

LifeJacket: Verifying Precise Floating-Point Optimizations in LLVM

Andres Nötzli, Fraser Brown

Stanford University

Motivating Example

Suppose we want to optimize:

```
float y = +0.0 - (-x);
```

Motivating Example

Suppose we want to optimize:

```
float y = +0.0 - (-x);
```

How about:

```
float y = x;
```

Motivating Example

Suppose we want to optimize:

```
float y = +0.0 - (-x);
```

How about:

```
float y = x;
```

Nope: if $x = -0.0$ then $y = +0.0$

Motivating Example

So, we know that: $+0.0 - (-x) \neq x$.

Who cares? $+0.0 == -0.0$ is `true`.

Motivating Example

So, we know that: $+0.0 - (-x) \neq x$.

Who cares? $+0.0 == -0.0$ is **true**.

Well, $\frac{1}{0} = \infty$, $\frac{1}{-0} = -\infty$ and $\infty \neq -\infty$. Ouch.

Motivating Example

So, we know that: $+0.0 - (-x) \neq x$.

Who cares? $+0.0 == -0.0$ is **true**.

Well, $\frac{1}{0} = \infty$, $\frac{1}{-0} = -\infty$ and $\infty \neq -\infty$. Ouch.

What about:

$$\infty + -\infty = \text{NaN}$$

Motivating Example

So, we know that: $+0.0 - (-x) \neq x$.

Who cares? $+0.0 == -0.0$ is **true**.

Well, $\frac{1}{0} = \infty$, $\frac{1}{-0} = -\infty$ and $\infty \neq -\infty$. Ouch.

What about:

$$\infty + -\infty = \text{NaN}$$

$$x + \text{NaN} = \text{NaN}$$

Motivating Example

So, we know that: $+0.0 - (-x) \neq x$.

Who cares? $+0.0 == -0.0$ is **true**.

Well, $\frac{1}{0} = \infty$, $\frac{1}{-0} = -\infty$ and $\infty \neq -\infty$. Ouch.

What about:

$$\infty + -\infty = \text{NaN}$$

$$x + \text{NaN} = \text{NaN}$$

$$a + (b + c) \neq (a + b) + c$$

Motivating Example

So, we know that: $+0.0 - (-x) \neq x$.

Who cares? $+0.0 == -0.0$ is **true**.

Well, $\frac{1}{0} = \infty$, $\frac{1}{-0} = -\infty$ and $\infty \neq -\infty$. Ouch.

What about:

$$\infty + -\infty = \text{NaN}$$

$$x + \text{NaN} = \text{NaN}$$

$$a + (b + c) \neq (a + b) + c$$

$$a * (b + c) \neq a * b + a * c$$

Motivating Example

So, we know that: $+0.0 - (-x) \neq x$.

Who cares? $+0.0 == -0.0$ is **true**.

Well, $\frac{1}{0} = \infty$, $\frac{1}{-0} = -\infty$ and $\infty \neq -\infty$. Ouch.

What about:

$$\infty + -\infty = \text{NaN}$$

$$x + \text{NaN} = \text{NaN}$$

$$a + (b + c) \neq (a + b) + c$$

$$a * (b + c) \neq a * b + a * c$$

Developers have to manually reason about edge cases.

Introduction

Alive¹ is a system for verifying peephole optimizations in LLVM.

Verification works as follows:

1. User specifies LLVM optimization.
2. Alive translates specification into SMT queries:
`src != tgt`
3. Alive uses Z3 to solve the SMT queries.

¹Nuno P Lopes et al. “Provably correct peephole optimizations with Alive”.
In: *PLDI*. 2015.

Introduction

Alive¹ is a system for verifying peephole optimizations in LLVM.

Verification works as follows:

1. User specifies LLVM optimization.
2. Alive translates specification into SMT queries:

src != tgt

3. Alive uses Z3 to solve the SMT queries.

Great, but no floating-point arithmetic.

LifeJacket = Alive + Floating-Point Arithmetic

¹Nuno P Lopes et al. “Provably correct peephole optimizations with Alive”.
In: *PLDI*. 2015.

Satisfiability Modulo Theory (SMT) solvers

Input

First-order logic formula extended with various functions and predicates.

Example

$$(x < y) \wedge (y < x - 1)$$

Output

A variable assignment that makes the formula true *or* unsatisfiable.

LifeJacket Example

(Incorrect) optimization from before

$+0.0 - (-x) \Rightarrow x$

In LifeJacket

```
%a = fsub -0.0, %x ←
```

```
%r = fsub +0.0, %a
```

\Rightarrow

```
%r = %x
```

LifeJacket Example

(Incorrect) optimization from before

$+0.0 - (-x) \Rightarrow x$

In LifeJacket

```
%a = fsub -0.0, %x ←
```

```
%r = fsub +0.0, %a
```

\Rightarrow

```
%r = %x
```

Note

No need to explicitly annotate type width.

LifeJacket Example

Input

```
%a = fsub -0.0, %x  
%r = fsub +0.0, %a  
=>  
%r = %x
```

Output

```
ERROR: Mismatch in values of f32 %r  
Example:  
%x f32 = -0.0 (0x8000000000000000)  
%a f32 = +0.0 (0x0000000000000000)  
Source value: +0.0 (0x0000000000000000)  
Target value: -0.0 (0x8000000000000000)
```

LifeJacket Example

Input

```
%a = fsub -0.0, %x
%r = fsub +0.0, %a
=>
%r = %x
```

Output

```
ERROR: Mismatch in values of f32 %r
Example:
%x f32 = -0.0 (0x8000000000000000)
%a f32 = +0.0 (0x0000000000000000)
Source value: +0.0 (0x0000000000000000)
Target value: -0.0 (0x8000000000000000)
```

Note

Precise optimizations.

Agenda

1. Floating-point types and instructions
2. Fast-math flags

Not today: Floating-point predicates

Agenda

1. Floating-point types and instructions
2. Fast-math flags

Not today: Floating-point predicates

LifeJacket: Floating-point types and instructions

Type support

half, single, double

Binary instructions

fadd, fsub, fmul, fdiv, frem

Conversions

fptrunc, fpext, fptoui, fptosi, uitofp, sitofp

Other

fabs, fcmp

LifeJacket: Floating-point types and instructions

Type support

half, single, double

Binary instructions

fadd, fsub, fmul, fdiv, frem

Conversions

fptrunc, fpext, fptoui, fptosi, uitofp, sitofp

Other

fabs, fcmp

Basic operations: direct translation to SMT-LIB operation

Agenda

1. Floating-point types and instructions
2. Fast-math flags

Not today: Floating-point predicates

LifeJacket: Fast-Math Flags

Example

```
%a = fsub nnan ninf C0, %x
```

LifeJacket supports three fast-math flags on instructions:

- **nnan**: Assume arguments and result are not **NaN**. Result undefined over **NaNs**.
- **ninf**: Assume arguments and result are not $\pm\infty$. Result undefined over $\pm\infty$.
- **nsz**: Allow optimizations to treat the sign of a zero argument or result as insignificant.

LifeJacket: Fast-Math Flags Example

Example

$$x + (0 - x) \Rightarrow 0$$

LifeJacket: Fast-Math Flags Example

Example

$$x + (0 - x) \Rightarrow 0$$

In LifeJacket

Precondition: AnyZero(C0)

```
%a = fsub nnan ninf C0, %x
```

```
%r = fadd %x, %a
```

```
=>
```

```
%r = 0.0
```

LifeJacket: Fast-Math Flags Example

Example

$$x + (0 - x) \Rightarrow 0$$

In LifeJacket

Precondition: AnyZero(C0)

```
%a = fsub nnan ninf C0, %x
```

```
%r = fadd %x, %a
```

```
=>
```

```
%r = 0.0
```

Translation of nnan/ninf:

If C0 or %x or %a is NaN/ $\pm\infty$ then %a unconstrained

LifeJacket: Fast-Math Flags Example

Example

$$x + (0 - x) \Rightarrow 0$$

In LifeJacket

Precondition: AnyZero(C0)

```
%a = fsub nnan ninf C0, %x
```

```
%r = fadd %x, %a
```

```
=>
```

```
%r = 0.0
```

Translation of nnan/ninf:

If C0 or %x or %a is NaN/ $\pm\infty$ then %a unconstrained

Correct?

LifeJacket: Fast-Math Flags Example

Example

$$x + (0 - x) \Rightarrow 0$$

In LifeJacket

```
Precondition: AnyZero(C0)
%a = fsub nnan ninf C0, %x
%r = fadd %x, %a
    =>
%r = 0.0
```

Translation of nnan/ninf:

If C0 or %x or %a is NaN/ $\pm\infty$ then %a unconstrained

Correct? No, consider %x = NaN.

Results

Results (LLVM 3.7.1)

File	Verified	Timeouts	Bugs
AddSub	7	1	1
MulDivRem	3	2	1
Compares	11	0	0
Simplify	22	0	6
Total	43	3	8

Results (LLVM 3.7.1)

File	Verified	Timeouts	Bugs
AddSub	7	1	1
MulDivRem	3	2	1
Compares	11	0	0
Simplify	22	0	6
Total	43	3	8

Insights

Few timeouts, high bug rate, bugs related to floating-point properties.

Limitations

LifeJacket currently has some limitations:

- No support for types wider than 64-bit.
- Fixed rounding mode.
- No support for floating-point exceptions/debug information in **NaNs**.

Limitations

LifeJacket currently has some limitations:

- No support for types wider than 64-bit.
- Fixed rounding mode.
- No support for floating-point exceptions/debug information in **NaNs**.

But: Errors found are true errors.


Conclusion

Automatic verification is possible: 43 verified optimizations.

Automatic verification is necessary: 8 bugs.

More automation = More optimizations, more boring compilers.

Would you like to know more?

 <https://github.com/4tXJ7f/alive>

 noetzli@stanford.edu