ELEVATE

A Language for Specifying Optimization Strategies

1 Wouldn't it be great...

...to ***look behind the curtains*** of optimizing compilers and actually understand how optimizations are applied?

...to define well-known optimizations once and ***reuse*** them whenever possible?

...to have ***one principled way*** to write compiler optimizations based on a strong theoretical foundation?

...to build your own optimizations using simple and ***composable building blocks***?

Optimizing Programs like it's ~~1998~~ 2019

Visser et. al.: Building program optimizers with rewriting strategies (ICFP 1998)

2 Basic Strategy Building Blocks

A ***Strategy*** is a function: ***Program → Program***

A ***Transformation*** is the simplest strategy

e.g. map(f) → join ∘ map(map f) ∘ splitJoin-Rule

id : Strategy
= λp . p

;seq : Strategy → Strategy → Strategy
= λf . λs . λp . s (f p)

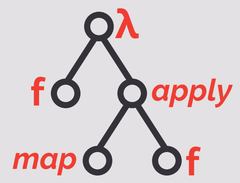
<+choice : Strategy → Strategy → Strategy
= λf . λs . λp . *try* (f p) *catch* (s p)

try : Strategy → Strategy
= λs . choice s id

repeat : Strategy → Strategy
= λs . try (s ; (repeat s))

norm : Strategy → Strategy
= λs . repeat(find s)

3 Traversals: Where to apply a strategy?

λf . map f → 

Expression AST

Generic one-level traversals: Strategy → Strategy

all ***one*** ***some***

Generic traversal strategies: Strategy → Strategy

topdown = λs . s ; all(topdown s)

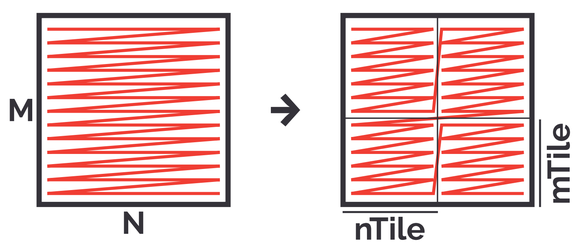
tryAll = λs . all(tryAll(try s)) ; (try s)

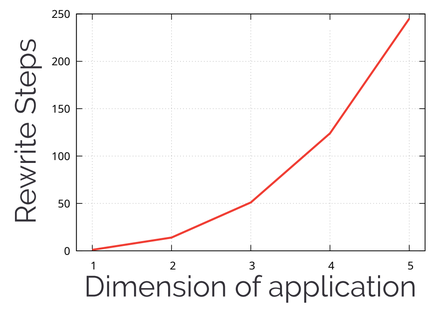
find = λs . s <+ one(find s)

⋮

4 Example: *Expressing tiling as a strategy*

in 3 simple steps...





a Tile every dimension

λn . tryAll(splitJoin n)

b Rewrite Normalform

norm(mapFission)

mapFission = *(f ∘ g) = *f ∘ *g

c Rearrange dimensions

```
λd . λp . d match
  case <2 : p
  case 2 : (shuffleDim d) p
  case _ : (rearrangeDim (d-1) ;
            (shuffleDim d)) p
```

Example input program: ***f*

**f → J ∘ **(J ∘ **f ∘ S) ∘ S →

J ∘ **J ∘ ****f ∘ **S ∘ S →

J ∘ **J ∘ *T ∘ ****f ∘ *T ∘ **S ∘ S

Lift (* = map | S = split(s) | J = join | T = transpose)

```
for(int i = 0; i < M; i++) {
 for(int j = 0; j < N; j++) {
    // f
}}
```

→

```
for(int i = 0; i < M/s; i++) {
 for(int ii = 0; ii < s; ii++) {
  for(int j = 0; j < N/s; j++) {
   for(int jj = 0; jj < s; jj++) {
        // f
}}}}
```

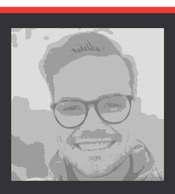
→

```
for(int i = 0; i < M/s; i++) {
 for(int j = 0; j < N/s; j++) {
  for(int ii = 0; ii < s; ii++) {
   for(int jj = 0; jj < s; jj++) {
        // f
}}}}
```

OpenCL

Bastian Hagedorn | **Michel Steuwer** | **Sergei Gorlatch**

University of Münster | University of Glasgow | University of Münster



b.hagedorn@wwu.de | bastianhagedorn.github.io