Implementing and Evaluating a Strategy-Iteration Based Static Analyser within the LLVM framework

Carlo Zancanaro

November 12, 2012

Bugs are bad

- Money - recently Knight Capital, US$440 million lost in a day

Bugs are bad

- Money - recently Knight Capital, US$440 million lost in a day
- Time - 50% of development time is spent debugging[5]

Bugs are bad

- Money - recently Knight Capital, US$440 million lost in a day
- Time - 50% of development time is spent debugging[5]
- Security - buffer overflows and other security flaws

Bugs are bad

- Money - recently Knight Capital, US$440 million lost in a day
- Time - 50% of development time is spent debugging[5]
- Security - buffer overflows and other security flaws
- Lives - X-ray machines, helicopters, cars

Static analysis is good

- The more bugs we can catch at compile time, the better

Static analysis is good

- The more bugs we can catch at compile time, the better
- We can't catch all bugs - Rice's theorem[4]

Modelling programs

$x = 0$
$y = 1$
while $x < 8$ $\quad\triangleright$ value at start of this line
  $x = x + 2$
  $y = y + 2$
endwhile

Modelling programs

$x = 0$
$y = 1$
while $x < 8$ $\triangleright$ value at start of this line
    $x = x + 2$
    $y = y + 2$
endwhile

$$x = \{0\}$$
$$y = \{1\}$$

Modelling programs

$x = 0$
$y = 1$
while $x < 8$ ▷ value at start of this line
    $x = x + 2$
    $y = y + 2$
endwhile

$$x = \{0, 2\}$$
$$y = \{1, 3\}$$

Modelling programs

$x = 0$
$y = 1$
while $x < 8$ $\triangleright$ value at start of this line
  $x = x + 2$
  $y = y + 2$
endwhile

$$x = \{0, 2, 4\}$$
$$y = \{1, 3, 5\}$$

Modelling programs

$x = 0$
$y = 1$
while $x < 8$ $\triangleright$ value at start of this line
    $x = x + 2$
    $y = y + 2$
endwhile

$$x = \{0, 2, 4, 6\}$$
$$y = \{1, 3, 5, 7\}$$

Modelling programs

$x = 0$
$y = 1$
while $x < 8$ ▷ value at start of this line
    $x = x + 2$
    $y = y + 2$
endwhile

$$x = \{0, 2, 4, 6, 8\}$$
$$y = \{1, 3, 5, 7, 9\}$$

Abstract interpretation

Basic idea: simplify your domain
Instead of arbitrary subsets of $\mathbb{Z}$, something less precise:

- signs[1]: $x \in \{\mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}^-, 0\}$
- ranges[1]: $x \leq a; -x \leq b, \ a, b \in \mathbb{Z}$
- zones[3]: $x - y \leq c; \pm x \leq c \ c \in Z$

Abstract interpretation

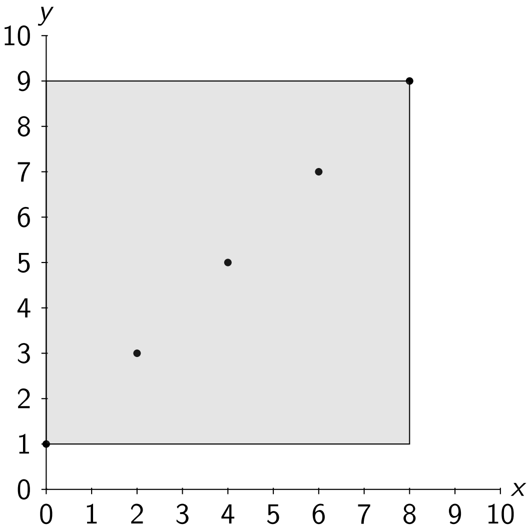

Figure: Comparison between concrete and abstract domains

max-strategy improvement

- Transform a program into equations
- Solve equations

max-strategy example

$x = 0$	$\triangleright$ A
while $x \leq 8$	$\triangleright$ B
$\quad x = x + 2$	
endwhile	
print(x)	$\triangleright$ C

$$
\begin{aligned}
ub(x)_A &\geq \infty \\
ub(x)_B &\geq 0 \\
ub(x)_B &\geq \min(ub(x)_B, 8) + 2 \\
ub(x)_C &\geq ub(x)_B
\end{aligned}
$$

max-strategy example

$$
\begin{array}{ll}
x = 0 & \triangleright\ A \\
\textbf{while } x \leq 8 & \triangleright\ B \\
\quad x = x + 2 & \\
\textbf{endwhile} & \\
\text{print}(x) & \triangleright\ C
\end{array}
$$

$$
\begin{aligned}
ub(x)_A &= \infty \\
ub(x)_B &= \max(0, \min(ub(x)_B, 8) + 2) \\
ub(x)_C &= ub(x)_B
\end{aligned}
$$

max-strategies

A max-strategy is a decision about which argument in a max-expression to choose.

$$
\begin{aligned}
ub(x)_A &= \infty \\
ub(x)_B &= \max(0, \min(ub(x)_B, 8) + 2) \\
ub(x)_C &= ub(x)_B
\end{aligned}
$$

max-strategies

A max-strategy is a decision about which argument in a max-expression to choose.

$$
\begin{aligned}
ub(x)_A &= \infty \\
ub(x)_B &= \max(0, \min(ub(x)_B, 8) + 2) \\
ub(x)_C &= ub(x)_B
\end{aligned}
$$

max-strategies

A max-strategy is a decision about which argument in a max-expression to choose.

$$
\begin{aligned}
ub(x)_A &= \infty \\
ub(x)_B &= \max(0, \min(ub(x)_B, 8) + 2) \\
ub(x)_C &= ub(x)_B
\end{aligned}
$$

Solver

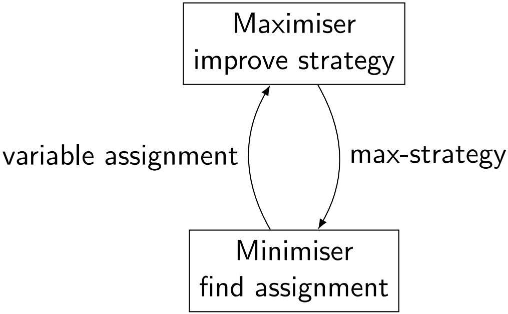

Figure: The high-level structure of the solver presented in [2]

Solver - enhancements

The big idea: take into account data-dependencies

$$
\begin{aligned}
x &= \max(0, \min(x - 1, y)) \\
y &= \max(0, x + 5, x) \\
z &= \max(0, z + 1, x)
\end{aligned}
$$

Solver - enhancements

The big idea: take into account data-dependencies

$$\begin{aligned} x &= \max(0, \min(x - 1, y)) \\ y &= \max(0, x + 5, x) \\ z &= \max(0, z + 1, x) \end{aligned}$$

Solver - enhancements

The big idea: take into account data-dependencies

$$\begin{aligned} x &= \max(0, \min(x-1, y)) \\ y &= \max(0, x+5, x) \\ z &= \max(0, z+1, x) \end{aligned}$$

Solver - enhancements

The big idea: take into account data-dependencies

$$\begin{aligned} x &= \max(0, \min(x - 1, y)) \\ y &= \max(0, x + 5, x) \\ z &= \max(0, z + 1, x) \end{aligned}$$

Solver - enhancements

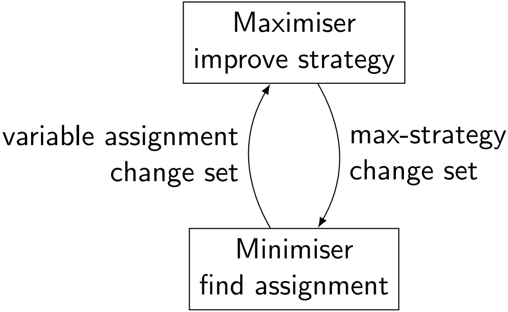

Figure: The high-level structure of our enhanced solver

Implementation

- Implemented in C++
- Integrated into the LLVM/Clang static analysis framework

Example system

$$
\begin{aligned}
x_0 &= \max(-\infty, 0) \\
x_1 &= \max(-\infty, x_0) \\
x_2 &= \max(-\infty, x_1) \\
\ldots & \ldots \\
x_n &= \max(-\infty, x_{n-1})
\end{aligned}
$$

Runtime improvements

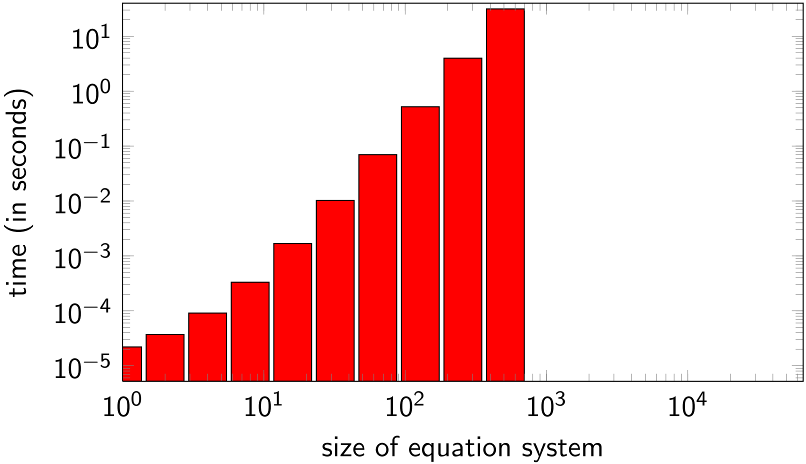

Figure: Performance of the naive algorithm

Runtime improvements

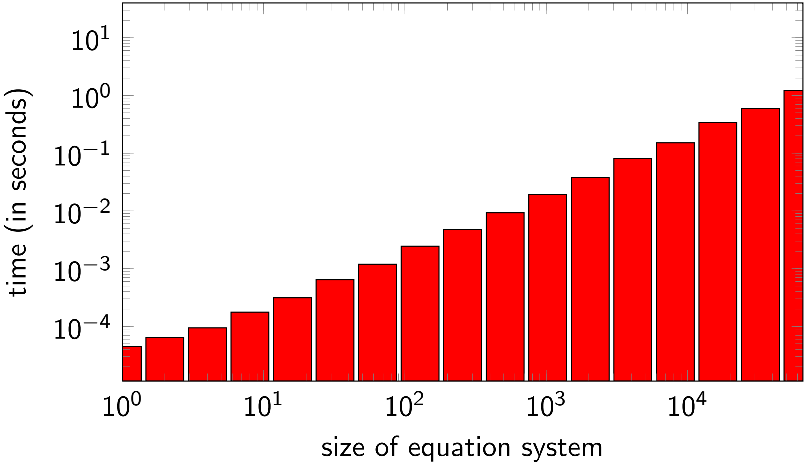

Figure: Performance of our improved algorithm

Future work

- Still slightly over-approximating dependencies
- LLVM/Clang integration is only a proof-of-concept

References I

[1] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.

[2] T. Gawlitza and H. Seidl. Precise fixpoint computation through strategy iteration. In *Proceedings of the 16th European conference on Programming*, ESOP'07, pages 300–315, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-71314-2. URL `http://dl.acm.org/citation.cfm?id=1762174.1762203`.

References II

[3] A. Miné. A new numerical abstract domain based on difference-bound matrices. In *Proc. of the 2d Symp. on Programs as Data Objects (PADO II)*, volume 2053 of *Lecture Notes in Computer Science*, pages 155–172. Springer, May 2001. `http://www.di.ens.fr/~mine/publi/article-mine-padoII.pdf`.

[4] H. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 83, 1953.

[5] A. Zeller. *Why Programs Fail: A Guide to Systematic Debugging*.

Contributions

- Improvement of max-strategy iteration algorithm, leveraging sparsity of variable dependencies
- Implementation of a max-strategy iteration based static analyser in the LLVM/Clang framework