>infinite non-monotonic</td>
</tr>
<tr>
<td></td>
<td>finite dependences</td>
</tr>
<tr>
<td>partial context</td>
<td>infinite non-monotonic</td>
</tr>
<tr>
<td></td>
<td>infinite dependences</td>
</tr>
</tbody>
</table>
Observation 2

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Constraint system</th>
</tr>
</thead>
<tbody>
<tr>
<td>no context</td>
<td>finite monotonic</td>
</tr>
<tr>
<td></td>
<td>finite dependences</td>
</tr>
<tr>
<td>context</td>
<td>infinite non-monotonic</td>
</tr>
<tr>
<td></td>
<td>finite dependences</td>
</tr>
<tr>
<td>partial context</td>
<td>infinite non-monotonic</td>
</tr>
<tr>
<td></td>
<td>infinite dependences</td>
</tr>
</tbody>
</table>

...  

2-phase algorithm is **unsound** !
Overview

- Combining widening and narrowing
- Termination
- Experiments
Combining Widening and Narrowing

Equations

\[ x_i = x_i \sqcap f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]
Combining Widening and Narrowing

Equations

\[ x_i = x_i \sqsubseteq f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]

Combined operator

\[ a \sqsubseteq b = \begin{cases} 
  b \sqsubseteq a & \text{then} \ a \sqsupseteq b \\
  \text{else} \ a \sqsupset b
\end{cases} \]
Idea

- Perform one joint iteration!
- Iterate until stabilization!
Idea

- Perform one joint iteration!
- Iterate until stabilization!

Theorem

Assume that the fixpoint algorithm performs a sequence of atomic evaluations of right-hand sides. Then

- Upon termination, a post-solution is attained ...
- This holds independent of the start values!
Combined

\[ i < 20 \]
\[ x = 0; i = 0; x++ \]
\[ i < 10 \]
\[ x = 0; \]
\[ i \geq 20 \]
\[ x = 0; i = 0; x++ \]
\[ i \geq 10 \]
\[ i = 0; x = 0; \]
\[ 0 \leq i \leq 10, 0 \leq x \leq 10 \]
\[ x - i = 0 \]
Combined

\[0 \leq i \leq 10, \ 0 \leq x \leq 10\]
\[x - i = 0\]
\[i = 0; \ x = 0;\]

\[i \geq 10\]

\[i = 0;\]

\[i < 10\]
\[x++;\]

\[i \geq 20\]

\[i = 0;\]

\[i < 20\]
\[x++;\]

\[i = 20\]
\[x = 30\]
\[i++;\]
Termination

- Any solver performing chaotic fixpoint iteration can be enhanced to a ⊓-solver.
- Termination, though, can no longer be guaranteed—even if right-hand sides are monotonic...
Round robin iteration

\[
\begin{align*}
x_1 &= x_2 \\
x_2 &= x_3 + 1 \\
x_3 &= x_1
\end{align*}
\]

for \( \mathbb{N} \cup \{\infty\} \) with \( \leq \)
Round robin iteration

\[ x_1 = x_2 \]
\[ x_2 = x_3 + 1 \]
\[ x_3 = x_1 \]

for \( \mathbb{N} \cup \{\infty\} \) with \( \leq \) results in:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>0</td>
<td>0</td>
<td>( \infty )</td>
<td>1</td>
<td>( \infty )</td>
<td>2</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>0</td>
<td>( \infty )</td>
<td>1</td>
<td>( \infty )</td>
<td>2</td>
<td>( \infty )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0</td>
<td>0</td>
<td>( \infty )</td>
<td>1</td>
<td>( \infty )</td>
<td>2</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>
Work list iteration

\[
\begin{align*}
x_1 &= x_1 + 1 \land x_2 + 1 \\
x_2 &= x_2 + 1 \land x_1 + 1
\end{align*}
\]
Work list iteration

\[ x_1 = x_1 + 1 \land x_2 + 1 \]
\[ x_2 = x_2 + 1 \land x_1 + 1 \]

results in:

<table>
<thead>
<tr>
<th>( W )</th>
<th>([x_1, x_2])</th>
<th>([x_1, x_2])</th>
<th>([x_1, x_2])</th>
<th>([x_2])</th>
<th>([x_2, x_1])</th>
<th>([x_2, x_1])</th>
<th>([x_1])</th>
<th>([x_1, x_2])</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>0</td>
<td>( \infty )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \infty )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \infty )</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

...
Bad News

Standard solvers fail to terminate already for trivial systems of equations.
Bad News

Standard solvers fail to terminate already for trivial systems of equations.

Good News

Variations of standard solvers are guaranteed to terminate—whenever right-hand sides are monotonic ...
void solve i  {
    if (i = 0) return;
    solve (i − 1);
    new ← ρ(x_i) ⊔ f_i ρ;
    if (ρ(x_i) ≠ new)  {
        ρ(x_i) ← new;
        solve i;
    }
}
Structured RR (cont.)

\[ x_1 = x_2 \]
\[ x_2 = x_3 + 1 \]
\[ x_3 = x_1 \]

results in:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>0</td>
<td>0</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \infty )</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>0</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
</tbody>
</table>
Theorem

(1) If SRR terminates, a post-solution has been found.

(2) SRR terminates, whenever all right-hand sides are monotonic.
Structured W

\[ Q \leftarrow \emptyset; \quad \text{for} \ (i \leftarrow 1; i \leq n; i++) \ \text{add} \ Q \ x_i; \]

\[ \text{while} \ (Q \neq \emptyset) \ {\}
\]

\[ x_i \leftarrow \text{extract}_\text{min} \ (Q); \]

\[ \text{new} \leftarrow \rho(x_i) \sqcup f_i \rho; \]

\[ \text{if} \ (\rho(x_i) \neq \text{new}) \ {\}
\]

\[ \rho(x_i) \leftarrow \text{new}; \quad \text{add} \ Q \ x_i; \]

\[ \text{forall} \ (x_j \in \text{infl}_i) \ \text{add} \ Q \ x_j; \]

\}
Structured W

\[ Q \leftarrow \emptyset; \quad \textbf{for} \ (i \leftarrow 1; \ i \leq n; \ i++) \ \textbf{add} \ Q \ x_i; \]

\[ \textbf{while} \ (Q \neq \emptyset) \ \{ \quad \text{// } Q \quad \text{priority queue} \]

\[ x_i \leftarrow \text{extract\_min} (Q); \]

\[ \text{new} \leftarrow \rho(x_i) \sqcup f_i \rho; \]

\[ \textbf{if} \ (\rho(x_i) \neq \text{new}) \ \{ \]

\[ \rho(x_i) \leftarrow \text{new}; \quad \text{add } Q \ x_i; \]

\[ \textbf{forall} \ (x_j \in \text{infl}_i) \ \text{add } Q \ x_j; \]

\[ \} \]

\}
Structured W (cont.)

\[ x_1 = x_1 + 1 \land x_2 + 1 \]
\[ x_2 = x_2 + 1 \land x_1 + 1 \]

results in:

\[
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
Q & [x_1, x_2] & [x_1, x_2] & [x_1, x_2] & [x_2] & [x_1, x_2] & [x_1, x_2] & [x_2] & \] \\
x_1 & 0 & \infty & 1 & 1 & 1 & \infty & \infty & \infty \\
x_2 & 0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty \\
\hline
\end{array}
\]
Theorem

(1) If $SW$ terminates, a post-solution has been found.

(2) SW terminates, whenever all right-hand sides are monotonic.
Extension

- Context-sensitive interprocedural analysis for complex invariants gives rise to \textit{infinite} equation systems with \textit{dynamic} variable dependences.

\[ \implies \text{local fixpoint iteration} \]
Extension

- Context-sensitive interprocedural analysis for complex invariants gives rise to infinite equation systems with dynamic variable dependences.
  \[ \Rightarrow \] local fixpoint iteration

- Partial contexts and the combination with flow-insensitive analysis can conveniently be handled using side-effecting ...
Extension (cont.)

- Two-phase widening/narrowing does neither work well with local solving nor with side-effecting.
- The local side-effecting solver, e.g., in Goblint does not perform chaotic fixpoint iteration.
Extension (cont.)

- Two-phase widening/narrowing does neither works well with *local* solving nor with *side-effecting*.
- The local side-effecting solver, e.g., in *Goblint* does not perform chaotic fixpoint iteration

\[\Rightarrow \text{new } \sqcap\text{-solver } \text{SLR}^+ \text{ (possibly with restart)}\]
Theorem

(1) If $\text{SLR}^+$ (w/o restart) terminates, a post-solution has been found.

(2) $\text{SLR}^+$ (w/o restart) terminates, whenever only finitely many unknowns are encountered and all right-hand sides are monotonic.
Experiments

**Precision**  2-phase vs. ⊓-solving

**Efficiency**  ⊓-solving against ⊔-solving
Experiments

**Precision**  2-phase vs. -solving

benchmark suite of the Märdalen WCET research group

**Efficiency**  -solving against -solving

C projects of the SpecCpu2006 benchmark suite that can be handled by CIL
Percentage of program points with improvement
### SpecCpu2006 without Contexts

<table>
<thead>
<tr>
<th>Program</th>
<th>Time(s)</th>
<th>Unknowns</th>
<th>Time(s)</th>
<th>Unknowns</th>
</tr>
</thead>
<tbody>
<tr>
<td>401.bzip2</td>
<td>3.3</td>
<td>6565</td>
<td>3.3</td>
<td>6578</td>
</tr>
<tr>
<td>429.mcf</td>
<td>0.4</td>
<td>1245</td>
<td>0.4</td>
<td>1245</td>
</tr>
<tr>
<td>433.milc</td>
<td>9.3</td>
<td>8206</td>
<td>9.3</td>
<td>8206</td>
</tr>
<tr>
<td>456.hmmer</td>
<td>11.2</td>
<td>11446</td>
<td>11.2</td>
<td>11446</td>
</tr>
<tr>
<td>458.sjeng</td>
<td>49.1</td>
<td>13306</td>
<td>52.1</td>
<td>13306</td>
</tr>
<tr>
<td>470.lbm</td>
<td>0.3</td>
<td>784</td>
<td>0.3</td>
<td>784</td>
</tr>
<tr>
<td>482.sphinx</td>
<td>14.0</td>
<td>11963</td>
<td>13.9</td>
<td>11963</td>
</tr>
</tbody>
</table>
### SpecCpu2006 with Contexts

<table>
<thead>
<tr>
<th>Program</th>
<th>Time(s)</th>
<th>Unknowns</th>
<th>Time(s)</th>
<th>Unknowns</th>
</tr>
</thead>
<tbody>
<tr>
<td>401.bzip2</td>
<td>3.3</td>
<td>7033</td>
<td>3.3</td>
<td>7028</td>
</tr>
<tr>
<td>429.mcf</td>
<td>0.5</td>
<td>1673</td>
<td>0.5</td>
<td>1673</td>
</tr>
<tr>
<td>433.milc</td>
<td>13.5</td>
<td>36720</td>
<td>13.4</td>
<td>36711</td>
</tr>
<tr>
<td>456.hmmer</td>
<td>12.6</td>
<td>80938</td>
<td>13.3</td>
<td>81038</td>
</tr>
<tr>
<td>458.sjeng</td>
<td>121.0</td>
<td>97785</td>
<td>131.0</td>
<td>108371</td>
</tr>
<tr>
<td>470.lbm</td>
<td>0.4</td>
<td>1577</td>
<td>0.2</td>
<td>899</td>
</tr>
<tr>
<td>482.sphinx</td>
<td>6.3</td>
<td>21610</td>
<td>6.3</td>
<td>21610</td>
</tr>
</tbody>
</table>
Results

- A significant amount of precision is retained by combining widening and narrowing!
- While sizes range from 1 to 33 kloc, the effective run-times essentially depend on the number of unknowns.
- Context-\textit{insensitive} analysis virtually is always faster than context-\textit{sensitive} analysis (exception: \texttt{482.sphinx}).
- In absence of context, the $\sqcap$-solver is only marginally slower than the corresponding $\sqcup$-solver.
Summary

- Combining widening and narrowing into a single operator allows to solve non-monotonic infinite systems of equations.
- Standard fixpoint algorithms, enhanced with the new operator, though, may not terminate.
- Termination can be enforced for monotonic systems when only finitely many variables are encountered—by modifying the fixpoint algorithms appropriately.
- The resulting solvers seem to improve precision significantly—at a decent extra cost.