

# Compiler Testing with Sized Types

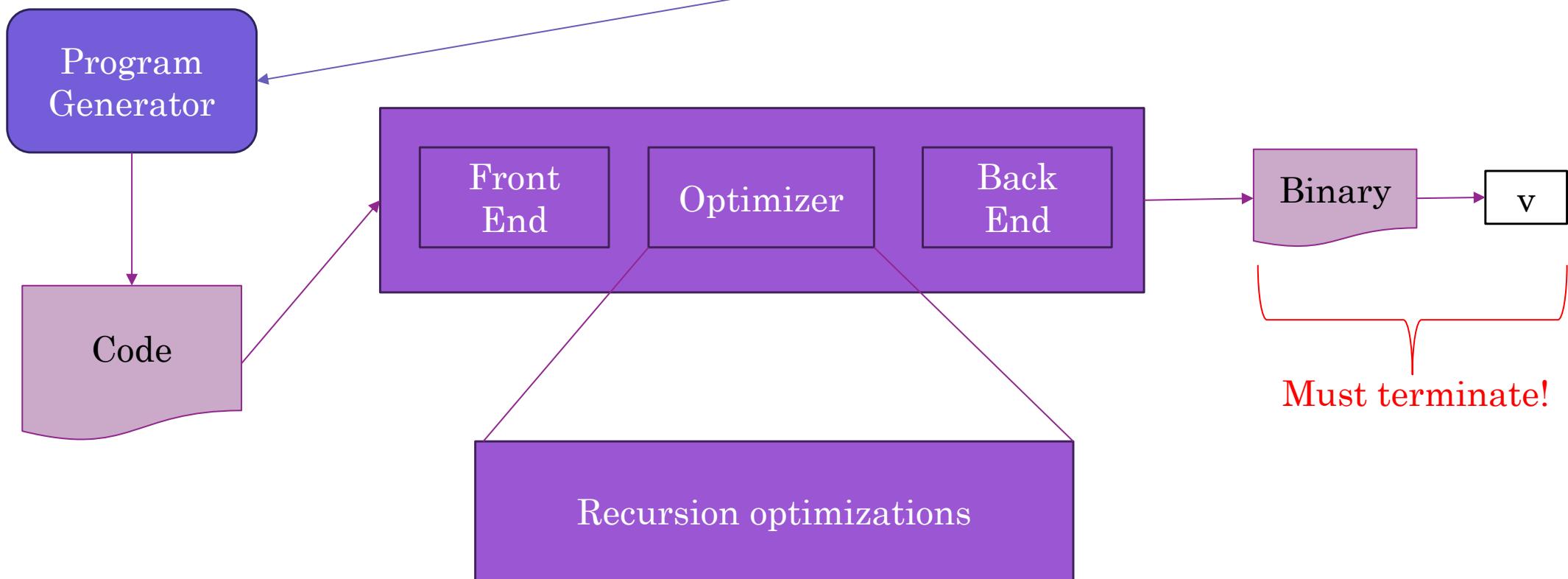
Caspar Popova, University of Maryland



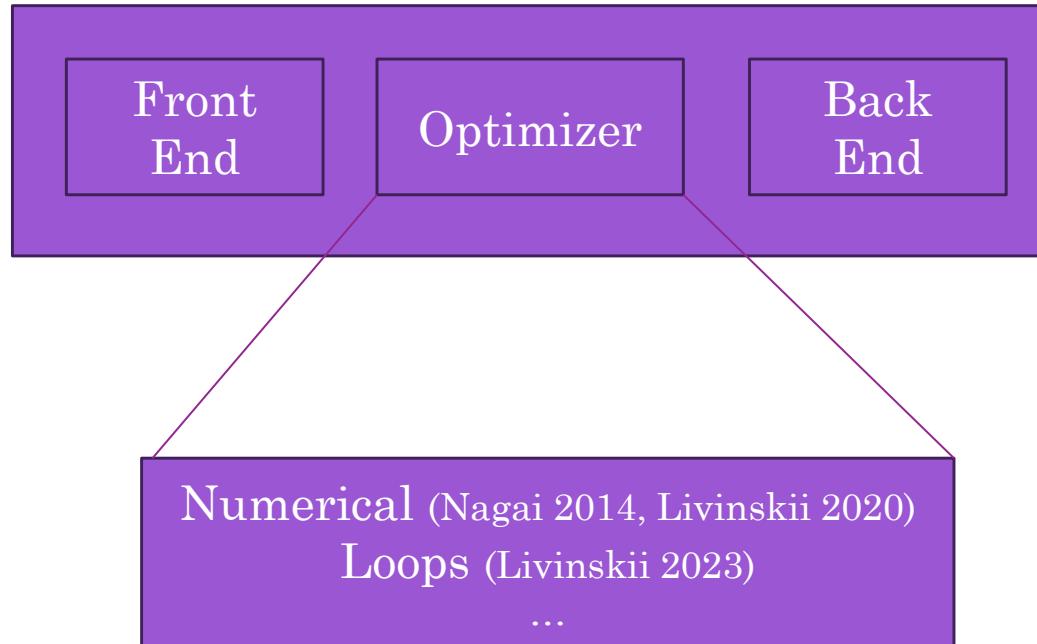
Joint work with Harry Goldstein, Leonidas Lampropoulos

NJPLS ~ December 05, 2025

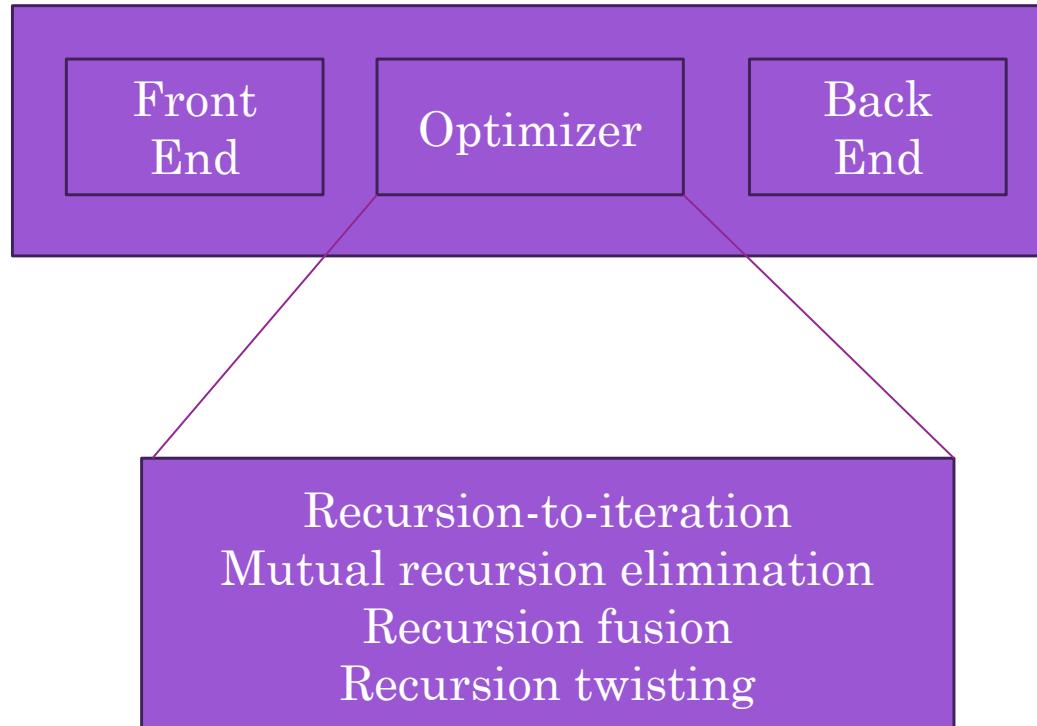
# Compiler Testing with Sized Types



# Optimization testing



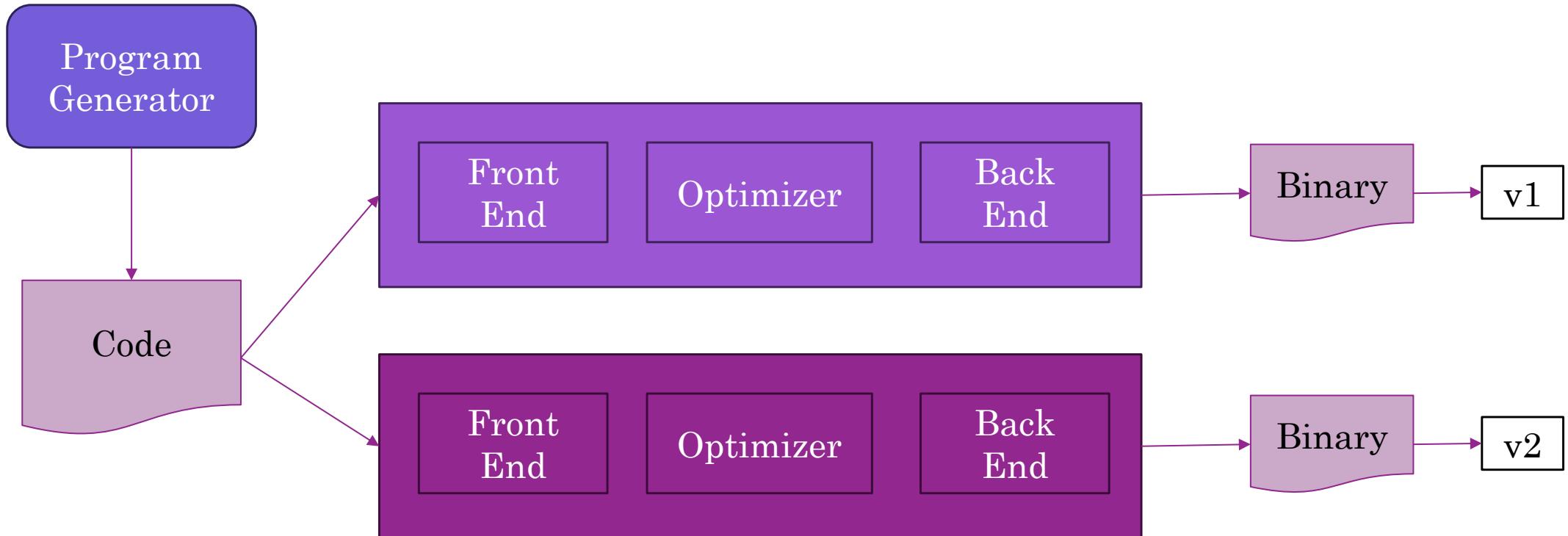
# Recursion Optimizations



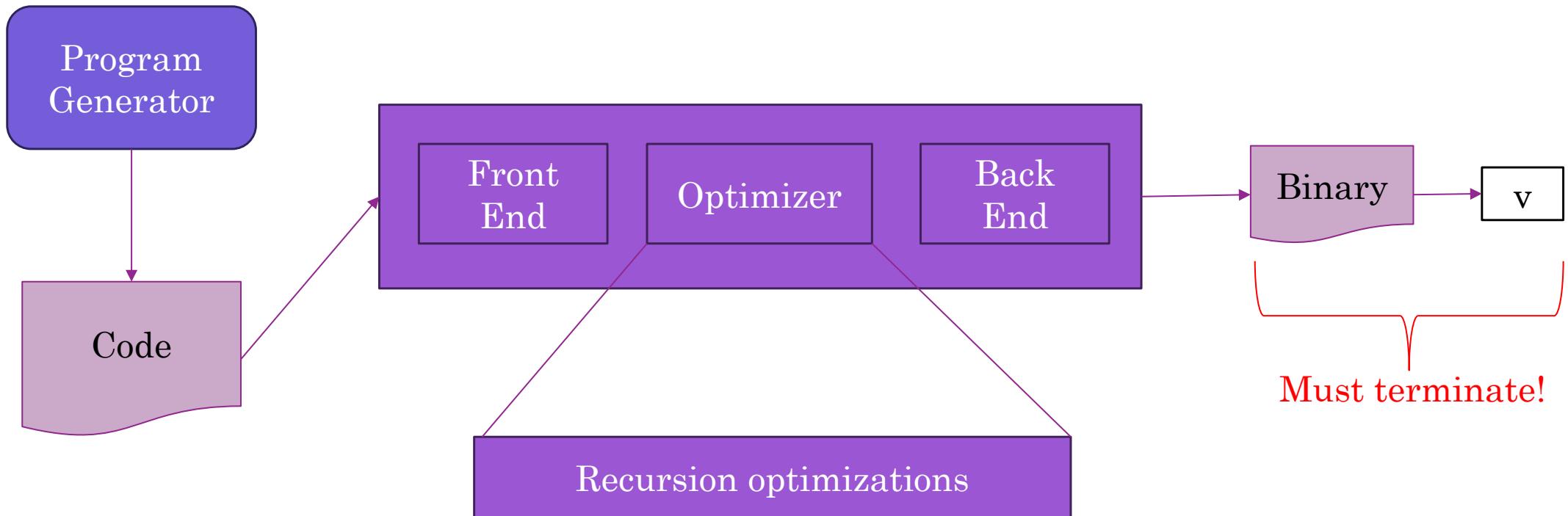
broken inlining  
heuristic  
(Racket#3027)



# Why do we need termination?



# Challenge: Termination



# Program Generation

$$\frac{\Gamma \vdash \square : \text{Nat} \rightsquigarrow e_1 : \text{Nat} \quad \Gamma \vdash \square : \text{Nat} \rightsquigarrow e_2 : \text{Nat}}{\Gamma \vdash \square : \text{Nat} \rightsquigarrow (e_1 + e_2)} \text{ ADD}$$

$$\square : \text{Nat} \rightsquigarrow (1 + 2)$$

# Program Generation

$$\frac{\Gamma, x : \tau_1, f : \tau_1 \rightarrow \tau_2 \vdash \square : \tau_2 \rightsquigarrow e}{\Gamma \vdash \square : \tau_1 \rightarrow \tau_2 \rightsquigarrow (\text{let rec } f \ x = e)} \text{ REC}$$

$\square : \text{Nat} \rightarrow \text{Nat} \rightsquigarrow$

```
let rec f x =
  match x with
  | 0 -> f x
  | S x' -> f x'
```

# Inspiration from termination checking

- Sized types → **type theoretic and compositional** 😊
- Guard predicate → used in Rocq; structural/syntactic condition, not compositional
- Well-founded relations → used in recursive program synthesis  
(Miltner 2024, Polikarpova 2024, Choi 2023); often requires proofs

# Sized Types to the rescue

```
Nat := o : Nati+1
      | s : Nati → Nati+1
```

Size annotations on  
inductive datatypes

```
add : Nati → Nat → Nat
minus : Nati → Nat → Nati
div : Nati → Nat → Nati
```

Size annotations on  
functions

# Sized Types to the rescue

$$\frac{\Gamma, \boxed{x : \text{Nat}^{i+1}}, \boxed{f : \text{Nat}^i \rightarrow \theta} \vdash \square : \boxed{\theta[i := i + 1]} \rightsquigarrow e}{\Gamma \vdash \square : \boxed{\forall i. \text{Nat}^i \rightarrow \theta} \rightsquigarrow (\text{let rec } f \ x = e)}$$
 REC

$\square : \text{Nat} \rightarrow \text{Nat} \rightsquigarrow$

`let rec f x =  
 f x`

$$\frac{i \leq i + 1}{\text{Nat}^i \sqsubseteq \text{Nat}^{i+1}} \text{ SUBTYPE}$$

Error! Does not terminate

# Sized Types to the rescue

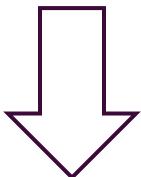
$$\frac{\Gamma \vdash \square : \boxed{\mathbf{Nat}^{i+1}} \rightsquigarrow e_1 \quad \Gamma \vdash \square : \theta \rightsquigarrow e_2 \quad \Gamma, \boxed{x' : \mathbf{Nat}^i} \vdash \square : \theta \rightsquigarrow e_3}{\Gamma \vdash \square : \theta \rightsquigarrow (\text{match } e_1 \text{ with } | o \rightarrow e_2 | s\ x' \rightarrow e_3)} \text{ MATCH}$$

$\square : \mathbf{Nat} \rightarrow \mathbf{Nat} \rightsquigarrow$   
`let rec f x =  
 match x with  
 | 0 -> e2  
 | S x' -> f x'`

# Sized Types & composition

```
let rec div n m =
  match n with
  | 0 -> 0
  | S n' -> S (div (□ : Nati) m)
```

$$\Gamma = \{\text{div} : \text{Nat}^i \rightarrow \text{Nat} \rightarrow \text{Nat}^i, \\ \text{minus} : \forall k. \text{Nat}^k \rightarrow \text{Nat} \rightarrow \text{Nat}^k, \\ n : \text{Nat}^{i+1}, \\ n' : \text{Nat}^i\}$$



```
let rec div n m =
  match n with
  | 0 -> 0
  | S n' -> S (div (minus n' m) m)
```

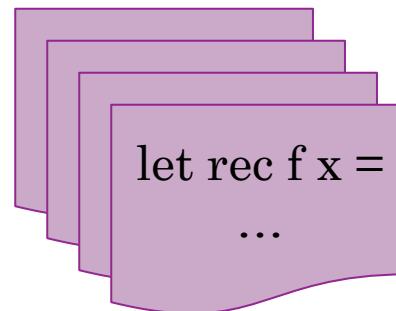
can be tricky with guard  
predicates & well-founded  
relations

# Ongoing Work

Implemented in OCaml

Remaining challenges:

- Mutual recursion generation
- Evaluation

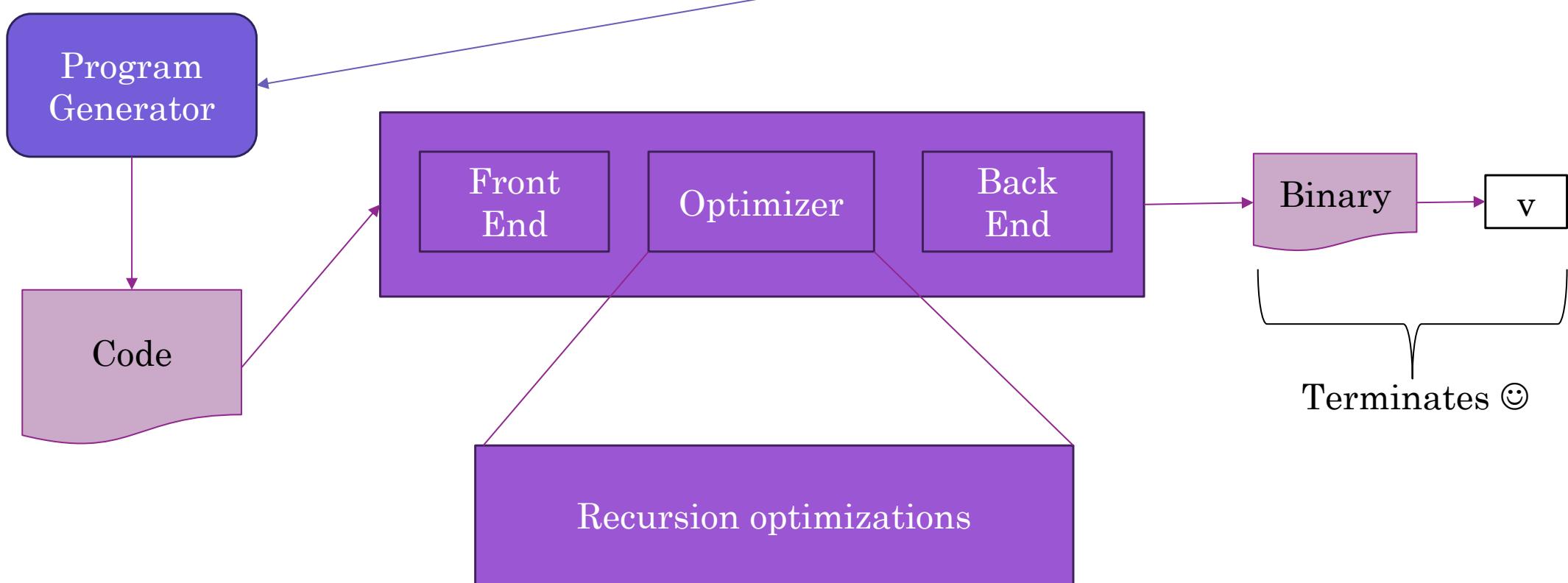


# (Planned) Evaluation

- Toy compiler with manually injected bugs (CMSC 430)
- Differential testing across Racket languages
- Differential testing across ML compilers



# Compiler Testing with Sized Types



Talk to me about recursion bugs & compiler optimization/testing! → Caspar Popova (caspar@umd.edu)