EXPERIENCE WITH THE DESIGN OF
A SPECIAL PURPOSE STATIC ANALYZER

P. Cousot

Patrick.Cousot@ens.fr http://www.di.ens.fr/~cousot

Biarritz IFIP-WG 2.3 meeting (4)

23 — 28 mars 2003, Hotel Miramar, Biarritz, France

Widening Operator

A widening operator $\nabla \in \overline{L} \times \overline{L} \longmapsto \overline{L}$ is such that:

- Correctness:
  - $\forall x, y \in \overline{L} : \gamma(x) \sqsubseteq \gamma(x \nabla y)$
  - $\forall x, y \in \overline{L} : \gamma(y) \sqsubseteq \gamma(x \nabla y)$
- Convergence:
  - for all increasing chains $x^0 \sqsubseteq x^1 \sqsubseteq \ldots$, 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

P. Cousot, R. Cousot: Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. PLILP, LNCS 631, 1992: 269-295, Springer.

Fixpoint Approximation with Widening

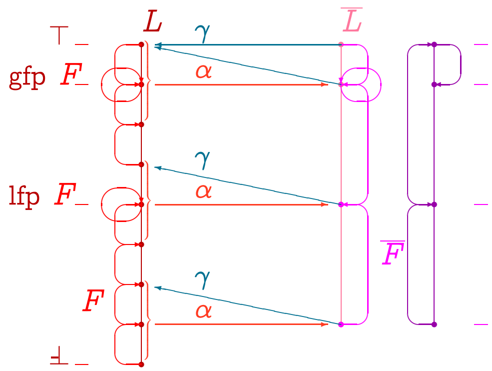
The upward iteration sequence with widening:

- $\tilde{X}^0 = \overline{\bot}$ (infimum)
- $\tilde{X}^{i+1} = \tilde{X}^i$ if $\overline{F}(\tilde{X}^i) \sqsubseteq \tilde{X}^i$
  $= \tilde{X}^i \nabla F(\tilde{X}^i)$ otherwise

is ultimately stationary and its limit $\tilde{A}$ is a sound upper approximation of $\mathrm{lfp}^{\overline{\bot}}\ \overline{F}$:

$$\mathrm{lfp}^{\overline{\bot}}\ \overline{F} \sqsubseteq \tilde{A}$$

Fixpoint Approximation with Widening/Narrowing



$$\begin{aligned}\tilde{X}^2 &= \tilde{X}^1 \nabla \overline{F}(\tilde{X}^1)\\ &= \overline{\top} = \check{X}^0\\ \check{X}^1 &= \check{X}^0 \underline{\Delta} \overline{F}(\check{X}^0)\\ &= \text{gfp}\overline{F} = \text{lfp}\overline{F}\end{aligned}$$

$$\tilde{X}^1 = \tilde{X}^0 \nabla \overline{F}(\tilde{X}^0)$$

$$\tilde{X}^0 = \overline{\bot}$$

Interval Widening with threshold set

- The threshold set T is a finite set of numbers (plus $+\infty$ and $-\infty$),
- $[a,b] \nabla_T [a',b'] = [\mathit{if}\ a' < a\ \mathit{then}\ \max\{\ell \in T \mid \ell \leq a'\}$
  $\mathit{else}\ a,$
  $\mathit{if}\ b' > b\ \mathit{then}\ \min\{h \in T \mid h \geq b'\}$
  $\mathit{else}\ b]$.
- Examples (intervals):
  - sign analysis: $T = \{-\infty, 0, +\infty\}$;
  - strict sign analysis: $T = \{-\infty, -1, 0, +1, +\infty\}$;
- T is a parameter of the analysis.

Interval Widening

- $\overline{L} = \{\bot\} \cup \{[\ell,\ u] \mid \ell \in \mathbb{Z} \cup \{-\infty\} \wedge u \in \mathbb{Z} \cup \{+\infty\} \wedge \ell \leq u\}$
- The widening extrapolates unstable bounds to infinity:

$$\begin{aligned}\bot \nabla X &= X\\ X \nabla \bot &= X\\ [\ell_0,\ u_0] \nabla [\ell_1,\ u_1] &= [\text{if } \ell_1 < \ell_0 \text{ then } -\infty \text{ else } \ell_0,\\ &\quad\ \text{if } u_1 > u_0 \text{ then } +\infty \text{ else } u_0]\end{aligned}$$

Not monotone. For example $[0,\ 1] \sqsubseteq [0,\ 2]$ but $[0,\ 1] \nabla [0,\ 2] = [0,\ +\infty] \not\sqsubseteq [0,\ 2] = [0,\ 2] \nabla [0,\ 2]$

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 := n1;
while X ≤ n2 do
  X := X + 1;
od
```

- 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;

  ⇒ No single finite abstract domain will do for all programs!

3.8 Application to the static analysis of critical real-time synchronous embedded software

3.8.1 General-Purpose versus Specializable Static Program Analysis

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.

[20] P. Cousot & R. Cousot. *Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation.* PLILP'92. LNCS 631, pp. 269–295. Springer.

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

[SARA '00] P. Cousot. Partial Completeness of Abstract Fixpoint Checking, invited paper. In 4^{th} *Int. Symp. SARA '2000*, LNAI 1864, Springer, pp. 1–25, 2000.

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].

Reference

[4] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm. Reliable and precise WCET determination for a real-life processor. *ESOP (2001)*, LNCS 2211, 469–485. 92

3.8.2 First Experience

Reference

[5] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Design and implementation of a special-purpose static program analyzer for safety-critical real-time embedded software. *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*, LNCS 2566, pages 85–108. Springer, 2002.

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/floating point arithmetic overflow, no array bound overflow);
- Comparative results (commercial software):
  - 70 false alarms, 2 days, 500 Megabytes;

First Experience Report

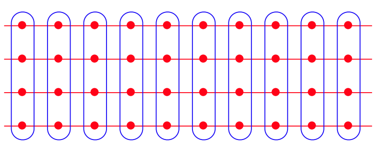
- Initial design: 2h, 110 false alarms (general purpose interval-based analyzer);
- Main redesign:
  - Reduced product with weak relational domain with time;
- Parametrisation:
  - Hypotheses on volatile inputs;
  - Staged widenings with thresholds;
  - Local refinements of the parameterized abstract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

Example of a Simple Idea That Does Not Scale Up

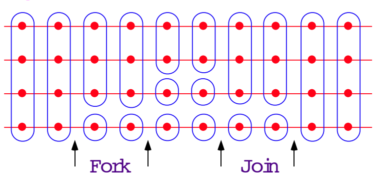
- Represent abstract environments $\bar{\mathcal{M}} = \mathbb{X} \longmapsto \bar{\mathcal{D}}$ where $\bar{\mathcal{D}}$ is the abstract domain as arrays/functional arrays;
- $\mathcal{O}(1)$ to access/change the abstract value of an identifier but, most variables are locally unchanged so a lot of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

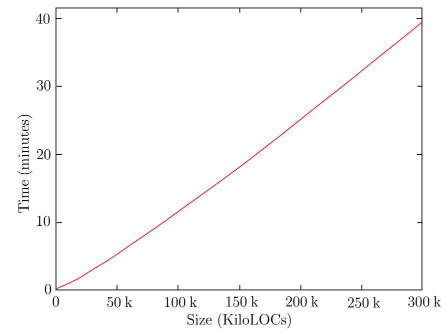


Trace partitionning:



Performance: Space and Time

$$\begin{aligned} \text{Space} &= \mathcal{O}(\text{LOCs}) \\ \text{Time} &= \mathcal{O}(\text{LOCs} \times (\ln(\text{LOCs}))^{1.5}) \end{aligned}$$



3.8.3 Second Experience

Reference

[6] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety critical software. *ACM PLDI'03*, San Diego, CA, June 2003, to appear.

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;

Reference

[7] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In 5^{th} *POPL*, pages 84–97, Tucson, AZ, 1978. ACM Press. 101

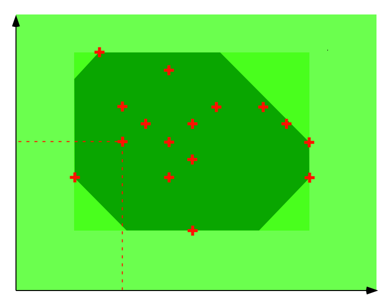
Example of Difficulty: Semantics Problems

- 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. $A+B < C \wedge D-B \leq C \not\Rightarrow A+D < 2 \times C$);
  - ESC is simply unsound with respect to modulo arithmetics [8].

Reference

[8] Flanagan, C., Leino, K.R.M., Lillibridge, M., Nelson, G., Saxe, J., Stata, R.: *Extended static checking for Java.* PLDI'02, ACM SIGPLAN Not. 37(5), (2002) 234–245. 102

Example of Refinement: Octagons



$$\begin{cases} 1 \leq x \leq 9 \\ x + y \leq 78 \\ 1 \leq y \leq 20 \\ x - y \leq 03 \end{cases}$$

Reference

[9] A. Miné. A New Numerical Abstract Domain Based on Difference-Bound Matrices. In *PADO'2001*, LNCS 2053, Springer, 2001, pp. 155–172.

Difficulty 1 with Octagons

- Most operations are $\mathcal{O}(n^2)$ in space and $\mathcal{O}(n^3)$ 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 [21]

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

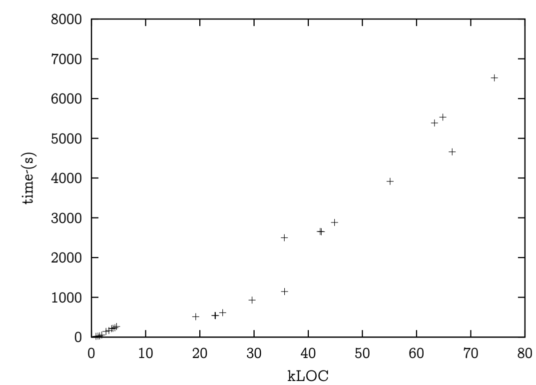
[21] An opened problem with polyhedra.

Second Experience (Preliminary) Report

- Comparative results (commercial software):

  2,000 (false?) alarms, 3 days;
- Results: 2 (false?) alarms, 1h30mn, 2 Gigabytes.

Benchmarks



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 ($x + \mathrm{clk} \in [a; b] \wedge x - \mathrm{clk} \in [a; b]$)
- 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) $\times$ 75,000 LOCs.