Optimization Coaching

Vincent St-Amour
Sam Tobin-Hochstadt
Matthias Felleisen

PLT

OOPSLA 2012 - October 23th, 2012
(define x (5))

(define x (6.5))

(define x (a))

(define x (b))
It renders text, too!
Now with "images/private" first:

```racket
(define-values (f s n) (make-values 3))
```

```
... 2 hours later...

Neil I
```

```racket
(define-values (f s n) (make-values 3))
```
define-syntax-rule
  (define (fomap-lift-helper f) ...)

[Redacted]
There must be a better way.
```racket
#lang typed/racket/base

(define/inline build-flomap
  (define fm x y)
  (define fmmin (flomap-normalize fm))
  (define fmabs (flomap-abs fmmin))
  (define fmmax (flomap-max fmabs))
  (define fmax-pos (flomap-max-index fmmax))
  (define fmexp (flomap-extract-fmexp fmmax))
  (define lerp-factor (floor (real->double-flonum (x y))))
  (define s (flomap-lift-helper fmexp lerp-factor))
  (define (flomap-lift-helper s x y)
    (match s
      [0 (s (x y))]
      [1 (s (x y))]
      [2 (s (x y))])
  )

(define/inline build-flomap-cw-rotate
  (define fm x y)
  (define fmmin (flomap-normalize fm))
  (define fmabs (flomap-abs fmmin))
  (define fmmax (flomap-max fmabs))
  (define fmax-pos (flomap-max-index fmmax))
  (define fmexp (flomap-extract-fmexp fmmax))
  (define lerp-factor (floor (real->double-flonum (x y))))
  (define s (flomap-lift-helper fmexp lerp-factor))
  (define (flomap-lift-helper s x y)
    (match s
      [0 (s (x y))]
      [1 (s (x y))]
      [2 (s (x y))])
  )
```

Missed Inlining (0 success out of 46) Consider turning this function into a macro to force inlining.
Dialog between compilers and programmers

404:34:

(/ (- (* a2 z2) (* a1 z1))
 (flsqrt (+ (* u u) (* v v))))

✓ Float arithmetic specialization.

20:0:

flmap-lift-helper

✗ Missed Inlining (0 success out of 46)
Consider turning this function into a macro to force inlining.
Dialog between compilers and programmers

Successes

404:34:

(/ (- (* a2 z2) (* a1 z1))
  (fsqrt (+ (* u u) (* v v))))

- Float arithmetic specialization.

20:0:

fomap-lift-helper

- Missed Inlining (0 success out of 46)
  Consider turning this function into a macro to force inlining.
Dialog between compilers and programmers

**Successes**

404:34:

```
(/ (- (* a2 z2) (* a1 z1))
   (fsqrt (+ (* u u) (* v v)))))
```

- Float arithmetic specialization.

**Near misses**

20:0:

```
flmap-lift-helper
```

- Missed Inlining (0 success out of 46)

  Consider turning this function into a macro to force inlining.

**Recommendations**
Programmers can do more than compilers

Recommendations can change semantics!

(\(\frac{1}{3}\))

1/3

\(\frac{1.0}{3.0}\) \(\rightarrow\) 0.3333333333333333
How does it work?
Overview

Compiler Instrumentation

Optimization Analysis

Recommendation Generation

Programmer Response
A day in the life of a near miss

#lang typed/racket/base

(define IM 139968)
(define IA 3877)
(define IC 29573)

(define last 42)
(define min 35.3)
(define max 156.8)
(define (gen-random)
  (set! last (modulo (+ (* last IA) IC) IM))
  (+ (/ (* (- max min) last) IM) min))
A day in the life of a near miss

#lang typed/racket/base

(define IM 139968)
(define IA 3877)
(define IC 29573)
(define last 42)
(define min 35.3)
(define max 156.8)
(define (gen-random)
  (set! last (modulo (+ (* last IA) IC) IM))
  (+ (/ (* (- max min) last) IM) min))
A day in the life of a near miss

(define IM 139968)
(define IA 3877)
(define IC 29573)
(define last 42)
(define min 35.3)
(define max 156.8)
(define (gen-random)
  (set! last (modulo (+ (* last IA) IC) IM))
  (+ (/ (* (- max min) last) IM) min))
Compiler Instrumentation

(- max min)

(fl-)

(fl- max min)

TR opt: prng-example.rkt 12:11
(- max min)
Float Float
binary float subtraction
Compiler Instrumentation

Float Integer

\((\ast (\neg \max \min) \text{last})\)

\((\ast <\text{Number}> <\text{Number}>) \); no change

\((\ast (\neg \max \min) \text{last})\)

TR opt failure: prng-example.rkt 12:8
\((\ast (\neg \max \min) \text{last})\)
Float Integer
generic multiplication
Optimization Analysis

Optimization proximity
Incomprehensible failure pruning
Irrelevant failure pruning
Harmless failure pruning
Irritant analysis
Causality merging
Locality merging
Optimization Analysis

Optimization proximity

Incomprehensible failure pruning
Irrelevant failure pruning
Harmless failure pruning
Irritant analysis
Causality merging
Locality merging
Optimization Analysis

Optimization proximity

\[ (\star (- \max \min) \text{last}) \]

\[ \Delta = 1 \]
Optimization Analysis

Optimization proximity

Near miss, report

\[ \Delta = 1 \]
Optimization Analysis

Optimization proximity

(*) last IA

Float
Integer

Float
Integer

Irritant
Irritant

\[ \Delta = 2 \]
Optimization Analysis

Optimization proximity

Too far, don't report

\[ \Delta = 2 \]
Recommendation Generation

\[(\ast (\neg \text{max min}) \text{last})\]

12:5:
\[/ (\ast (\neg \text{max min}) \text{last}) \text{IM}\]

This expression has a Real type. The optimizer could optimize it if it had type Float. To fix, change the highlighted expression(s) to have Float type(s).
12:5:

```
(+/(*(-max min)last)IM) min)
```

This expression has a Real type. The optimizer could optimize it if it had type Float. To fix, change the highlighted expression(s) to have Float type(s).
How well does it work?
Baseline: Non-optimized

Coached: Followed recommendations (Minutes of work)

Lower is better
Baseline: Non-optimized
Coached: Followed recommendations (Minutes of work)

Lower is better
Baseline: Non-optimized  
Coached: Followed recommendations  (Minutes of work)  
Gold standard: Hand-optimized by experts  (Days of work)  

Lower is better
The take-away

*Key idea*: The compiler *talks back*

*General* optimization analysis techniques

$+$ *Optimization-specific* heuristics

*Targeted* recommendations
The take-away

Key idea: The compiler talks back

General optimization analysis techniques
+ Optimization-specific heuristics

Targeted recommendations

racket-lang.org