Typing the Numeric Tower

Vincent St-Amour, Sam Tobin-Hochstadt,
Matthew Flatt, Matthias Felleisen

PLT

PADL 2012 - January 24th, 2012

Approaches to numerics

- Traditional

  Java, C(++), Fortran, ...

- Type Classes

  Haskell, Clean, ...

- Numeric Tower

  Racket, Scheme, Smalltalk, ...

Approaches to numerics

- Traditional

  Java, C(++), Fortran, ...

- Type Classes

  Haskell, Clean, ...

- Numeric Tower

  Racket, Scheme, Smalltalk, ...

Approaches to numerics

- Traditional

  Java, C(++), Fortran, ...

- Type Classes

  Haskell, Clean, ...

- Numeric Tower

  Racket, Scheme, Smalltalk, ...

$\}$ Typed Numeric Tower

Typed Racket

Our criteria

- Ease of expression
- Domain fidelity
- Static checking
- Performance

Our benchmarks

$$x_{i+1} \equiv A \cdot x_i \pmod{p}$$

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

Type Classes

$$x_{i+1} \equiv A \cdot x_i \pmod{p}$$

```
p = (2^31) - 1
a = 7^5
x = unsafePerformIO $ newIORef 42

genRandom min max
  = do old <- readIORef x
       writeIORef x (mod (a * old) p)
       new <- readIORef x
       return $ min + (((max-min) * (fromInteger new))
                       / (fromInteger p))
```

Type Classes

$$x_{i+1} \equiv A \cdot x_i \pmod p$$

```
p = (2^31) - 1
a = 7^5
x = unsafePerformIO $ newIORef 42

genRandom min max
  = do old <- readIORef x
       writeIORef x (mod (a * old) p)
       new <- readIORef x
       return $ min + (((max-min) * (fromInteger new))
                       / (fromInteger p))

genRandom 2.3 7.4
  -- => 2.3016764082953687
```

Type Classes

$$x_{i+1} \equiv A \cdot x_i \pmod{p}$$

```
p = (2^31) - 1
a = 7^5
x = unsafePerformIO $ newIORef 42

genRandom min max
  = do old <- readIORef x
       writeIORef x (mod (a * old) p)
       new <- readIORef x
       return $ min + (((max-min) * (fromInteger new))
                      / (fromInteger p))

genRandom 2.3 7.4
  -- => 2.3016764082953687
genRandom (toRational 2) (toRational 7)
  -- => 9927678409 % 2147483647
```

Type Classes

Ease of expression ✗

```
p = (2^31) - 1
a = 7^5
x = unsafePerformIO $ newIORef 42

genRandom min max
  = do old <- readIORef x
       writeIORef x (mod (a * old) p)
       new <- readIORef x
       return $ min + (((max-min) * (fromInteger new))
                       / (fromInteger p))

genRandom 2.3 7.4
  -- => 2.3016764082953687
genRandom (toRational 2) (toRational 7)
  -- => 9927678409 % 2147483647
```

Type Classes

Ease of expression	✗
Domain fidelity	✓

```haskell
p = (2^31) - 1
a = 7^5
x = unsafePerformIO $ newIORef 42

genRandom min max
  = do old <- readIORef x
       writeIORef x (mod (a * old) p)
       new <- readIORef x
       return $ min + (((max-min) * (fromInteger new))
                       / (fromInteger p))

genRandom 2.3 7.4
  -- => 2.3016764082953687
genRandom (toRational 2) (toRational 7)
  -- => 9927678409 % 2147483647
```

Type Classes

Ease of expression	✗
Domain fidelity	✓
Static checking	✓

```
p = (2^31) - 1
a = 7^5
x = unsafePerformIO $ newIORef 42

genRandom min max
  = do old <- readIORef x
       writeIORef x (mod (a * old) p)
       new <- readIORef x
       return $ min + (((max-min) * (fromInteger new))
                      / (fromInteger p))

genRandom 2.3 7.4
  -- => 2.3016764082953687
genRandom (toRational 2) (toRational 7)
  -- => 9927678409 % 2147483647
```

Type Classes

Ease of expression	✗
Domain fidelity	✓
Static checking	✓
Performance	✓

```
p = (2^31) - 1
a = 7^5
x = unsafePerformIO $ newIORef 42

genRandom min max
  = do old <- readIORef x
       writeIORef x (mod (a * old) p)
       new <- readIORef x
       return $ min + (((max-min) * (fromInteger new))
                            / (fromInteger p))

genRandom 2.3 7.4
  -- => 2.3016764082953687
genRandom (toRational 2) (toRational 7)
  -- => 9927678409 % 2147483647
```

Numeric Tower

$$x_{i+1} \equiv A \cdot x_i \pmod{p}$$

```
(define p (- (expt 2 31) 1))
(define A (expt 7 5))
(define x 42) ; state of the PRNG

(define (gen-random min max)
  (set! x (modulo (* A x) p))
  (+ min (/ (* (- max min) x) p)))
```

Numeric Tower

$$x_{i+1} \equiv A \cdot x_i \pmod p$$

```
(define p (- (expt 2 31) 1))
(define A (expt 7 5))
(define x 42) ; state of the PRNG

(define (gen-random min max)
  (set! x (modulo (* A x) p))
  (+ min (/ (* (- max min) x) p)))

(gen-random 2.3 7.4) ; => 2.3016764082953687
```

Numeric Tower

$$x_{i+1} \equiv A \cdot x_i \pmod{p}$$

```
(define p (- (expt 2 31) 1))
(define A (expt 7 5))
(define x 42) ; state of the PRNG

(define (gen-random min max)
  (set! x (modulo (* A x) p))
  (+ min (/ (* (- max min) x) p)))

(gen-random 2.3 7.4) ; => 2.3016764082953687
(gen-random 2   7)   ; => 9927678409/2147483647
```

Numeric Tower

Ease of expression ✓

```
(define p (- (expt 2 31) 1))
(define A (expt 7 5))
(define x 42) ; state of the PRNG

(define (gen-random min max)
  (set! x (modulo (* A x) p))
  (+ min (/ (* (- max min) x) p)))

(gen-random 2.3 7.4) ; => 2.3016764082953687
(gen-random 2   7)   ; => 9927678409/2147483647
```

Numeric Tower

Ease of expression ✓
Domain fidelity ✓

```
(define p (- (expt 2 31) 1))
(define A (expt 7 5))
(define x 42) ; state of the PRNG

(define (gen-random min max)
  (set! x (modulo (* A x) p))
  (+ min (/ (* (- max min) x) p)))

(gen-random 2.3 7.4) ; => 2.3016764082953687
(gen-random 2   7)   ; => 9927678409/2147483647
```

Numeric Tower

Ease of expression	✓
Domain fidelity	✓
Static checking	✗

```
(define p (- (expt 2 31) 1))
(define A (expt 7 5))
(define x 42) ; state of the PRNG

(define (gen-random min max)
  (set! x (modulo (* A x) p))
  (+ min (/ (* (- max min) x) p)))

(gen-random 2.3 7.4) ; => 2.3016764082953687
(gen-random 2   7)   ; => 9927678409/2147483647
```

Numeric Tower

Ease of expression	✓
Domain fidelity	✓
Static checking	✗
Performance	✗

```
(define p (- (expt 2 31) 1))
(define A (expt 7 5))
(define x 42) ; state of the PRNG

(define (gen-random min max)
  (set! x (modulo (* A x) p))
  (+ min (/ (* (- max min) x) p)))

(gen-random 2.3 7.4) ; => 2.3016764082953687
(gen-random 2   7)   ; => 9927678409/2147483647
```

Type Classes

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

Type Classes

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

$2.4x^2 + 36.2x - 7.5$

```
q 2.4 36.2 (-7.5)
  -- => 0.20441209042786124
```

Type Classes

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

$2.4x^2 + 36.2x - 7.5$

```
q 2.4 36.2 (-7.5)
  -- => 0.20441209042786124
```

$x^2 + 3x - 4$

```
q (fromInteger 1) (fromInteger 3) (fromInteger (-4))
  -- => 1.0
```

Type Classes

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

$2.4x^2 + 36.2x - 7.5$

```
q 2.4 36.2 (-7.5)
  -- => 0.20441209042786124
```

$x^2 + 3x - 4$

```
q (fromInteger 1) (fromInteger 3) (fromInteger (-4))
  -- => 1.0
```

$x^2 + 3x + 3$

```
q (fromInteger 1) (fromInteger 3) (fromInteger 3)
  -- => NaN
```

Type Classes

Ease of expression ✗

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

$2.4x^2 + 36.2x - 7.5$

```
q 2.4 36.2 (-7.5)
  -- => 0.20441209042786124
```

$x^2 + 3x - 4$

```
q (fromInteger 1) (fromInteger 3) (fromInteger (-4))
  -- => 1.0
```

$x^2 + 3x + 3$

```
q (fromInteger 1) (fromInteger 3) (fromInteger 3)
  -- => NaN
```

Type Classes

Ease of expression ✗
Domain fidelity ✗

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

$2.4x^2 + 36.2x - 7.5$

```
q 2.4 36.2 (-7.5)
  -- => 0.20441209042786124
```

$x^2 + 3x - 4$

```
q (fromInteger 1) (fromInteger 3) (fromInteger (-4))
  -- => 1.0
```

$x^2 + 3x + 3$

```
q (fromInteger 1) (fromInteger 3) (fromInteger 3)
  -- => NaN
```

Type Classes

Ease of expression	✗
Domain fidelity	✗
Static checking	✓

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

$2.4x^2 + 36.2x - 7.5$

```
q 2.4 36.2 (-7.5)
  -- => 0.20441209042786124
```

$x^2 + 3x - 4$

```
q (fromInteger 1) (fromInteger 3) (fromInteger (-4))
  -- => 1.0
```

$x^2 + 3x + 3$

```
q (fromInteger 1) (fromInteger 3) (fromInteger 3)
  -- => NaN
```

Type Classes

Ease of expression	✗
Domain fidelity	✗
Static checking	✓
Performance	✓

```
q :: (Floating a) => a -> a -> a -> a
q a b c = (-b + sqrt (b**2 - 4*a*c)) / (2*a)
```

$$2.4x^2 + 36.2x - 7.5$$

```
q 2.4 36.2 (-7.5)
  -- => 0.20441209042786124
```

$$x^2 + 3x - 4$$

```
q (fromInteger 1) (fromInteger 3) (fromInteger (-4))
  -- => 1.0
```

$$x^2 + 3x + 3$$

```
q (fromInteger 1) (fromInteger 3) (fromInteger 3)
  -- => NaN
```

Numeric Tower

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

Numeric Tower

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

$2.4x^2 + 36.2x - 7.5$

```
(q 2.4 36.2 -7.5) ; => 0.20441209042786124
```

Numeric Tower

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

$2.4x^2 + 36.2x - 7.5$

```
(q 2.4 36.2 -7.5) ; => 0.20441209042786124
```

$x^2 + 3x - 4$

```
(q 1   3    -4)   ; => 1
```

Numeric Tower

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

$2.4x^2 + 36.2x - 7.5$

```
(q 2.4 36.2 -7.5) ; => 0.20441209042786124
```

$x^2 + 3x - 4$

```
(q 1   3    -4)    ; => 1
```

$x^2 + 3x + 3$

```
(q 1   3    3)     ; => -1.5+0.8660254037844386i
```

Numeric Tower

Ease of expression ✓

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

$2.4x^2 + 36.2x - 7.5$

```
(q 2.4 36.2 -7.5) ; => 0.20441209042786124
```

$x^2 + 3x - 4$

```
(q 1   3    -4)    ; => 1
```

$x^2 + 3x + 3$

```
(q 1   3    3)     ; => -1.5+0.8660254037844386i
```

Numeric Tower

Ease of expression ✓
Domain fidelity ✓

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

$2.4x^2 + 36.2x - 7.5$

```
(q 2.4 36.2 -7.5) ; => 0.20441209042786124
```

$x^2 + 3x - 4$

```
(q 1   3    -4)   ; => 1
```

$x^2 + 3x + 3$

```
(q 1   3    3)    ; => -1.5+0.8660254037844386i
```

Numeric Tower

Ease of expression	✓
Domain fidelity	✓
Static checking	✗

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

$2.4x^2 + 36.2x - 7.5$

```
(q 2.4 36.2 -7.5) ; => 0.20441209042786124
```

$x^2 + 3x - 4$

```
(q 1   3    -4)    ; => 1
```

$x^2 + 3x + 3$

```
(q 1   3    3)     ; => -1.5+0.8660254037844386i
```

Numeric Tower

Ease of expression	✓
Domain fidelity	✓
Static checking	✗
Performance	✗

```
(define (q a b c)
  (/ (+ (- b) (sqrt (- (sqr b) (* 4 a c))))
     (* 2 a)))
```

$2.4x^2 + 36.2x - 7.5$

```
(q 2.4 36.2 -7.5) ; => 0.20441209042786124
```

$x^2 + 3x - 4$

```
(q 1   3    -4)   ; => 1
```

$x^2 + 3x + 3$

```
(q 1   3    3)    ; => -1.5+0.8660254037844386i
```

	Type Classes		Numeric Tower	
	PRNG	Quad	PRNG	Quad
Ease of expression	✗	✗	✓	✓
Domain fidelity	✓	✗	✓	✓
Static checking	✓	✓	✗	✗
Performance	✓	✓	✗	✗

	Type Classes		Numeric Tower		Typed Numeric Tower
	PRNG	Quad	PRNG	Quad	All programs
Ease of expression	✗	✗	✓	✓	✓
Domain fidelity	✓	✗	✓	✓	✓
Static checking	✓	✓	✗	✗	✓
Performance	✓	✓	✗	✗	✓

Typed Numeric Tower

Ease of expression	✓
Domain fidelity	✓
Static checking	✗
Performance	✗

Typed Numeric Tower

Ease of expression	✓	
Domain fidelity	✓	
Static checking	✓	Powerful type system
Performance	✗	

Typed Numeric Tower

Ease of expression	✓	
Domain fidelity	✓	
Static checking	✓	Powerful type system
Performance	✓	Type-driven optimization

The Type System

- Union types
- Function intersection types
- Occurrence typing

Union types

```
302 : Integer
302 : (U Integer Float)
3.2 : (U Integer Float)
```

Union types

```
302 : Integer
302 : (U Integer Float)
3.2 : (U Integer Float)
```

- No injections! No projections!

Union types

```
302 : Integer
302 : (U Integer Float)
3.2 : (U Integer Float)
```

- No injections! No projections!
  Ease of Expression ✓

Union types

```
302 : Integer
302 : (U Integer Float)
3.2 : (U Integer Float)
```

- No injections! No projections!
  Ease of Expression ✓

- No tags!

```
23.2 : Positive-Float
-3.2 : Negative-Float
```

Union types

```
302 : Integer
302 : (U Integer Float)
3.2 : (U Integer Float)
```

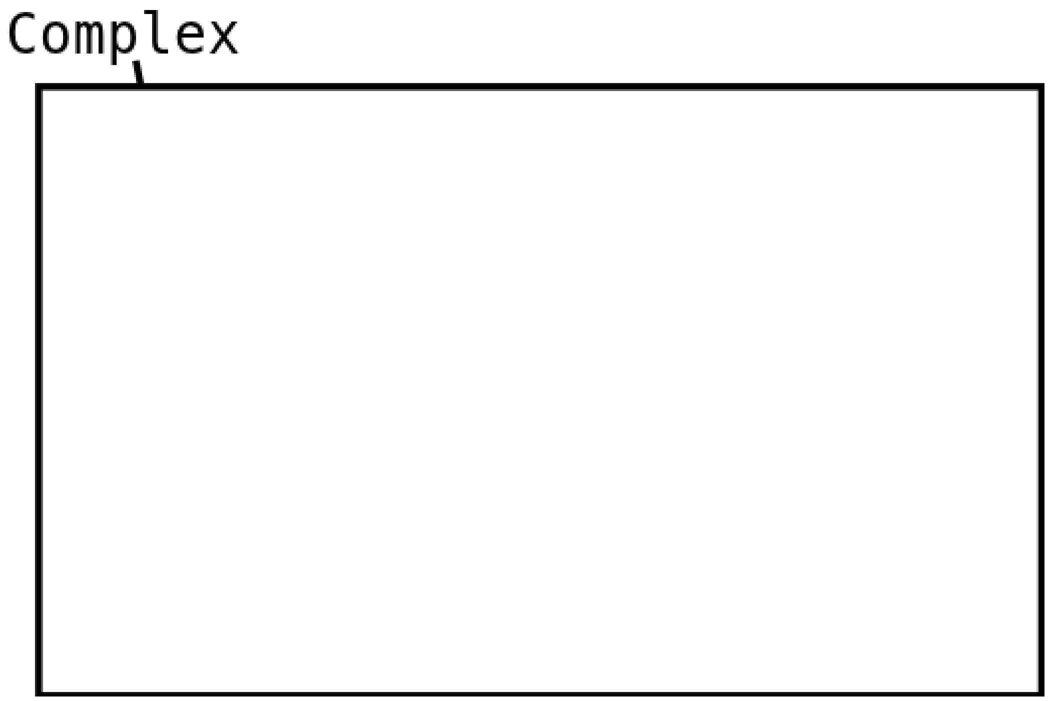
- No injections! No projections!
  Ease of Expression ✓

- No tags!

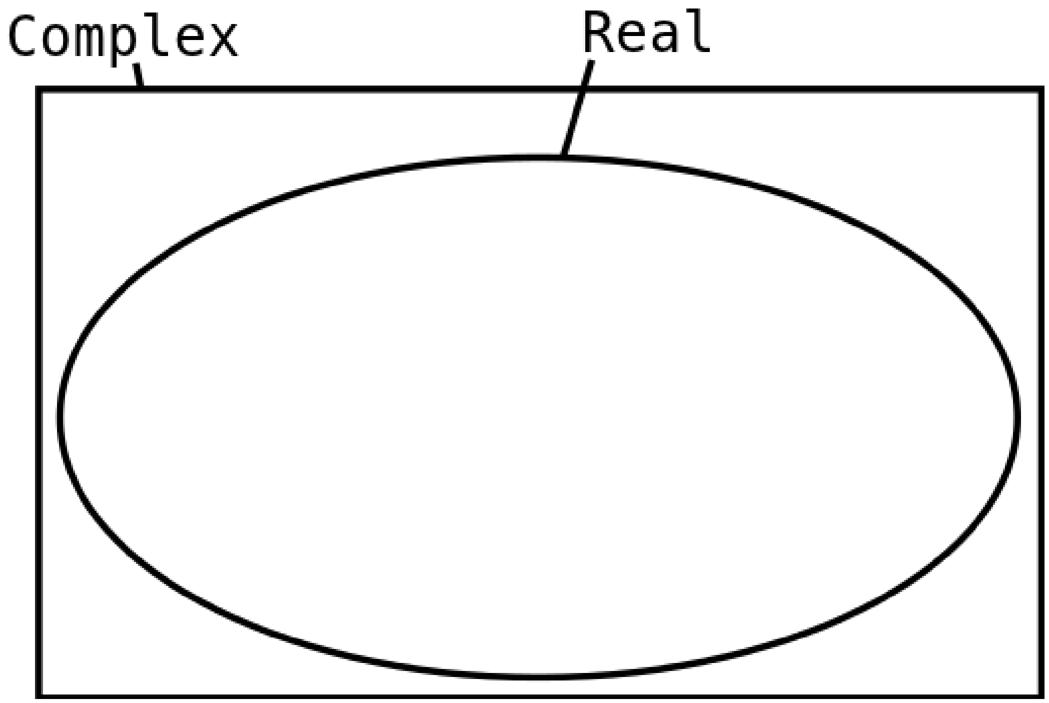
  ```
  23.2 : Positive-Float
  -3.2 : Negative-Float
  ```

  Representation independence
  Performance ✓

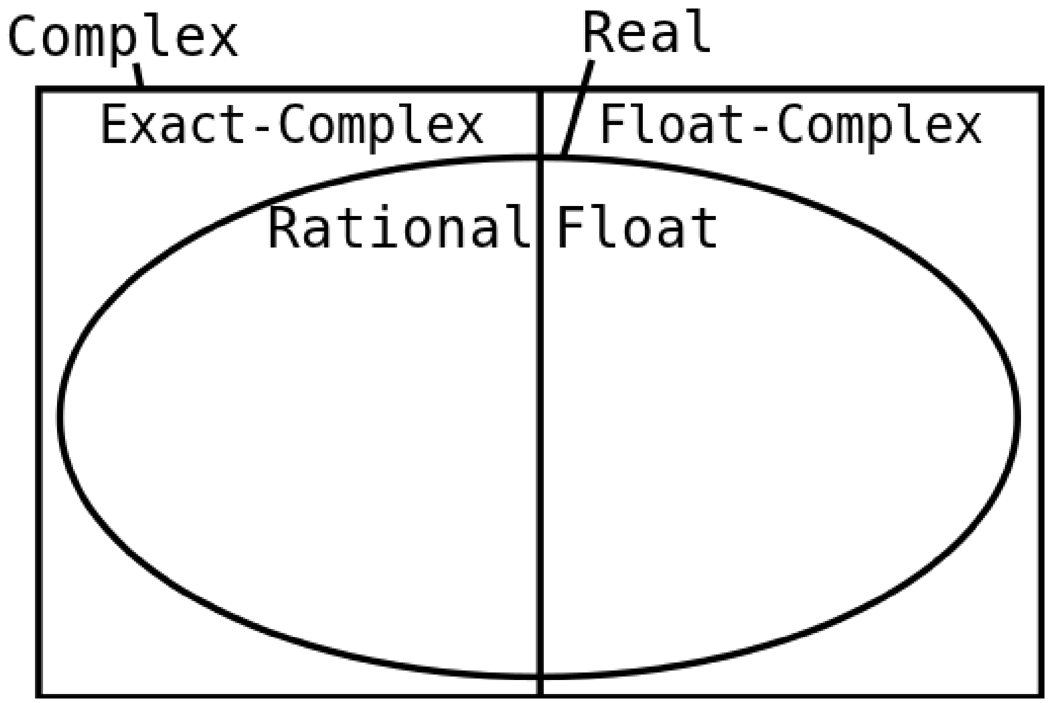
Union types



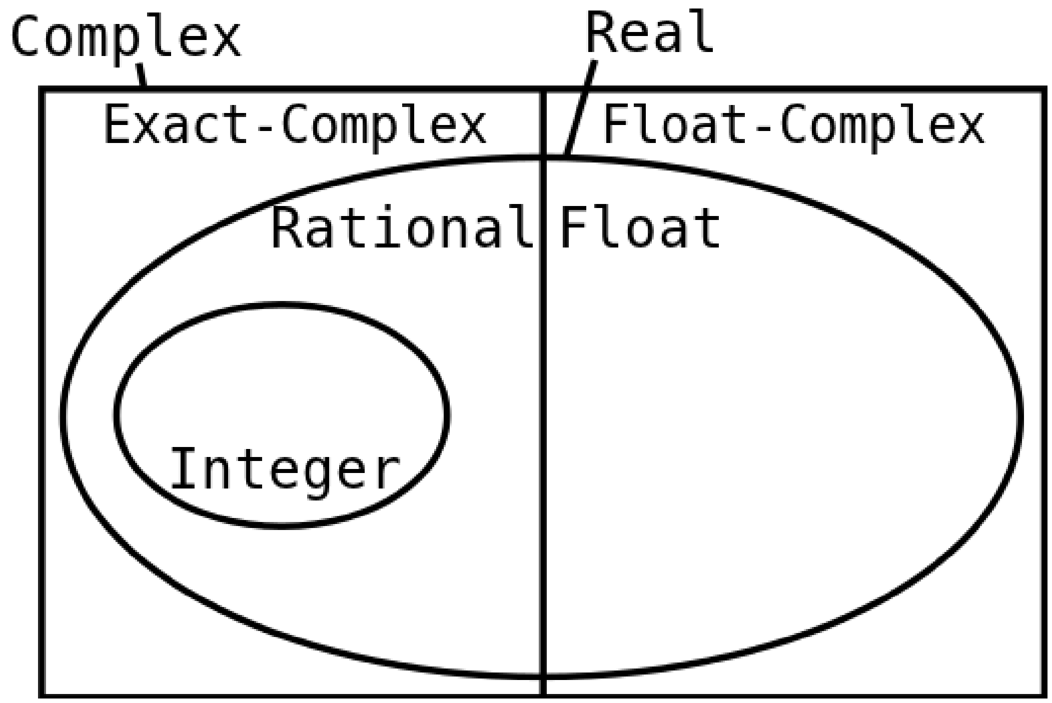
Union types



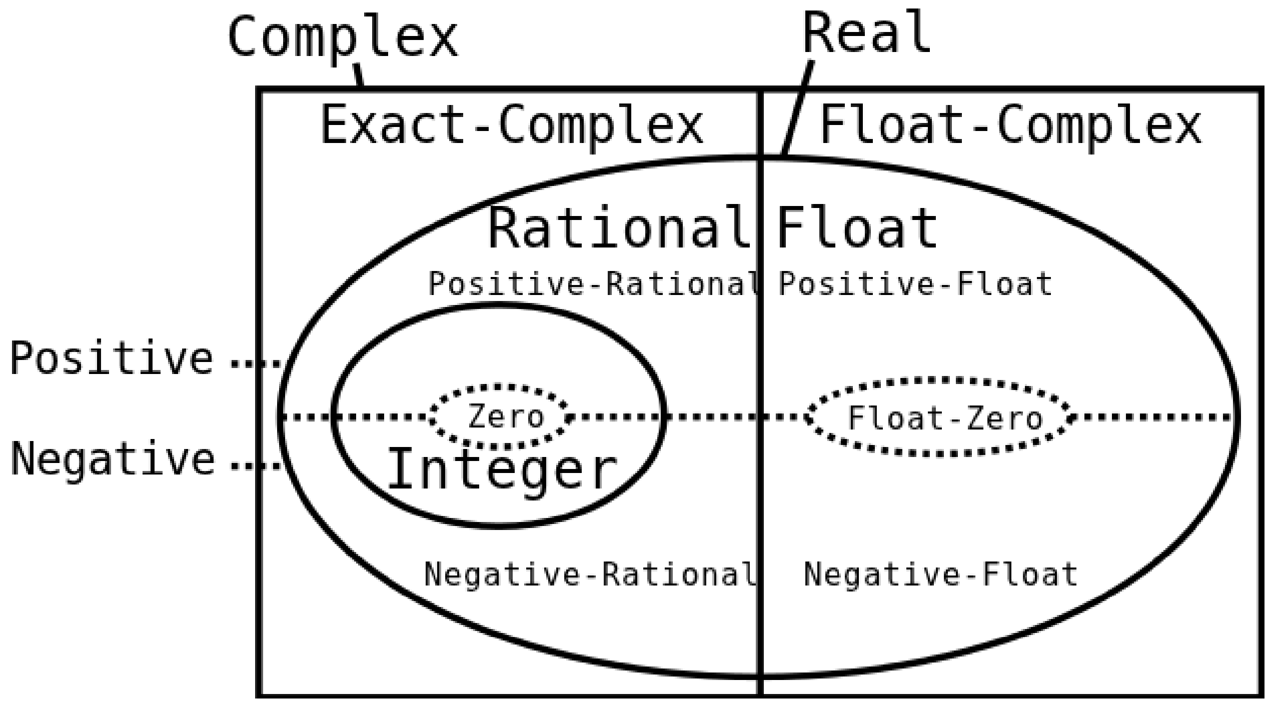
Union types



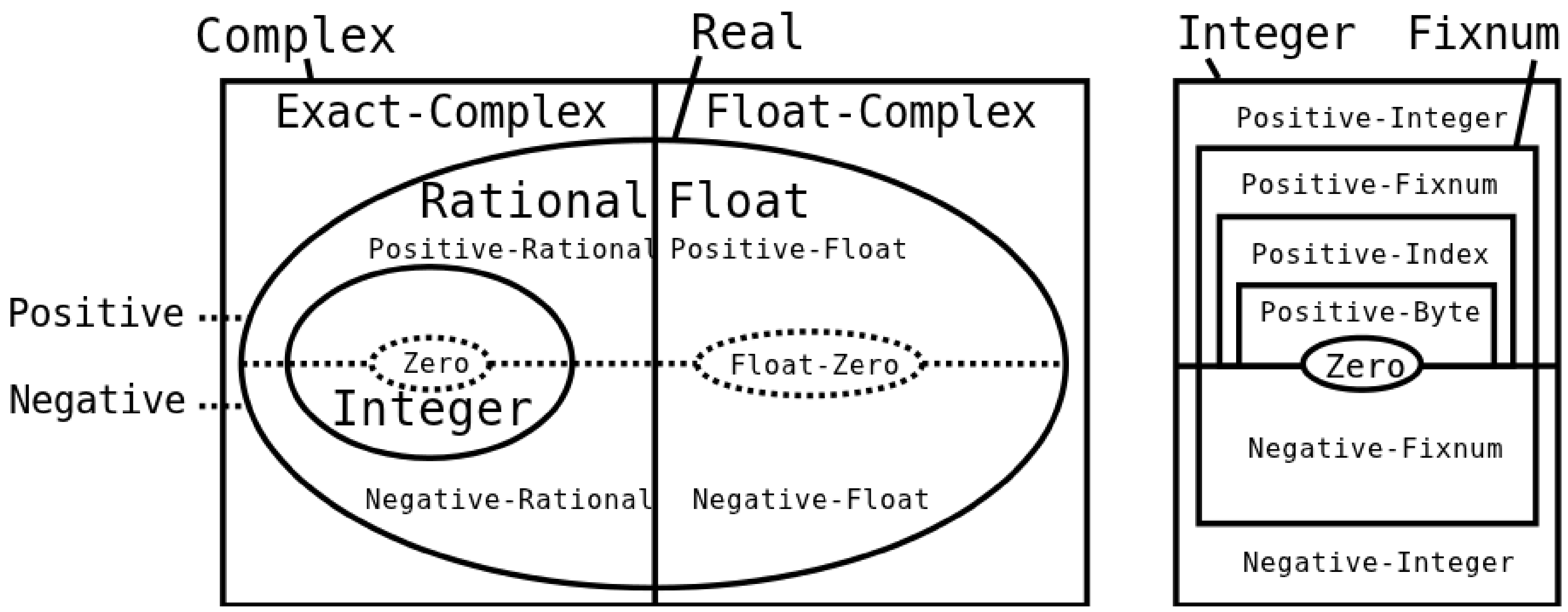
Union types


Complex
Real
Exact-Complex
Float-Complex
Rational
Float
Integer

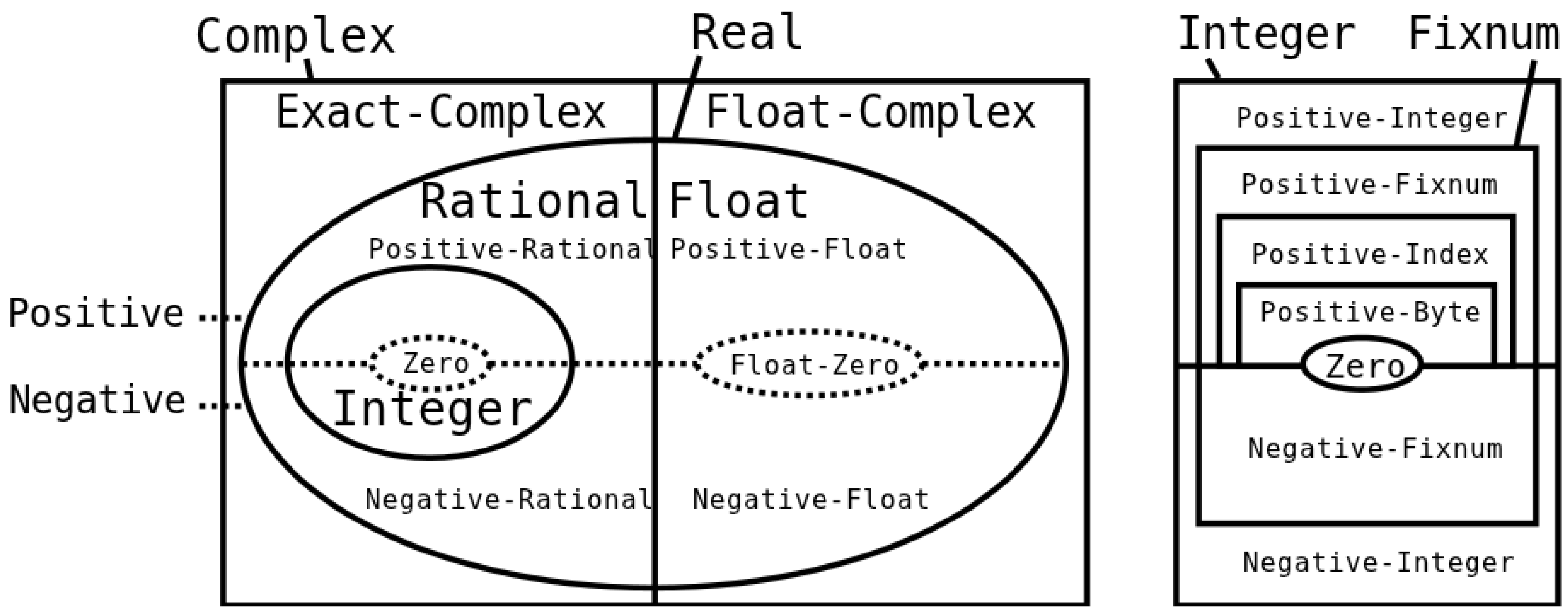
Union types


Complex
Real
Exact-Complex
Float-Complex
Rational
Float
Positive-Rational
Positive-Float
Positive
Zero
Float-Zero
Negative
Integer
Negative-Rational
Negative-Float

Union types


Complex
Real
Exact-Complex
Float-Complex
Rational
Float
Positive-Rational
Positive-Float
Positive
Negative
Zero
Float-Zero
Integer
Negative-Rational
Negative-Float
Integer
Fixnum
Positive-Integer
Positive-Fixnum
Positive-Index
Positive-Byte
Zero
Negative-Fixnum
Negative-Integer

Union types



Performance ✓

Function intersection types

```
(: f (case→ (Number → Number)
            (String → Number)))
```

Function intersection types

```
(: f (case→ (Number → Number)
            (String → Number)))

(define (f x) 0)
```

Function intersection types

```
(: f (case→ (Number → Number)
            (String → Number)))

(define (f x) 0)
```

Representation independence

Performance ✓

Function intersection types

```
(: + (case→ (Integer Integer → Integer)
            (Float   Float   → Float)

            …
            (Number  Number  → Number)))
```

Function intersection types

```
(: + (case→ (Integer Integer → Integer)
            (Float   Float   → Float)
            (Integer Float   → Float)
            …
            (Number  Number  → Number)))
```

Function intersection types

```
(: + (case→ (Integer Integer → Integer)
            (Float   Float   → Float)
            (Integer Float   → Float)
            …
            (Number  Number  → Number)))
```

Ease of expression ✓

```
(* (- max min) x)
```

Function intersection types

```
(: + (case→ (Integer Integer → Integer)
            (Float   Float   → Float)
            (Integer Float   → Float)
            …
            (Number  Number  → Number)))
```

Ease of expression ✓

```
(* (- max min) x)
```

```
(: sqrt (case→ (Nonnegative-Real → Nonnegative-Real)
               (Real → Complex)
               …))
```

Function intersection types

```
(: + (case→ (Integer Integer → Integer)
            (Float   Float   → Float)
            (Integer Float   → Float)
            ...
            (Number  Number  → Number)))
```

Ease of expression ✓

```
(* (- max min) x)
```

```
(: sqrt (case→ (Nonnegative-Real → Nonnegative-Real)
               (Real → Complex)
               ...))
```

Domain fidelity ✓

```
(q 1 3 3) ; => -1.5+0.8660254037844386i
```

Occurrence typing

```
(: abs : Real → Nonnegative-Real)
(define (abs x)
  (if (> x 0)
      x
      (- x)))
```

Occurrence typing

```
(: abs : Real → Nonnegative-Real)
(define (abs x)
  (if (> x 0)
      x
      (- x)))
```

$$\texttt{>} \;:\; \left(x : \texttt{Real}\ \ y : \texttt{Nonnegative-Real} \xrightarrow{\texttt{Positive}_x} \texttt{Boolean}\right)$$

Occurrence typing

```
(: abs : Real → Nonnegative-Real)
(define (abs x)
  (if (> x 0)
      x
      (- x)))
```

$$> \; : \; \left(x : \texttt{Real} \;\; y : \texttt{Nonnegative-Real} \xrightarrow{\texttt{Positive}_x} \texttt{Boolean}\right)$$

Ease of expression ✓

Type-driven optimization

- Arithmetic specialization
- Unboxing
- Arity raising of complex numbers
- ...

Type-driven optimization

- Arithmetic specialization
- Unboxing
- Arity raising of complex numbers
- ...

Mostly standard

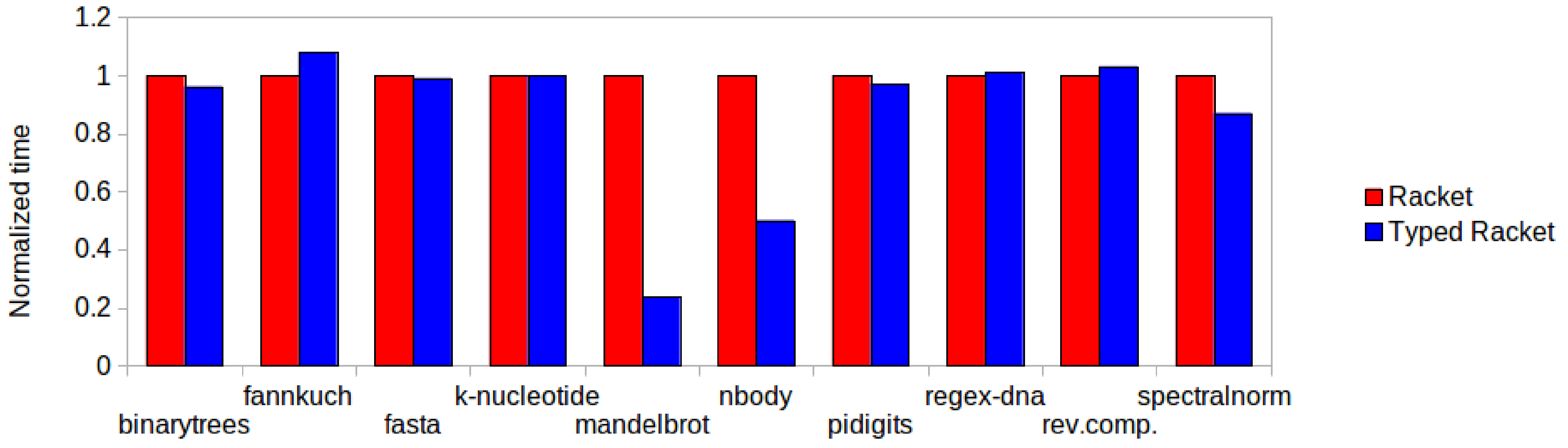
Type-driven optimization

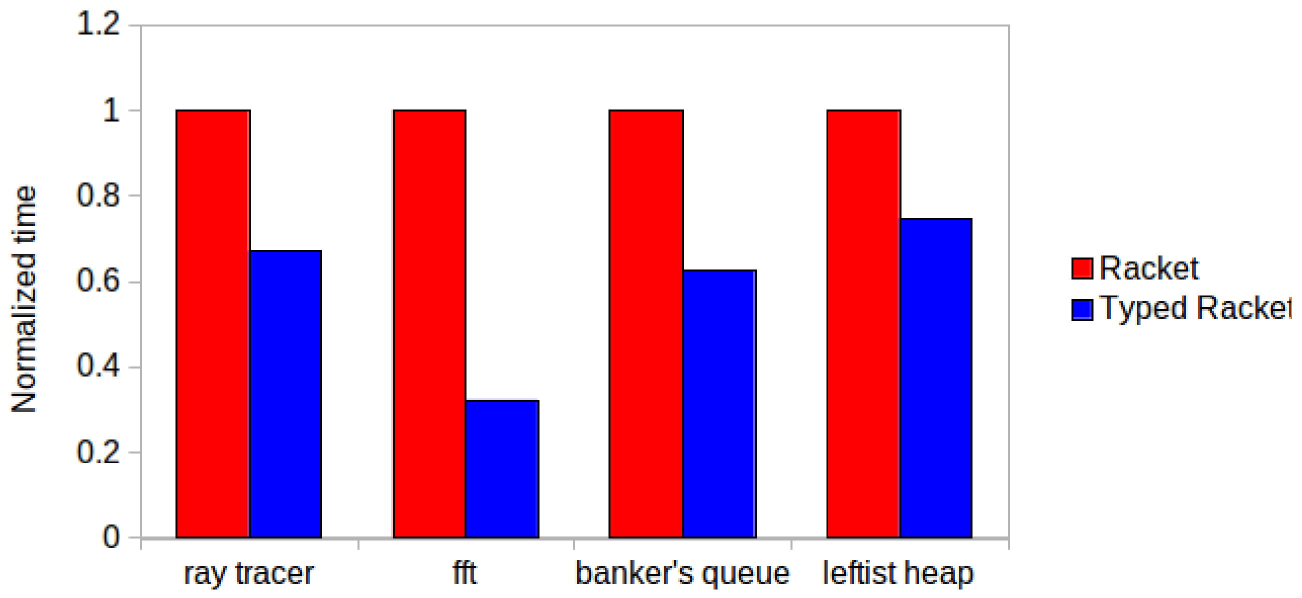
- Arithmetic specialization
- Unboxing
- Arity raising of complex numbers
- ...

Mostly standard

Representation independence

```
Positive-Float <: Float
```





Smaller is better

Usability

```
> (+ 1 "A")

Type Checker: No function domains
  matched in function application:
Domains:
  Zero Zero
  Zero Positive-Byte
  Byte Positive-Byte
  Byte Byte
... <snip 58 lines> ...
  Real Real
  Float-Complex Number
  Number Float-Complex
  Number Number
Arguments: Positive-Byte String
 in: (+ 1 "A")
```

```
[+ (from-cases
    (binop -Zero)
    (map (lambda (t) (commutative-binop t -Zero t))
         (list -One -PosByte -Byte -PosIndex -Index
               -PosFixnum -NonNegFixnum -NegFixnum -NonPosFixnum -Fixnum))
    (-> -PosByte -PosByte -PosIndex)
    (-> -Byte -Byte -Index)
    (-> -PosByte -PosByte -PosByte -PosIndex)
    (-> -Byte -Byte -Byte -Index)
    (commutative-binop -PosIndex -Index -PosFixnum)
    (-> -PosIndex -Index -Index -PosFixnum)
    (-> -Index -PosIndex -Index -PosFixnum)
    (-> -Index -Index -PosIndex -PosFixnum)
    (-> -Index -Index -NonNegFixnum)
    (-> -Index -Index -Index -NonNegFixnum)
    (commutative-binop -NegFixnum -One -NonPosFixnum)
    (commutative-binop -NonPosFixnum -NonNegFixnum -Fixnum)
    (commutative-case -PosInt -Nat -PosInt)
    (commutative-case -NegInt -NonPosInt -NegInt)
    (map varop (list -Nat -NonPosInt -Int))
    (commutative-case -PosRat -NonNegRat -PosRat)
    (commutative-case -NegRat -NonPosRat -NegRat)
    (map varop (list -NonNegRat -NonPosRat -Rat))
    (commutative-case -PosFlonum -NonNegReal -PosFlonum)
    (commutative-case -PosReal -NonNegFlonum -PosFlonum)
    (commutative-case -NegFlonum -NonPosReal -NegFlonum)
    (commutative-case -NegReal -NonPosFlonum -NegFlonum)
    (commutative-case -NonNegFlonum -NonNegReal -NonNegFlonum)
    (commutative-case -NonPosFlonum -NonPosReal -NonPosFlonum)
    (commutative-case -Flonum -Real -Flonum)
    (commutative-case -PosSingleFlonum (Un -NonNegRat -NonNegSingleFlonum) -PosSingleFlonum)
    (commutative-case (Un -PosRat -PosSingleFlonum) -NonNegSingleFlonum -PosSingleFlonum)
    (commutative-case -NegSingleFlonum (Un -NonPosRat -NonPosSingleFlonum) -NegSingleFlonum)
    (commutative-case (Un -NegRat -NegSingleFlonum) -NonPosSingleFlonum -NegSingleFlonum)
    (commutative-case -NonNegSingleFlonum (Un -NonNegRat -NonNegSingleFlonum) -NonNegSingleFlonum)
    (commutative-case -NonPosSingleFlonum (Un -NonPosRat -NonPosSingleFlonum) -NonPosSingleFlonum)
    (commutative-case -SingleFlonum (Un -Rat -SingleFlonum) -SingleFlonum)
    (commutative-case -PosInexactReal -NonNegReal -PosInexactReal)
    (commutative-case -PosReal -NonNegInexactReal -PosInexactReal)
    (commutative-case -NegInexactReal -NonPosReal -NegInexactReal)
    (commutative-case -NegReal -NonPosInexactReal -NegInexactReal)
    (commutative-case -NonNegInexactReal -NonNegReal -NonNegInexactReal)
    (commutative-case -NonPosInexactReal -NonPosReal -NonPosInexactReal)
    (commutative-case -InexactReal -Real -InexactReal)
    (commutative-case -PosReal -NonNegReal -PosReal)
    (commutative-case -NegReal -NonPosReal -NegReal)
    (map varop (list -NonNegReal -NonPosReal -Real -ExactNumber))
    (commutative-case -FloatComplex N -FloatComplex)
    (commutative-case -Flonum -InexactComplex -FloatComplex)
    (commutative-case -SingleFlonumComplex (Un -Rat -SingleFlonum -SingleFlonumComplex) -SingleFlonumComplex)
    (commutative-case -InexactComplex (Un -Rat -InexactReal -InexactComplex) -InexactComplex)
    (varop N))]
```

56 lines of type DSL
22k of printout

```
> (+ 1 "A")

Type Checker: No function domains
  matched in function application:
Domains: Number Number
Arguments: Positive-Byte String
 in: (+ 1 "A")
```

```
(define x (box 3))
```

```
(define x (box 3)) : (Boxof Positive-Byte)
```

```
(define x (box 3)) : (Boxof Positive-Byte)

(set-box! x 2000)          ✗
```

```
(define x (box 3)) : (Boxof Number)

(set-box! x 2000)          ✓
```

```
(define x (box 3)) : (Boxof Number)

(set-box! x 2000)          ✓

(vector-ref v (unbox x))   ✗
```

```
(define x (box 3)) : (Boxof Natural)

(set-box! x 2000)          ✓

(vector-ref v (unbox x))   ✓
```

Typed Numeric Tower

Ease of expression	✓
Domain fidelity	✓
Static checking	✓
Performance	✓

Key type system features

- Union types
- Function intersection types
- Occurrence typing

Typed Numeric Tower

Ease of expression	✓
Domain fidelity	✓
Static checking	✓
Performance	✓

Key type system features

- Union types
- Function intersection types
- Occurrence typing

`racket-lang.org`