EXPERIENCE WITH THE DESIGN OF A SPECIAL PURPOSE STATIC ANALYZER

P. Cousot
Patrick.Cousot@ens.fr http://www.di.ens.fr/~cousot
Biarritz IFIP-WG2.3 meeting (4)
23 — 28 mars 2003, Hotel Miramar, Biarritz, France
© P. Cousot, all rights reserved.

3. Application to Static Analysis
An Introduction to Abstract Interpretation, © P. Cousot, 25/3/03 — 3:1/58 — J I ¬ 或其他 » | m, Toc

WIDENING OPERATOR

A widening operator \( \nabla \in \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L} \) is such that:

- **Correctness:**
  - \( \forall x, y \in \mathcal{L} : \gamma(x) \subseteq \gamma(x \nabla y) \)
  - \( \forall x, y \in \mathcal{L} : \gamma(y) \subseteq \gamma(x \nabla y) \)

- **Convergence:**
  - for all increasing chains \( x^0 \subseteq x^1 \subseteq \ldots \), the increasing chain defined by \( y^0 = x^0, \ldots, y^{i+1} = y^i \nabla x^{i+1} \ldots \) is not strictly increasing.

3.5 FIXPOINT APPROXIMATION WITH CONVERGENCE ACCELERATION BY WIDENING/NARROWING


FIXPOINT APPROXIMATION WITH WIDENING

The upward iteration sequence with widening:

- \( \tilde{X}^0 = \sqcap \) (infimum)
- \( \tilde{X}^{i+1} = \tilde{X}^i \) if \( \mathcal{F}(\tilde{X}^i) \subseteq \tilde{X}^i \)
  \[ = \tilde{X}^i \nabla \mathcal{F}(\tilde{X}^i) \quad \text{otherwise} \]

is ultimately stationary and its limit \( \tilde{A} \) is a sound upper approximation of \( \text{lfp}^+ \mathcal{F} \):

\[ \text{lfp}^+ \mathcal{F} \subseteq \tilde{A} \]
**Interval Widening with threshold set**

- The threshold set $T$ is a finite set of numbers (plus $+\infty$ and $-\infty$).
- $[a, b] \triangledown_T [a', b'] = [\text{if } a' < a \text{ then } \max\{\ell \in T \mid \ell \leq a'\} \text{ else } a,$
  
  $\text{if } b' > b \text{ then } \min\{h \in T \mid h \geq b'\} \text{ else } b]$.  

- Examples (intervals):
  - sign analysis: $T = \{-\infty, 0, +\infty\}$;
  - strict sign analysis: $T = \{-\infty, -1, 0, +1, +\infty\}$;

- $T$ is a parameter of the analysis.

**Non-Existence of Finite Abstractions**

Let us consider the infinite family of programs parameterized by the mathematical constants $n_1, n_2$ ($n_1 \leq n_2$):

\[
x := n_1; \\
\text{while } x \leq n_2 \text{ do} \\
x := x + 1;
\]

- An interval analysis with widening/narrowing will discover the loop invariant $x \in [n_1, n_2]$;

- To handle all programs in the family without false alarm, the abstract domain must contain all such intervals;

$\Rightarrow$ No single finite abstract domain will do for all programs!
3.8 Application to the static analysis of critical real-time synchronous embedded software

General-Purpose Static Program Analyzers

- To handle infinitely many programs for non-trivial properties, a general-purpose analyser must use an infinite abstract domain \(^{20}\).
- Such analyzers are huge for complex languages hence very costly to develop but reusable;
- There are always programs for which they lead to false alarms;
- Although incomplete, they are very useful for verifying/testing/debugging.


3.8.1 General-Purpose versus Specializable Static Program Analysis

Parametric Specializable Static Program Analyzers

- The abstraction can provably be tailored to one program without any false alarm [SARA ’00];
- So, may be, the abstraction can be tailored to significant classes of programs (e.g. critical synchronous real-time embedded systems);
- This would lead to very efficient analyzers with zero (or almost no) false alarm even for large programs.

Reference

**The Class of Periodic Synchronous Programs**

```
declare volatile input, state and output variables;
initialize state variables;
loop forever
    - read volatile input variables,
    - compute output and state variables,
    - write to volatile output variables;
wait for next clock tick;
end loop
```

- All computations originates from non-linear control theory;
- The only allowed interrupts are clock ticks;
- Execution time of loop body less than a clock tick [4].

---

**A First Experience of Parametric Specializable Static Program Analyzers**

- C programs: safety critical embedded real-time synchronous software for non-linear control of complex systems;
- 10 000 LOCs, 1300 global variables (booleans, integers, floats, arrays, macros, non-recursive procedures);
- Implicit specification: absence of runtime errors (no integer/float point arithmetic overflow, no array bound overflow);
- Comparative results (commercial software):
  - 70 false alarms, 2 days, 500 Megabytes;

---

**3.8.2 First Experience**

- Initial design: 2h, 110 false alarms (general purpose interval-based analyzer);
- Main redesign:
  - Reduced product with weak relational domain with time;
- Parametrisation:
  - Hypotheses on volatile inputs;
  - Staged widenings with thresholds;
  - Local refinements of the parameterized abstract domains;
- Results: No false alarm, 14s, 20 Megabytes.

---

**References**


**Example of a Simple Idea That Does Not Scale Up**

- Represent abstract environments $\bar{M} = X \mapssto \bar{D}$ where $\bar{D}$ is the abstract domain as arrays/functional arrays;
- $O(1)$ to access/change the abstract value of an identifier but, most variables are locally unchanged so a lot of time is lost in unions $P \cup \bar{P} = \bar{P}$ and widenings $P \bigtriangledown \bar{P} = \bar{P}$;
- **Solution**: shared balanced binary tree (maps in CAML);
- $O(\ln n)$ among $n$ to access/change the abstract value of an identifier but, most of the tree is unchanged in unions and widenings (gained factor 7 in time).

**Example of refinement: trace partitionning**

Control point partitionning:

Trace partitionning:

<table>
<thead>
<tr>
<th>Fork</th>
<th>Join</th>
</tr>
</thead>
</table>

**Performance: Space and Time**

$$\text{Space} = O(\text{LOCs})$$

$$\text{Time} = O(\text{LOCs} \times (\ln(\text{LOCs}))^{1.5})$$

**3.8.3 Second Experience**

A Second Experience of Parametric Specializable Static Program Analyzers

- Same C programs for synchronous non-linear control of very complex systems;
- 132,000 lines of C, 75,000 LOCs after preprocessing, 10,000 global variables, over 21,000 after expansion of small arrays;
- Same implicit specification: absence of runtime errors + no modulo arithmetic;
- Analyzer of first experience: 30mn, 1,200 false alarms;

Some Difficulties (Among Others)

- Ignoring the value of any variable at any program point creates false alarms;
- Most precise abstract domains (e.g. polyhedra [7]) simply do not scale up;
- Tracing the fixpoint computation will produce huge log files crashing usual text editors;

Example of Difficulty: Semantics Problems

- For C programs, the abstract transfer functions have to take the machine-level semantics into account;
- For example:
  - floating-point arithmetic with rounding errors as opposed to real numbers (e.g. \( A + B < C \land D - B \leq C \neq A + D < 2 \times C \));
  - ESC is simply unsound with respect to modulo arithmetics [8].

Example of Refinement: Octagons

\[ \begin{align*}
1 \leq x & \leq 9 \\
1 \leq y & \leq 20 \\
x + y & \leq 78 \\
x - y & \leq 03
\end{align*} \]
Difficulty 1 with Octagons

- Most operations are $O(n^2)$ in space and $O(n^3)$ in time, so does not scale up;
- **Solution:**
  - Parameterize with packs of variables/program points where to use octagons,
  - Automatize the determination of the packs by experimentation (to eliminate the useless ones);

Difficulty 2 with Octagons

- Must be correct with respect to the IEEE 754 floating-point arithmetic norm;
- **Solution:** sophisticated algorithmics to correctly handle concrete and abstract rounding errors

An Introduction to Abstract Interpretation, P. Cousot, 24/3/03 — 3:104/121 — © P. Cousot, Toc
Mastering Invariant Size Explosion

The main loop invariant: a textual file over 4.5 Mb with

- 6,900 boolean interval assertions ($x \in [0; 1]$)
- 9,600 interval assertions ($x \in [a; b]$)
- 25,400 clock assertions ($x + \text{clk} \in [a; b] \land x - \text{clk} \in [a; b]$)
- 19,100 additive octagonal assertions ($a \leq x + y \leq b$)
- 19,200 subtractive octagonal assertions ($a \leq x - y \leq b$)
- 100 decision trees
- etc, ...

involving over 16,000 floating point constants (only 550 appearing in the program text) × 75,000 LOCs.