Validating Optimizations of Concurrent C/C++ Programs

Soham Chakraborty   Viktor Vafeiadis

MPI-SWS

CGO 2016

$$\texttt{int X} = \texttt{0;} \quad \texttt{int Y} = \texttt{0;}$$

$$\begin{array}{l|l} \texttt{Y} = \texttt{4;} & \texttt{if (X)} \\ \texttt{X} = \texttt{1;} & \quad \texttt{r} = \texttt{Y;} \end{array}$$

$$\texttt{int X} = 0; \quad \texttt{int Y} = 0;$$

$$\begin{array}{l|l} \texttt{Y} = 4; & \texttt{if}\,(\texttt{X}) \\ \texttt{X} = 1; & \quad \texttt{r} = \texttt{Y}; \end{array}$$

Race on X $\rightsquigarrow$ undefined semantics

$X == 1 \land r \neq 4$ is possible

(i.e., the program is wrong)

$$\texttt{atomic\_int}\ X = 0; \quad \texttt{int}\ Y = 0;$$

$$\begin{array}{l|l} Y = 4; & \texttt{if}\,(\texttt{atomic\_load}(\&X, \\ & \qquad \texttt{mo\_acquire})) \\ \texttt{atomic\_store}(\&X, 1, & \\ \quad \texttt{mo\_release}); & \quad r = Y; \end{array}$$

$$\texttt{atomic\_int}\ X = 0; \quad \texttt{int}\ Y = 0;$$

$$\begin{array}{l|l}
Y = 4; & \texttt{if}\,(\texttt{atomic_load}(\&X, \\
 & \qquad \texttt{mo_acquire})) \\
\texttt{atomic_store}(\&X, 1, & \\
\quad \texttt{mo_release}); & \quad r = Y;
\end{array}$$

$$\texttt{atomic\_int}\ X = 0; \quad \texttt{int}\ Y = 0;$$

$$
\begin{array}{l|l}
Y = 4; & \texttt{if}\ (\texttt{atomic\_load}(\&X, \\
\texttt{atomic\_store}(\&X, 1, & \qquad \texttt{mo\_acquire})) \\
\qquad \texttt{mo\_release}); & \qquad r = Y;
\end{array}
$$

```
atomic_int X = 0;   int Y = 0;

Y = 4;                          || if (atomic_load(&X,
                                ||         mo_acquire))
atomic_store(&X, 1,             ||
    mo_release);                ||     r = Y;
```

$$\texttt{atomic\_int } X = 0; \quad \texttt{int } Y = 0;$$

$Y = 4;$	$\texttt{if}\,(\texttt{atomic\_load}(\&X,$
$\texttt{atomic\_store}(\&X, 1,$	$\texttt{mo\_acquire}))$
$\texttt{mo\_release});$	$r = Y;$

```
atomic_int X = 0;   int Y = 0;
```

`Y = 4;`	`if (atomic_load(&X,`
`atomic_store(&X, 1,`	`        mo_acquire))`
`    mo_release);`	`    r = Y;`

```
atomic_int X = 0;   int Y = 0;
```

`Y = 4;`
`atomic_store(&X, 1,`
`    mo_release);`
	`if (atomic_load(&X,`
	`        mo_acquire))`
	`    r = Y;`

Concurrent Programming in C11

```
atomic_int X = 0;   int Y = 0;

Y = 4;                           ||  if (atomic_load(&X,
atomic_store(&X, 1,              ||          mo_acquire))
        mo_release);             ||        r = Y;
```

$$\Downarrow$$

$$X = Y = 0;$$

$$\begin{array}{l||l} Y \quad = 4; & \text{if}(X_{\text{acq}}) \\ X_{\text{rel}} = 1; & \quad r = Y; \end{array}$$

An Unsafe Reordering

$$X = Y = 0;$$

$$\begin{array}{l||l} & r = 4; \\ Y = 4; & \text{if}(X_{\text{acq}}) \\ X_{\text{rel}} = 1; & \quad r = Y; \end{array} \quad \rightsquigarrow \quad \begin{array}{l||l} & r = 4; \\ X_{\text{rel}} = 1; & \text{if}(X_{\text{acq}}) \\ Y = 4; & \quad r = Y; \end{array}$$

$$X = Y = 0;$$

Always returns $r == 4$

May return $r == 0$

An Unsafe Reordering

$$X = Y = 0;$$
$$\begin{array}{l} Y = 4; \\ X_{\text{rel}} = 1; \end{array} \;\Bigg\|\; \begin{array}{l} r = 4; \\ \text{if}(X_{\text{acq}}) \\ \quad r = Y; \end{array}$$

$\leadsto$

$$X = Y = 0;$$
$$\begin{array}{l} X_{\text{rel}} = 1; \\ Y = 4; \end{array} \;\Bigg\|\; \begin{array}{l} r = 4; \\ \text{if}(X_{\text{acq}}) \\ \quad r = Y; \end{array}$$

Always returns $r == 4$

May return $r == 0$

Optimizations for sequential programs are NOT always safe for concurrent programs.

$$X = Y = 0;$$

$$\begin{array}{l|l} & f = \mathit{false}; \\ Y = 4; & \cdots \\ X_{\mathsf{rel}} = 1; & a = f\ ?\ Y : 0; \\ & b = X_{\mathsf{acq}}\ ?\ Y : 4; \end{array}$$

Another Example

$$X = Y = 0;$$

$$\begin{array}{l|l} & f = \mathit{false}; \\ Y = 4; & \ldots \\ X_{\text{rel}} = 1; & a = f \,?\, Y : 0; \\ & b = X_{\text{acq}} \,?\, Y : 4; \end{array}$$

Output: $b == 4$ always

$$
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
a = f \; ? \; Y : 0; \\
b = X_{\mathrm{acq}} \; ? \; Y : 4;
\end{array}
\quad \overset{-O3}{\leadsto} \quad
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
s = Y; \\
a = f \; ? \; s : 0; \\
t = X_{\mathrm{acq}}; \\
b = t \; ? \; s : 4;
\end{array}
$$

Context:

$$
\left[\; \Big\| \; \begin{array}{l} Y = 4; \\ X_{\mathrm{rel}} = 1; \end{array} \right]
$$

Output $b == 0$ possible in target.

LLVM Compilation Bug in More Detail

$$X = Y = 0;$$
$$f = \mathit{false};$$
$$\ldots$$
$$a = f\ ?\ Y : 0;$$
$$b = X_{\text{acq}}\ ?\ Y : 4;$$

$\overset{(1)}{\rightsquigarrow}$

$$X = Y = 0;$$
$$f = \mathit{false};$$
$$\ldots$$
$$s = \boxed{Y;}$$
$$a = f\ ?\ s : 0;$$
$$t = X_{\text{acq}};$$
$$r = \boxed{Y;}$$
$$b = t\ ?\ r : 4;$$

$\overset{(2)}{\rightsquigarrow}$

$$X = Y = 0;$$
$$f = \mathit{false};$$
$$\ldots$$
$$s = Y;$$
$$a = f\ ?\ s : 0;$$
$$t = X_{\text{acq}};$$
$$\cancel{r = Y};$$
$$b = t\ ?\ s : 4;$$

LLVM Compilation Bug in More Detail

$$
\begin{aligned}
&X = Y = 0;\\
&f = \mathit{false};\\
&\ldots\\
&a = f\ ?\ Y : 0;\\
&b = X_{\text{acq}}\ ?\ Y : 4;
\end{aligned}
\quad \overset{(1)}{\leadsto} \quad
\begin{aligned}
&X = Y = 0;\\
&f = \mathit{false};\\
&\ldots\\
&s = Y;\\
&a = f\ ?\ s : 0;\\
&t = X_{\text{acq}};\\
&r = Y;\\
&b = t\ ?\ \boxed{r} : 4;
\end{aligned}
\quad \overset{(2)}{\leadsto} \quad
\begin{aligned}
&X = Y = 0;\\
&f = \mathit{false};\\
&\ldots\\
&s = Y;\\
&a = f\ ?\ s : 0;\\
&t = X_{\text{acq}};\\
&\cancel{r = Y;}\\
&b = t\ ?\ \boxed{s} : 4;
\end{aligned}
$$

LLVM Compilation Bug in More Detail

$$
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\dots \\
a = f \;?\; {\color{blue}Y} : 0; \\
b = {\color{blue}X_{\mathsf{acq}}} \;?\; {\color{blue}Y} : 4;
\end{array}
\overset{(1)}{\leadsto}
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\dots \\
s = {\color{blue}Y;} \\
a = f \;?\; s : 0; \\
t = {\color{blue}X_{\mathsf{acq}};} \\
r = {\color{blue}Y;} \\
b = t \;?\; r : 4;
\end{array}
\overset{(2)}{\leadsto}
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\dots \\
s = {\color{blue}Y;} \\
a = f \;?\; s : 0; \\
t = {\color{blue}X_{\mathsf{acq}};} \\
\cancel{r = Y;} \\
b = t \;?\; s : 4;
\end{array}
$$

C11: (1) **Error**

LLVM Compilation Bug in More Detail

$$
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
a = f \; ? \; Y : 0; \\
b = X_{\mathrm{acq}} \; ? \; Y : 4;
\end{array}
\quad \overset{(1)}{\leadsto} \quad
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
s = Y; \\
a = f \; ? \; s : 0; \\
t = X_{\mathrm{acq}}; \\
r = Y; \\
b = t \; ? \; r : 4;
\end{array}
\quad \overset{(2)}{\leadsto} \quad
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
s = Y; \\
a = f \; ? \; s : 0; \\
t = X_{\mathrm{acq}}; \\
\cancel{r = Y;} \\
b = t \; ? \; s : 4;
\end{array}
$$

C11: (1) **Error** (2) **Correct**

LLVM Compilation Bug in More Detail

$$
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
a = f \mathbin{?} Y : 0; \\
b = X_{\mathsf{acq}} \mathbin{?} Y : 4;
\end{array}
\quad \overset{(1)}{\rightsquigarrow} \quad
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
s = Y; \\
a = f \mathbin{?} s : 0; \\
t = X_{\mathsf{acq}}; \\
r = Y; \\
b = t \mathbin{?} r : 4;
\end{array}
\quad \overset{(2)}{\rightsquigarrow} \quad
\begin{array}{l}
X = Y = 0; \\
f = \mathit{false}; \\
\ldots \\
s = Y; \\
a = f \mathbin{?} s : 0; \\
t = X_{\mathsf{acq}}; \\
\cancel{r = Y;} \\
b = t \mathbin{?} s : 4;
\end{array}
$$

C11: (1) **Error** (2) **Correct**

LLVM: (1) **Correct**

LLVM Compilation Bug in More Detail

$$X = Y = 0;\quad f = \mathit{false};\quad \ldots\quad a = f\ ?\ Y : 0;\quad b = X_{\mathsf{acq}}\ ?\ Y : 4;$$

$$\overset{(1)}{\rightsquigarrow}$$

$$X = Y = 0;\quad f = \mathit{false};\quad \ldots\quad s = Y;\quad a = f\ ?\ s : 0;\quad t = X_{\mathsf{acq}};\quad r = Y;\quad b = t\ ?\ r : 4;$$

$$\overset{(2)}{\rightsquigarrow}$$

$$X = Y = 0;\quad f = \mathit{false};\quad \ldots\quad s = Y;\quad a = f\ ?\ s : 0;\quad t = X_{\mathsf{acq}};\quad \cancel{r = Y;}\quad b = t\ ?\ s : 4;$$

C11:	(1) **Error**	(2) **Correct**
LLVM:	(1) **Correct**	(2) **Error**

This Work: LLVM Validation

$$P_{src} \xrightarrow{\text{LLVM}} P_{tgt} \; ? \; \textbf{Correct} : \textbf{Potential Error}$$

$$\Downarrow$$

$$P_{src} \xrightarrow{(R \cup E)^*} P_{tgt} \; ? \; \textbf{Correct} : \textbf{Potential Error}$$

Define a set of safe reorderings & eliminations:

- For the LLVM model
- For the C11 model [POPL'15]

Can be used in validating other compilers.

Steps:

- Identify corresponding program paths
- Compute deletability of accesses
- Match access sequences and analyze

$$s_1 = X$$
$$s_2 = X$$
$$V = 1$$
$$s_4 = Z_{\mathrm{acq}}$$
$$Y = 1$$
$$Y = 2$$

✓ $s_1 = X$

$s_2 = X$

$V = 1$

$s_4 = Z_{\mathrm{acq}}$

$Y = 1$

$Y = 2$

✓ $s_1 = X$

✗ $s_2 = X$

$V = 1$

$s_4 = Z_{\mathrm{acq}}$

$Y = 1$

$Y = 2$

✓ $s_1 = X$

✗ $s_2 = X$

$V = 1$

✓ $s_4 = Z_{\mathrm{acq}}$

$Y = 1$

$Y = 2$

Compiler Independent Matching

✓ $s_1 = X$

✗ $s_2 = X$

$V = 1$

✓ $s_4 = Z_{\mathrm{acq}}$

$Y = 1$

✓ $Y = 2$

✓ $s_1 = X$

✗ $s_2 = X$

$V = 1$

✓ $s_4 = Z_{\mathrm{acq}}$

✗ $Y = 1$

✓ $Y = 2$

- ✓ $s_1 = X$
- ✗ $s_2 = X$
- ✓ $V = 1$
- ✓ $s_4 = Z_{\mathrm{acq}}$
- ✗ $Y = 1$
- ✓ $Y = 2$

Compiler Independent Matching

✓ $s_1 = X$
✗ $s_2 = X$
✓ $V = 1$
✓ $s_4 = Z_{\text{acq}}$
✗ $Y = 1$
✓ $Y = 2$

$t_1 = X$
$t_2 = Z_{\text{acq}}$
$Y = 2$
$V = 1$

✓ $s_1 = X$

✗ $s_2 = X$

✓ $V = 1$

✓ $s_4 = Z_{\mathrm{acq}}$

✗ $Y = 1$

✓ $Y = 2$

$t_1 = X$

$t_2 = Z_{\mathrm{acq}}$

$Y = 2$

$V = 1$

Compiler Independent Matching

✓ $s_1 = X$

✗ $s_2 = X$

✓ $V = 1$

✓ $s_4 = Z_{\mathrm{acq}}$

✗ $Y = 1$

✓ $Y = 2$

$t_1 = X$

$t_2 = Z_{\mathrm{acq}}$

$Y = 2$

$V = 1$

- ✓ $s_1 = X$
- ✗ $s_2 = X$
- ✓ $V = 1$
- ✓ $s_4 = Z_{\mathrm{acq}}$
- ✗ $Y = 1$
- ✓ $Y = 2$

- $t_1 = X$
- $t_2 = Z_{\mathrm{acq}}$
- $Y = 2$
- $V = 1$

✓ $s_1 = X$

✗ $s_2 = X$

✓ $V = 1$

✓ $s_4 = Z_{\text{acq}}$

✗ $Y = 1$

✓ $Y = 2$

$t_1 = X$

$t_2 = Z_{\text{acq}}$

$Y = 2$

$V = 1$

Compiler Independent Matching

✓ $s_1 = X$

✗ $s_2 = X$

✓ $V = 1$

✓ $s_4 = Z_{\text{acq}}$

✗ $Y = 1$

✓ $Y = 2$

$t_1 = X$

$t_2 = Z_{\text{acq}}$

$Y = 2$

$V = 1$

- Check that unmatched accesses are deletable
- Check that reorderings are allowed

✓ $s_1 = X$	
✗ $s_2 = X$	$t_1 = X$
✓ $V = 1$	$t_2 = Z_{acq}$
✓ $s_4 = Z_{acq}$	$Y = 2$
✗ $Y = 1$	$V = 1$
✓ $Y = 2$	

- Check that unmatched accesses are deletable
- Check that reorderings are allowed

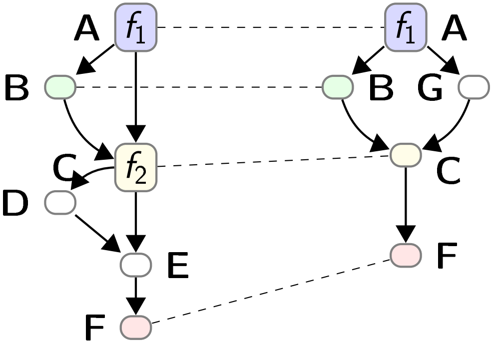
✓ $s_1 = X$
✗ $s_2 = X$
✓ $V = 1$
✓ $s_4 = Z_{\mathrm{acq}}$
✗ $Y = 1$
✓ $Y = 2$

$t_1 = X$
$t_2 = Z_{\mathrm{acq}}$
$Y = 2$
$V = 1$

Correct

- Check that unmatched accesses are deletable
- Check that reorderings are allowed

Control Flow Matching



- Use branching conditions to match the paths
- Unroll loops a fixed number of times

Experimental Evaluation

Validated according to LLVM memory model

Test class (100 prog./class)	# Reported errors	
	LLVM 3.6	LLVM 3.7rc2
Straightline	95	0
With branches	64	0
With dead paths	58	0
With loops	49	0
Smaller tests	32	0

- Examples frequently expose errors in LLVM 3.6
- No false positives!

Validated according to C11 memory model

Test class (100 prog./class)	# Reported errors LLVM 3.6	LLVM 3.7rc2
Straightline	0	0
With branches	13	1
With dead paths	6	0
With loops	6	0
Smaller tests	7	5

- Errors often masked by adjacent accesses

Masking of Errors by Adjacent Accesses

$s_2 = Z_{\mathrm{acq}}$ $\qquad$ $t_2 = X$

$s_3 = X$ $\qquad$ $t_3 = Z_{\mathrm{acq}}$

Masking of Errors by Adjacent Accesses

$$
\begin{array}{lcl}
s_1 = X & \qquad\qquad & t_1 = X \\
s_2 = Z_{\mathrm{acq}} & & t_2 = X \\
s_3 = X & & t_3 = Z_{\mathrm{acq}} \\
s_1 = X & & t_4 = X
\end{array}
$$

$s_1 = X$	$t_1 = X$
$s_2 = Z_{\text{acq}}$	$t_2 = X$
$s_3 = X$	$t_3 = Z_{\text{acq}}$
$s_1 = X$	$t_4 = X$

$s_1 = X$!A	$t_1 = X$!A
$s_2 = Z_{\mathrm{acq}}$!B	$t_2 = X$!C
$s_3 = X$!C	$t_3 = Z_{\mathrm{acq}}$!B
$s_4 = X$!D	$t_4 = X$!D

Summary

- C11 and LLVM semantics are different
- Reported three LLVM concurrency compilation bugs; all were fixed.
- Validator: `http://plv.mpi-sws.org/validc/`

Future Work

- Handle arrays, pointers, sequential optimisations
- Integrate with sequential validator
- Formalize the LLVM concurrency model