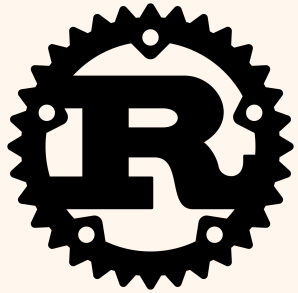
Towards Collection-Oriented Compilation in LLVM

Tommy McMichen
Ph.D. Candidate
Northwestern University

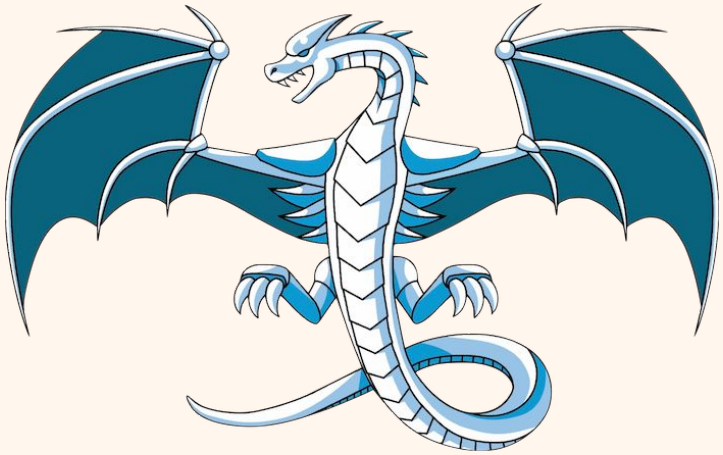
m&moir

Shared Representation across Languages

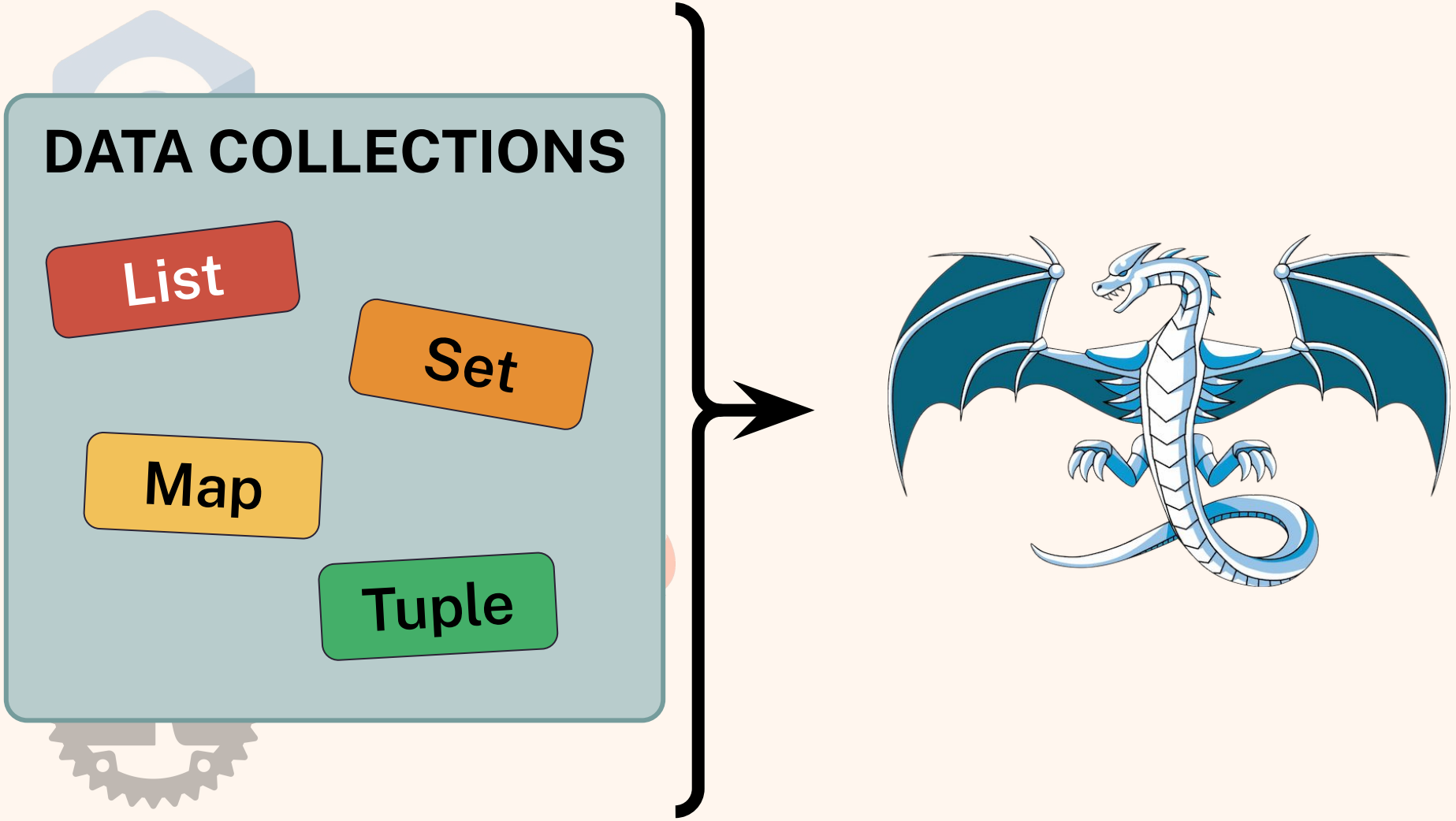


julia

Mojo 🔥



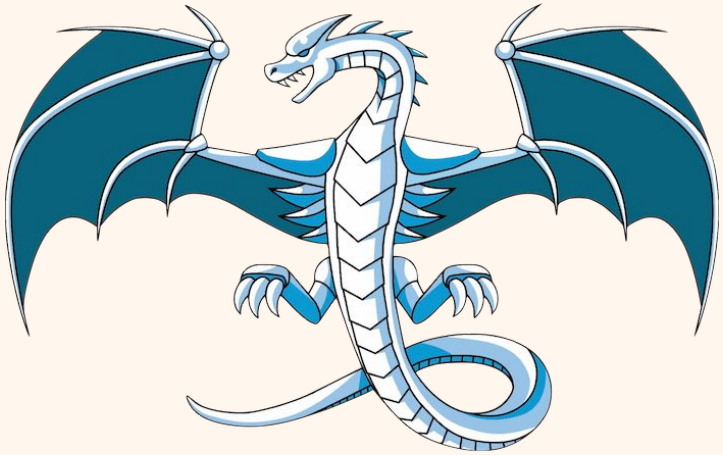
Shared Representation across Languages



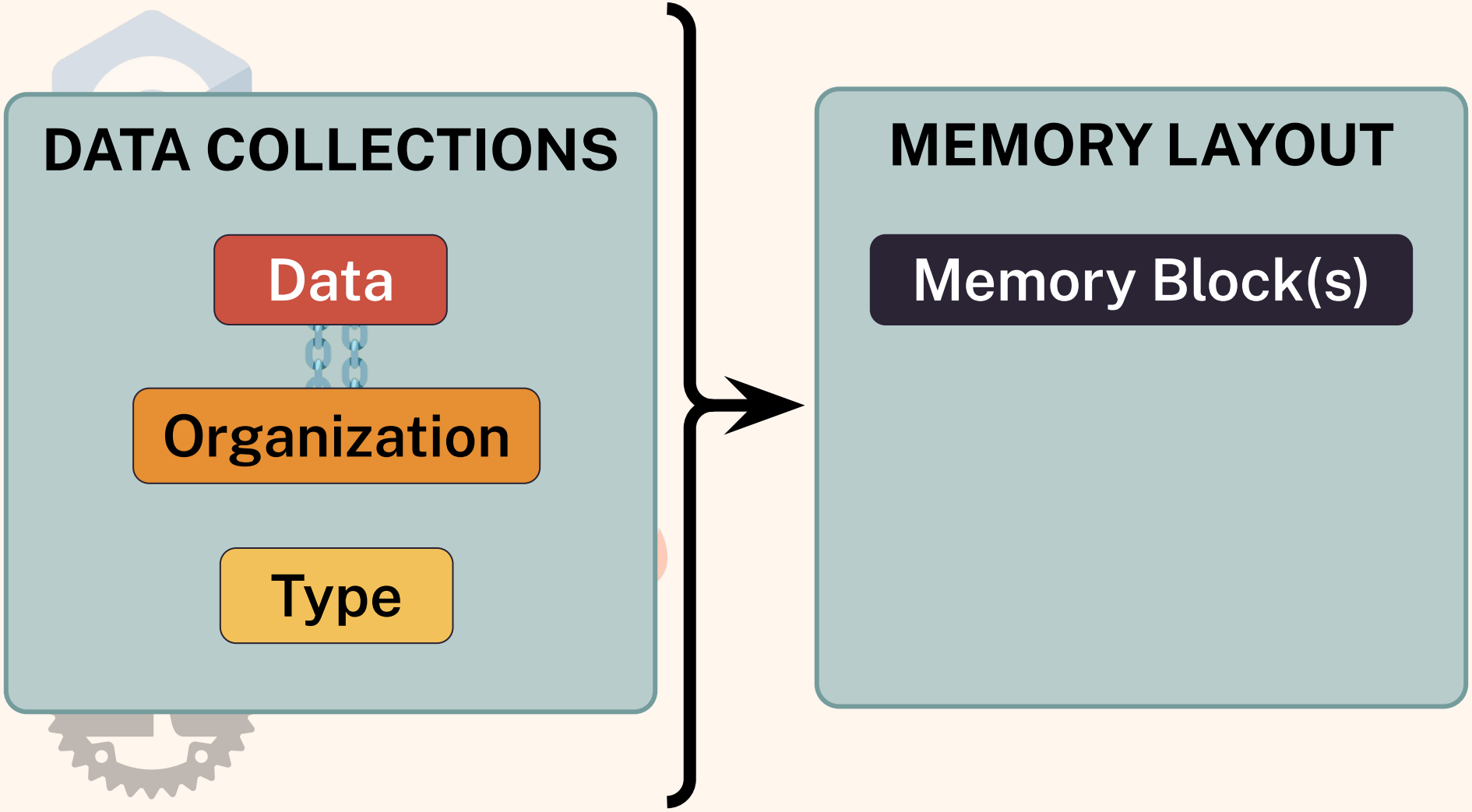
Shared Representation across Languages

DATA COLLECTIONS

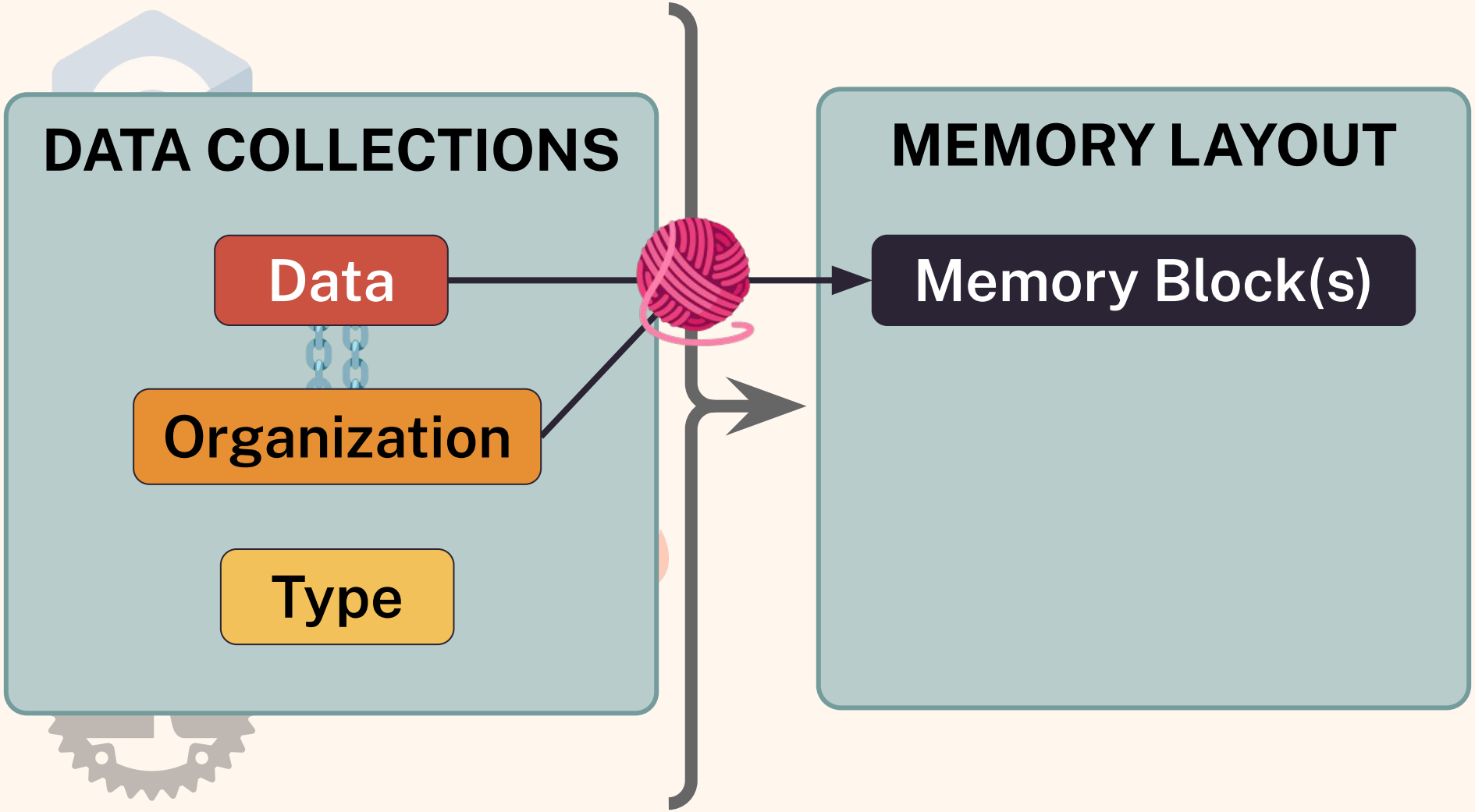
- Data
- Organization
- Type



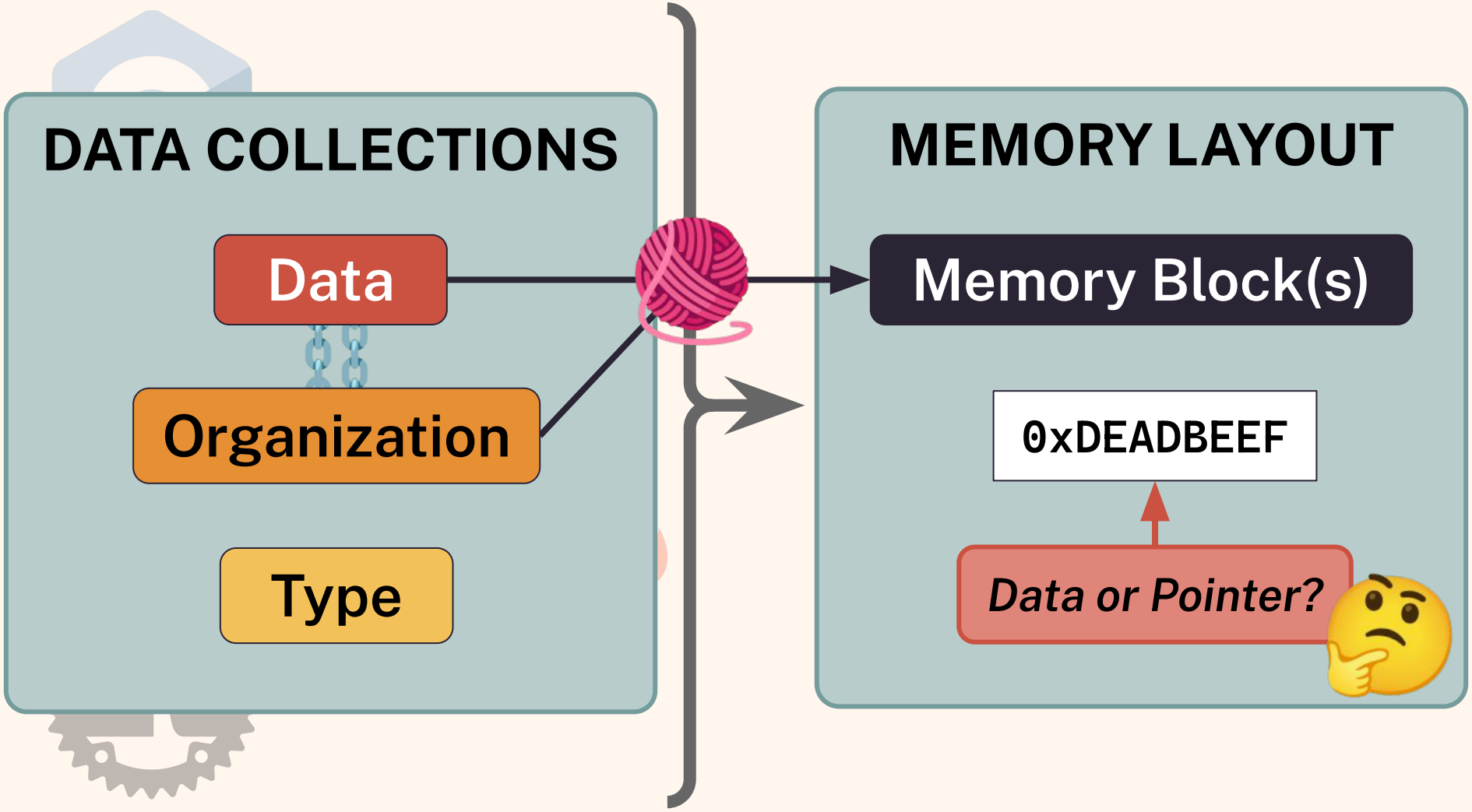
Shared Representation across Languages



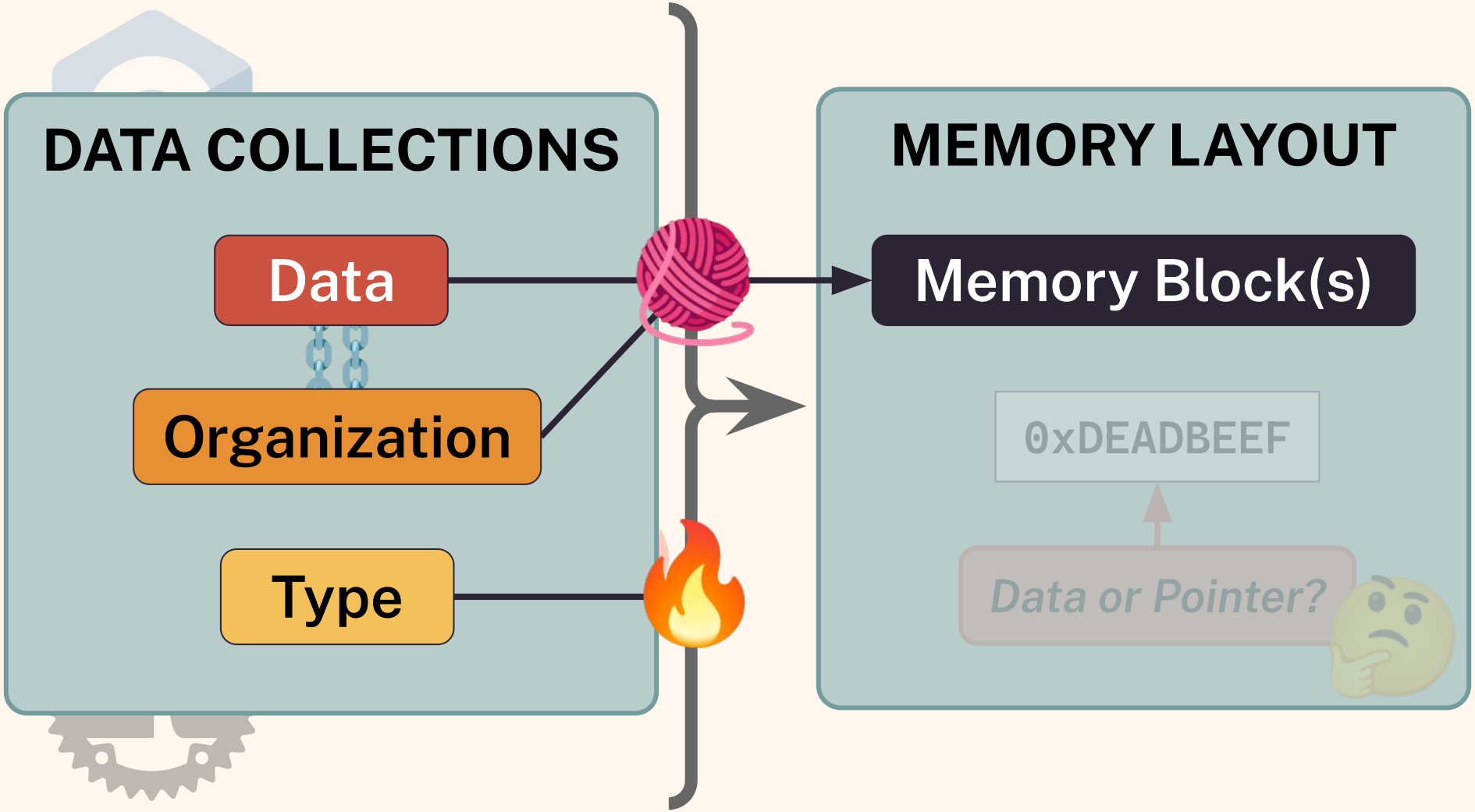
Shared Representation across Languages



Shared Representation across Languages



Shared Representation across Languages

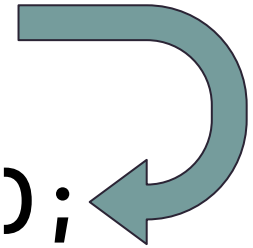


No Representation Makes Simple Things Hard!

```
std::unordered_map<int, int> &table;

table[1] = 10;
table[2] = 20;
print(table[1]);
```

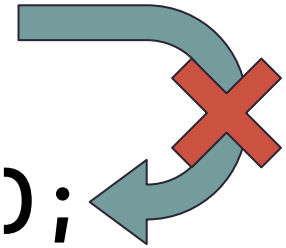
No Representation Makes Simple Things Hard!

```
std::unordered_map<int, int> &table;

table[1] = 10;
table[2] = 20;
print(table[1]);
```



No Representation Makes Simple Things Hard!

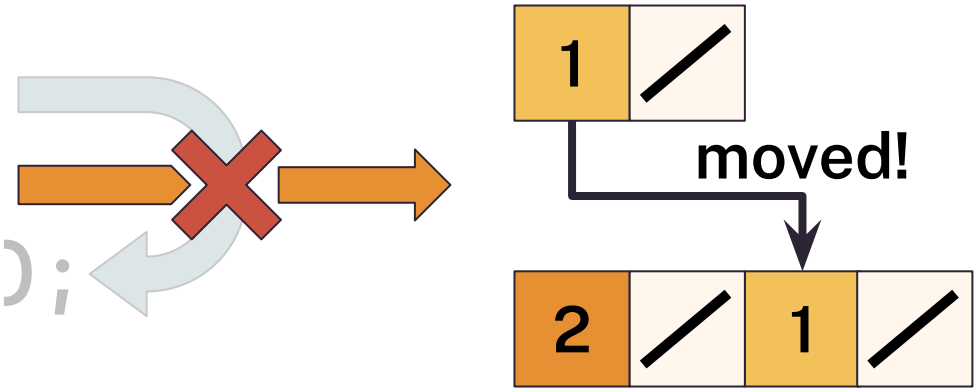
```
std::unordered_map<int, int> &table;

table[1] = 10;
table[2] = 20;
print(table[1]);
```



No production compiler can propagate 10!

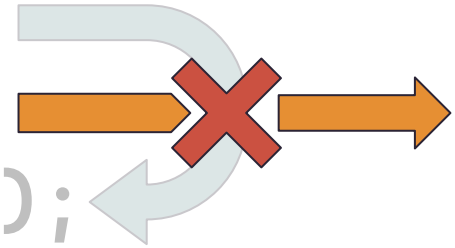
No Representation Makes Simple Things Hard!

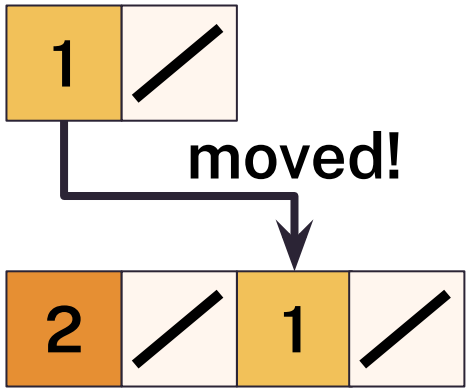
```
std::unordered_map<int, int> &table;

table[1] = 10;
table[2] = 20;
print(table[1]);
```



No Representation Makes Simple Things Hard!

```
std::unordered_map<int, int> &table;

table[1] = 10;
table[2] = 20;
print(table[1]);
```





Rehashing may move a ***logical element*** to a new ***memory location***

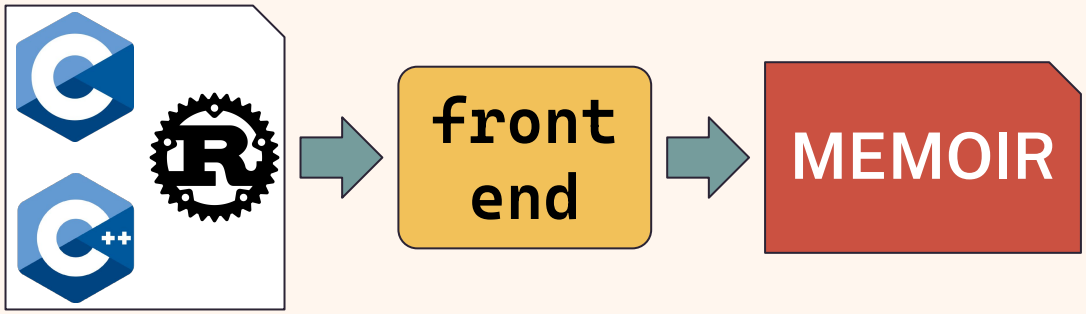
MEMOIR

LLVM IR Extension

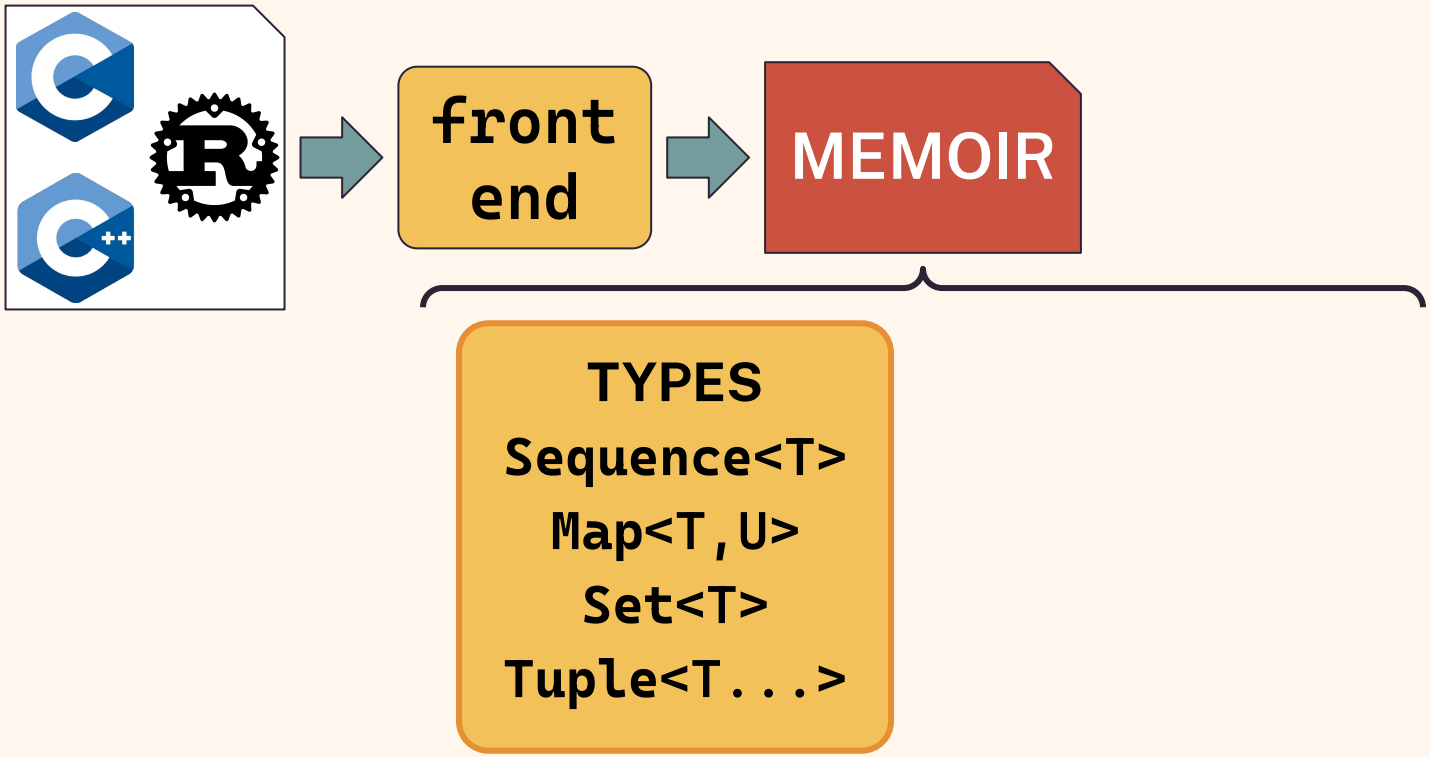
with *Data Collections*

in *SSA Form*

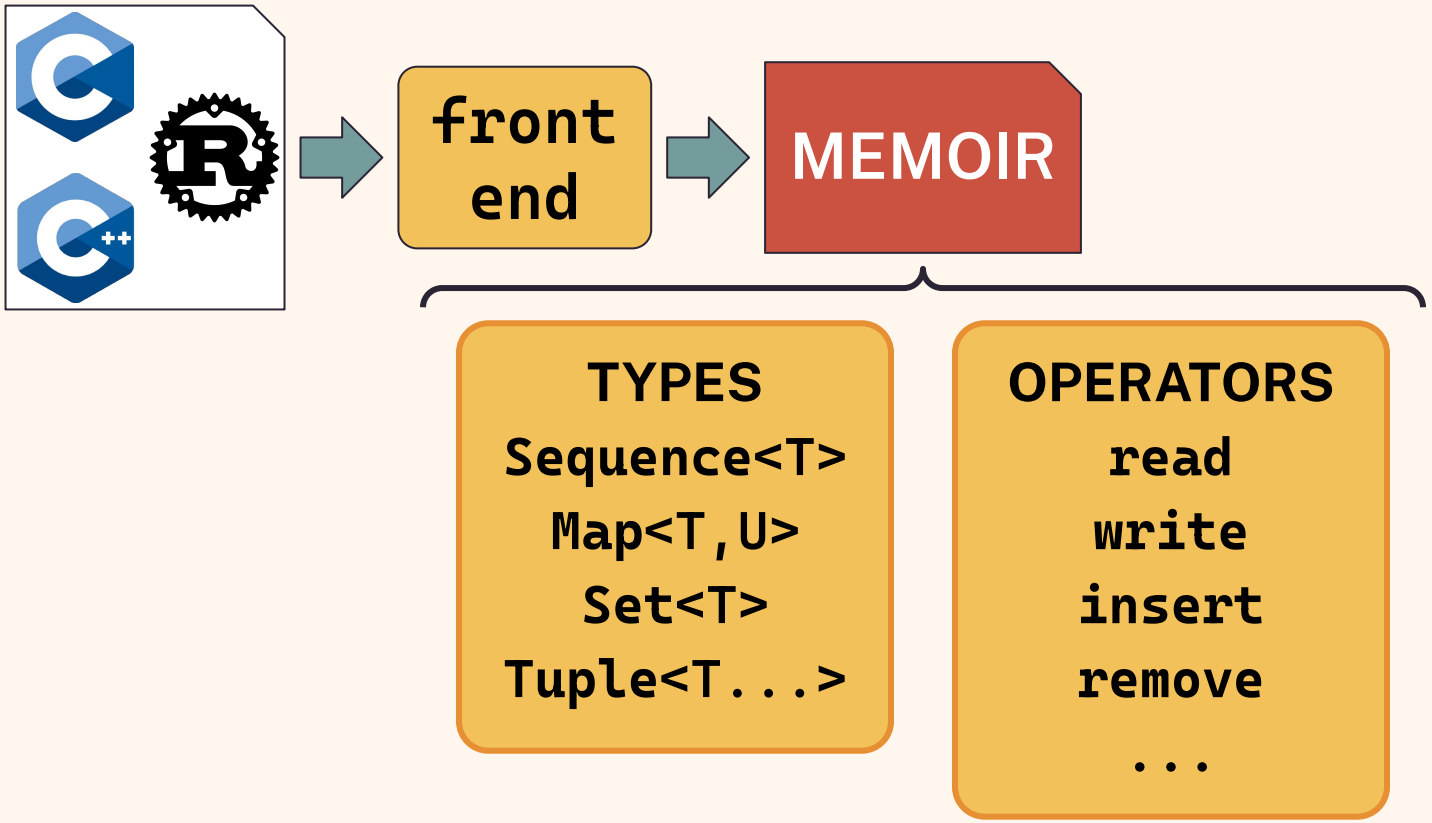
MEMOIR Design



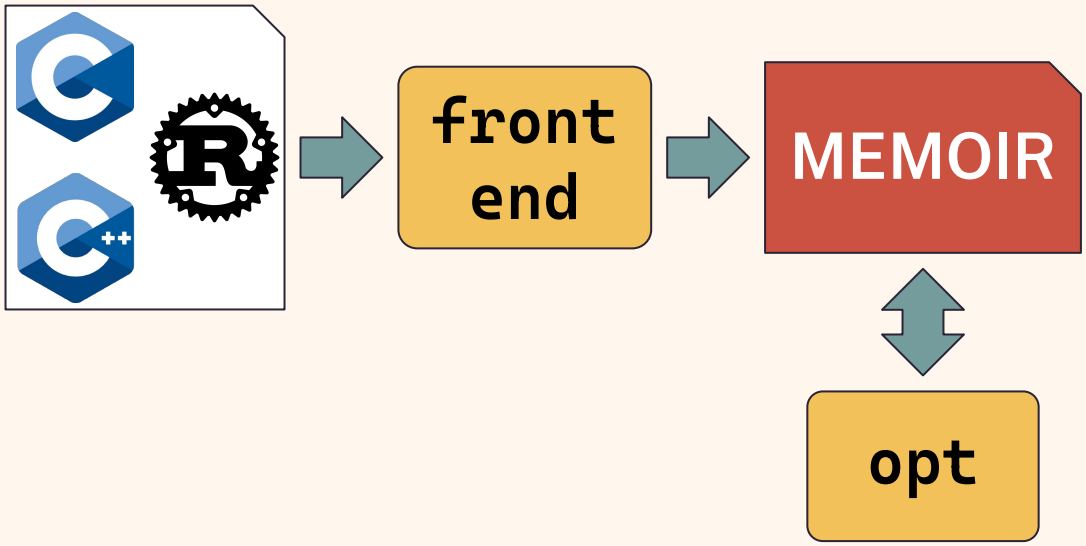
MEMOIR Design



MEMOIR Design



MEMOIR Design



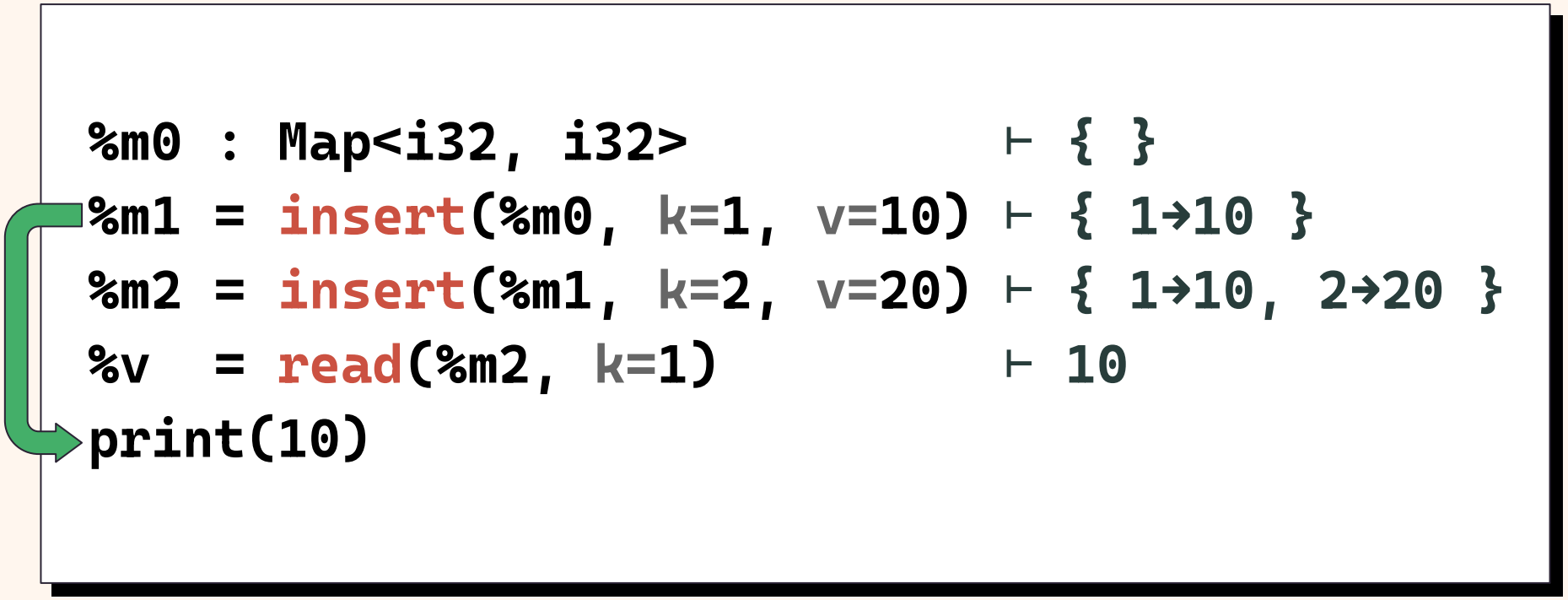
Improving Scalar Optimizations

```
%m0 : Map<i32, i32>
%m1 = insert(%m0, k=1, v=10)
%m2 = insert(%m1, k=2, v=20)
%v  = read(%m2, k=1)
print(%v)
```

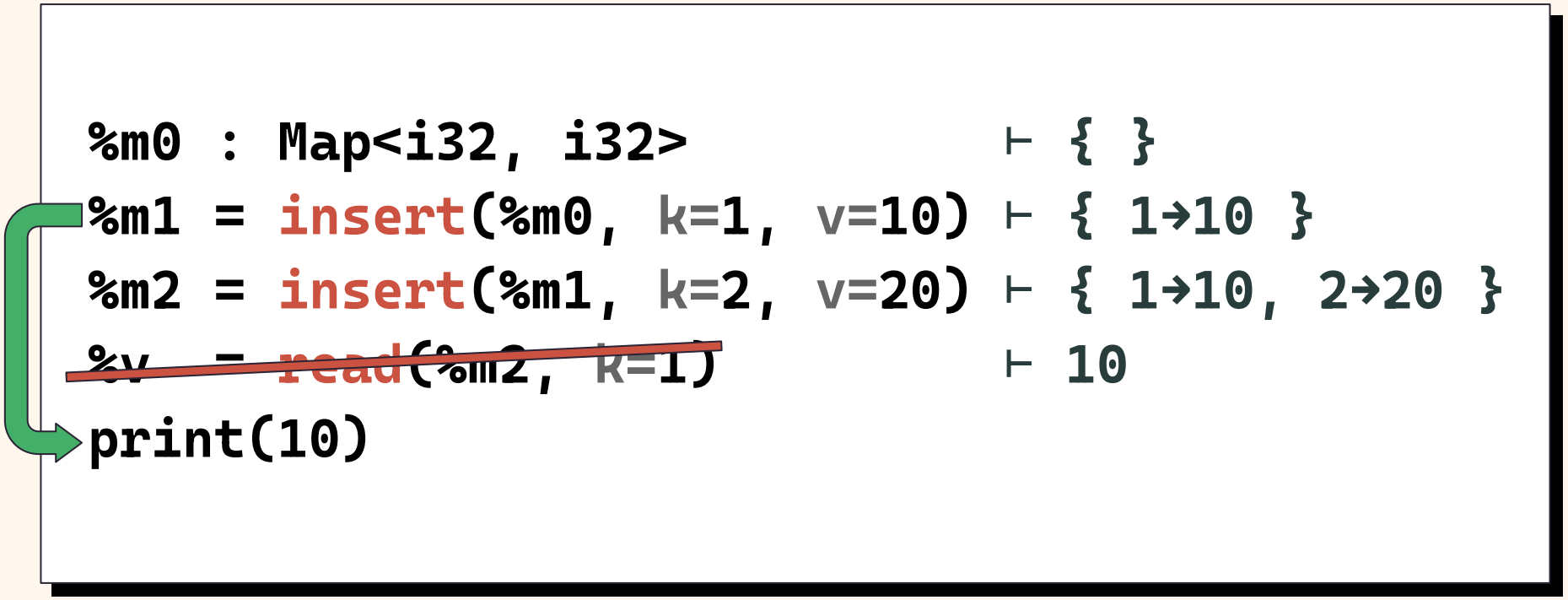
Improving Scalar Optimizations

```
%m0 : Map<i32, i32>                ⊢ { }
%m1 = insert(%m0, k=1, v=10)       ⊢ { 1→10 }
%m2 = insert(%m1, k=2, v=20)       ⊢ { 1→10, 2→20 }
%v  = read(%m2, k=1)               ⊢ 10
print(%v)
```

Improving Scalar Optimizations



Improving Scalar Optimizations



Data Transformations Made Simple

Data Transformations Made Simple

Eliminate Dead Fields

Data Transformations Made Simple

Eliminate Dead Fields

Migrate Cold Fields out of Hot Objects

Data Transformations Made Simple

Eliminate Dead Fields

Migrate Cold Fields out of Hot Objects

Specialize Memory Layout

MEMOIR Optimizations *Improve Performance!*

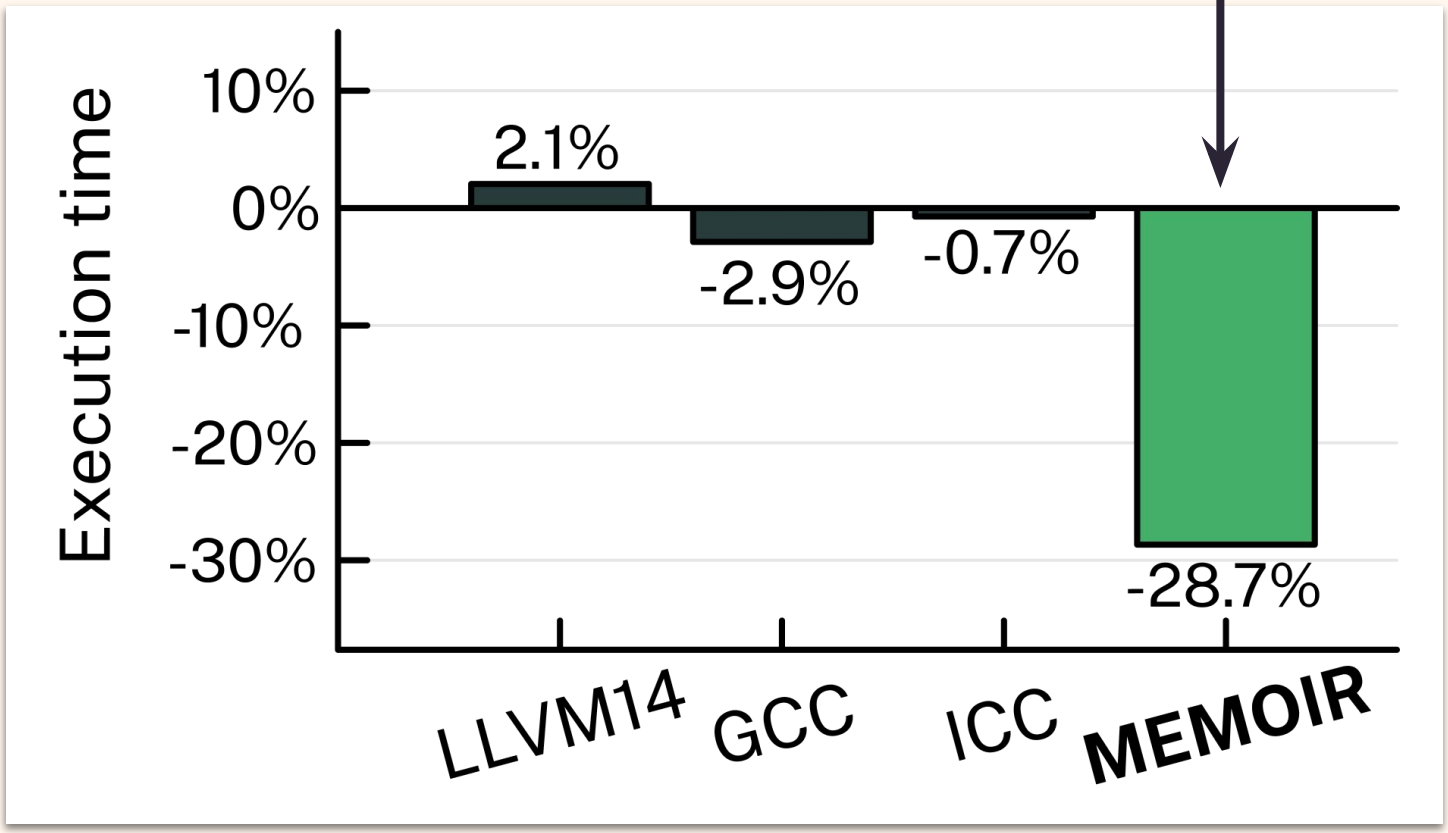


Figure: Execution time of mcf_s with refspeed input, 10 trials. Normalized to LLVM9.

MEMOIR Optimizations *Reduce Memory Usage!*

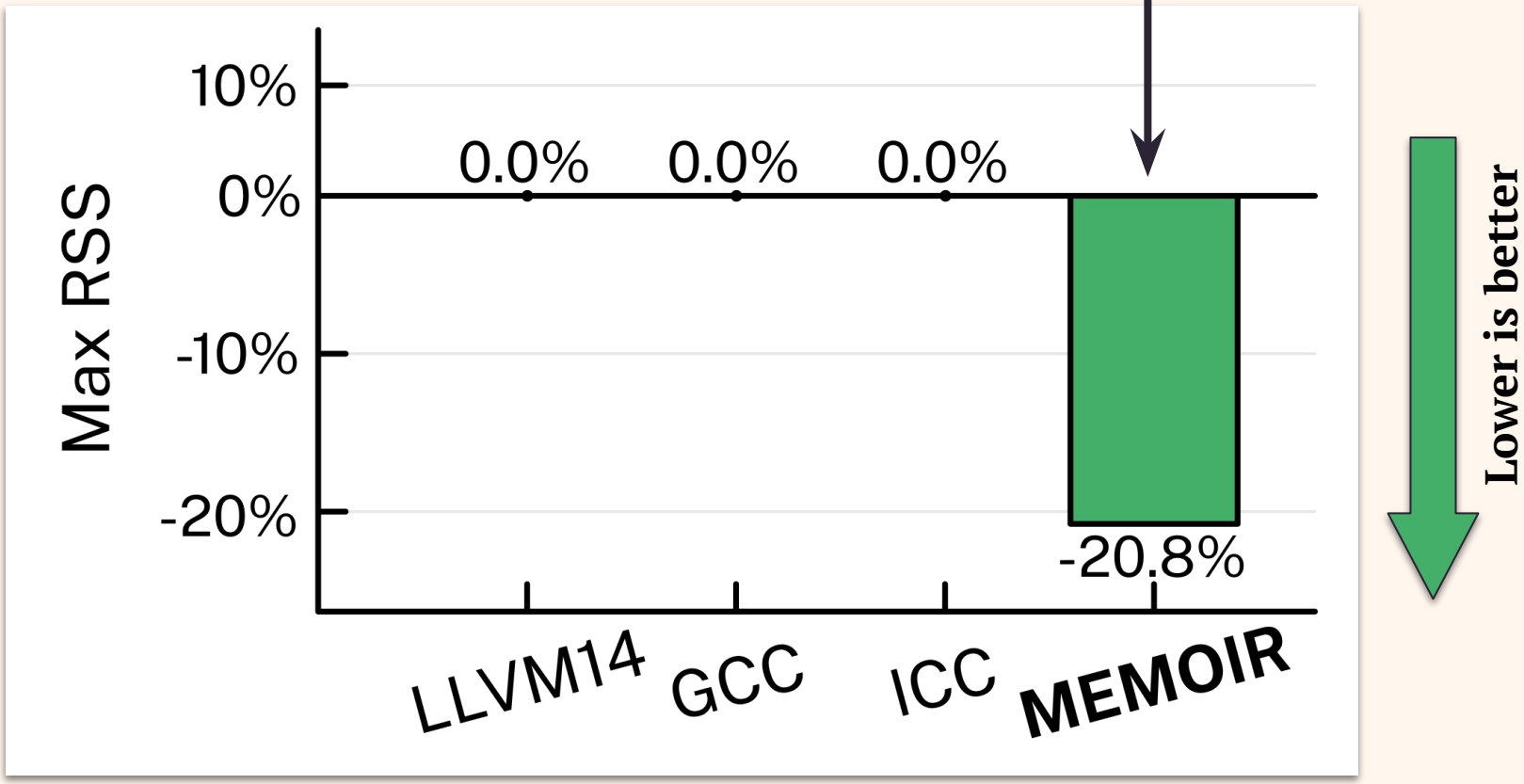
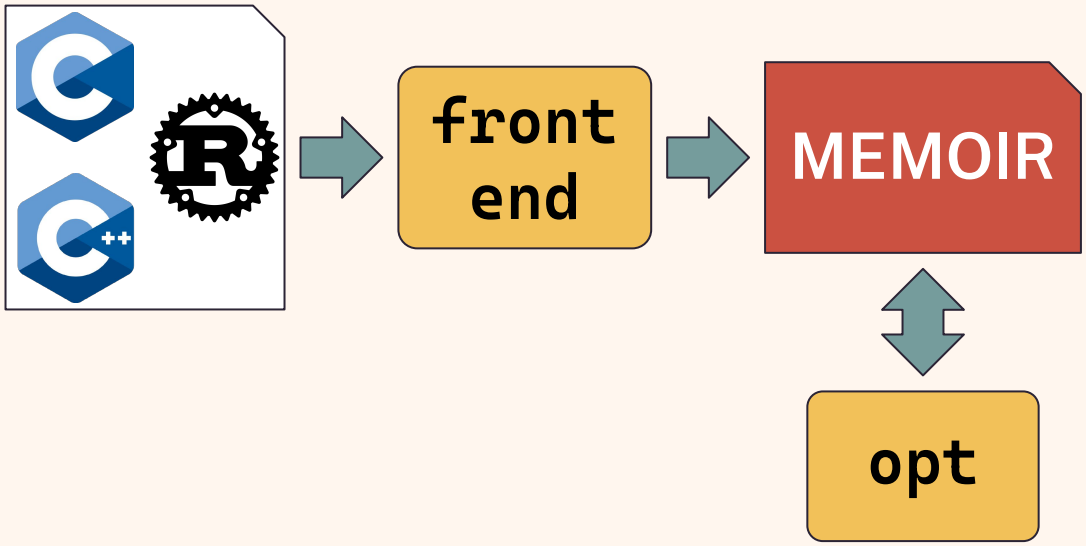
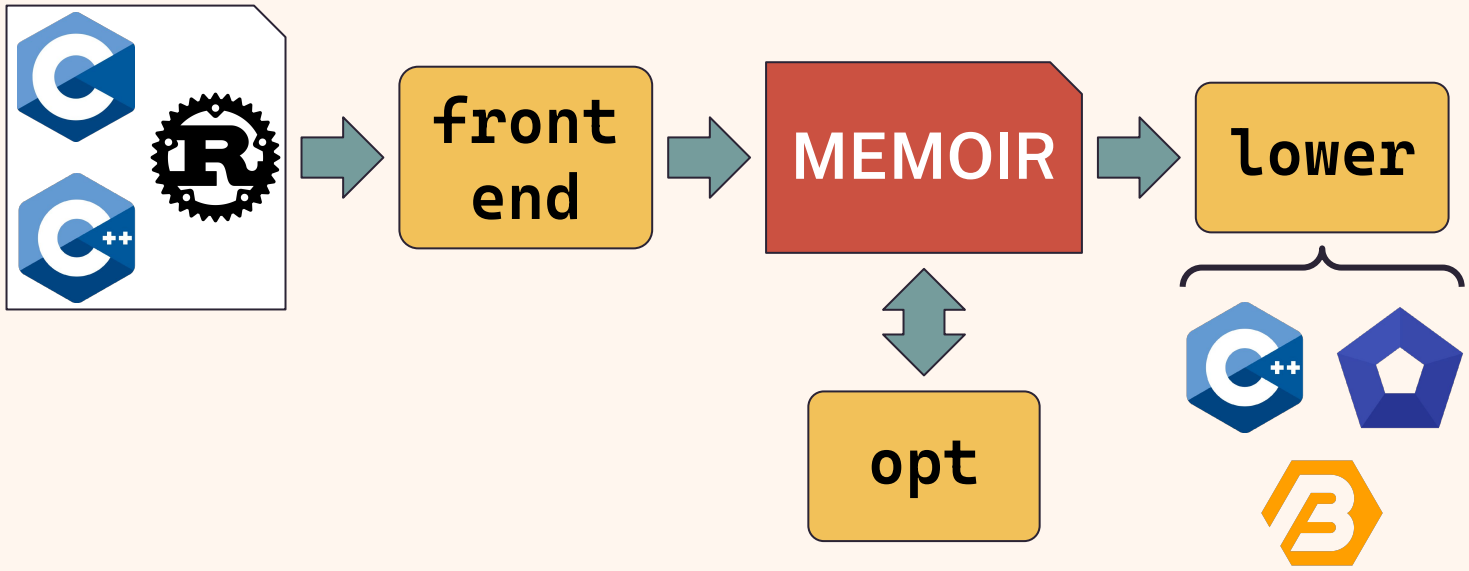


Figure: Max resident set size of mcf_s with refspeed input, 10 trials. Normalized to LLVM9.

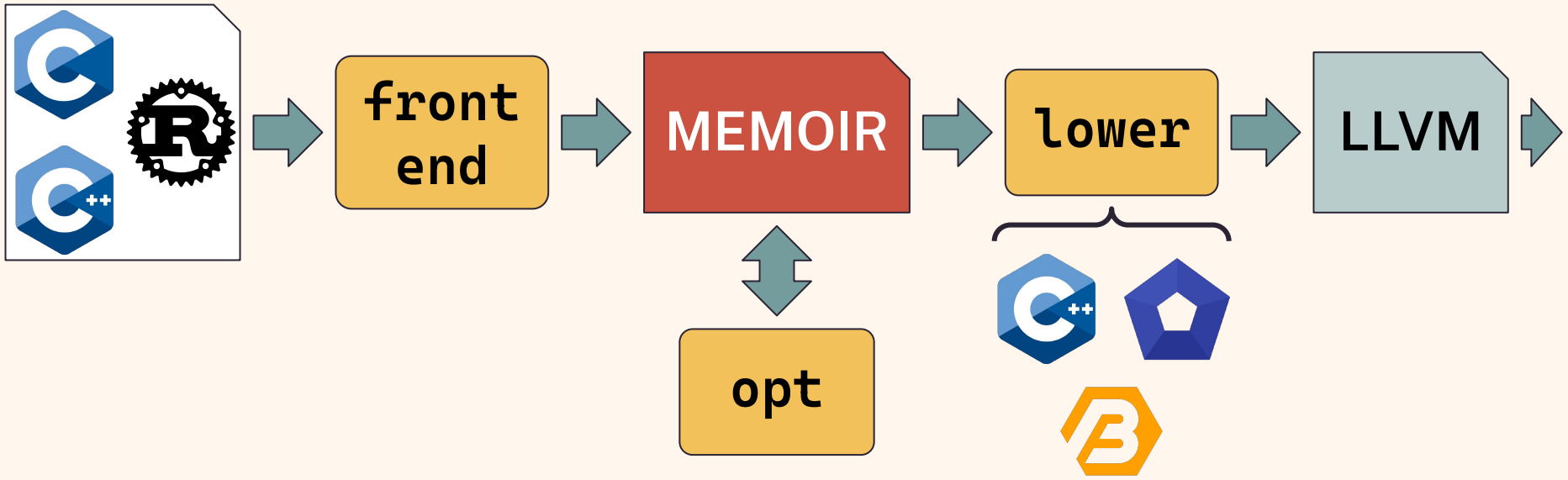
MEMOIR Design



MEMOIR Design

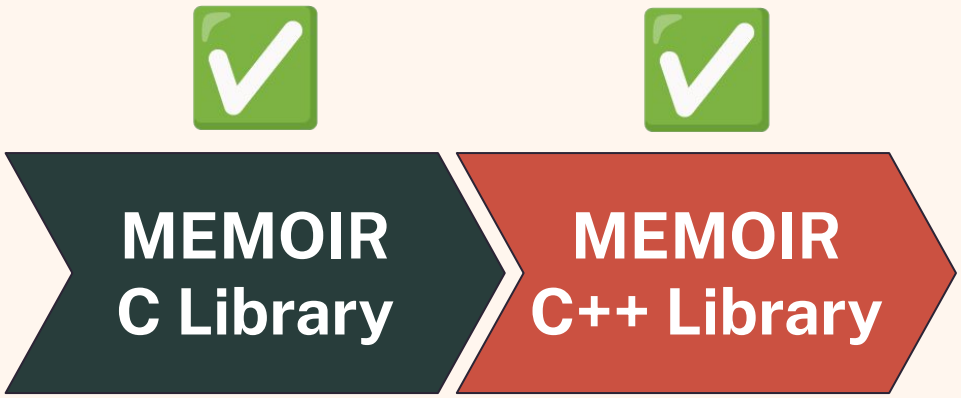


MEMOIR Design



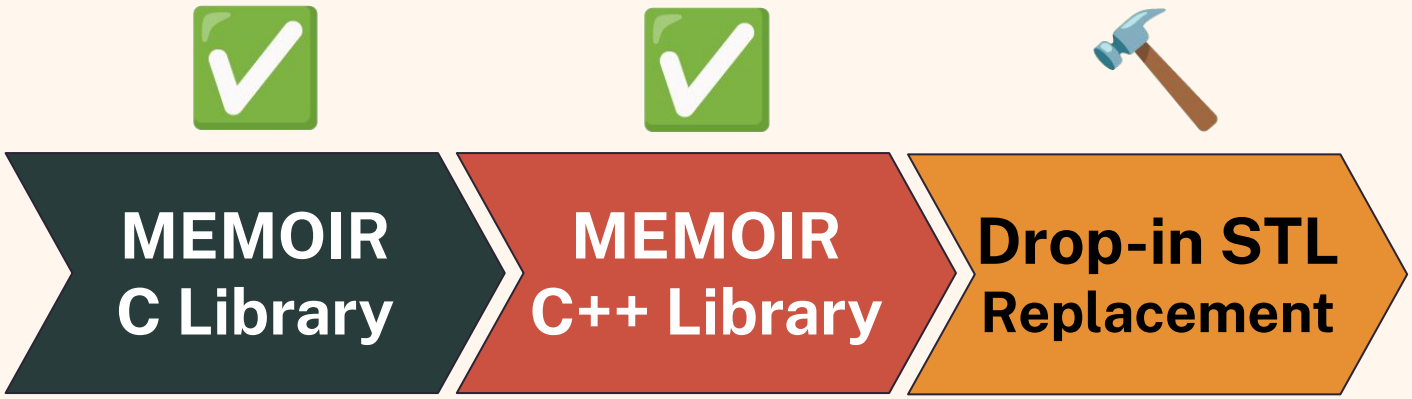
Next Steps

Towards Fully Transparent Raising



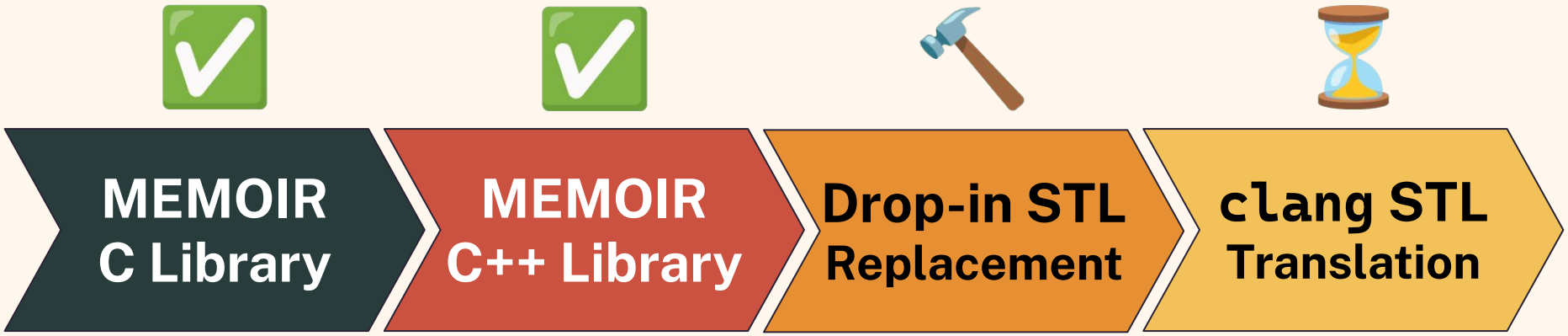
Next Steps

Towards Fully Transparent Raising

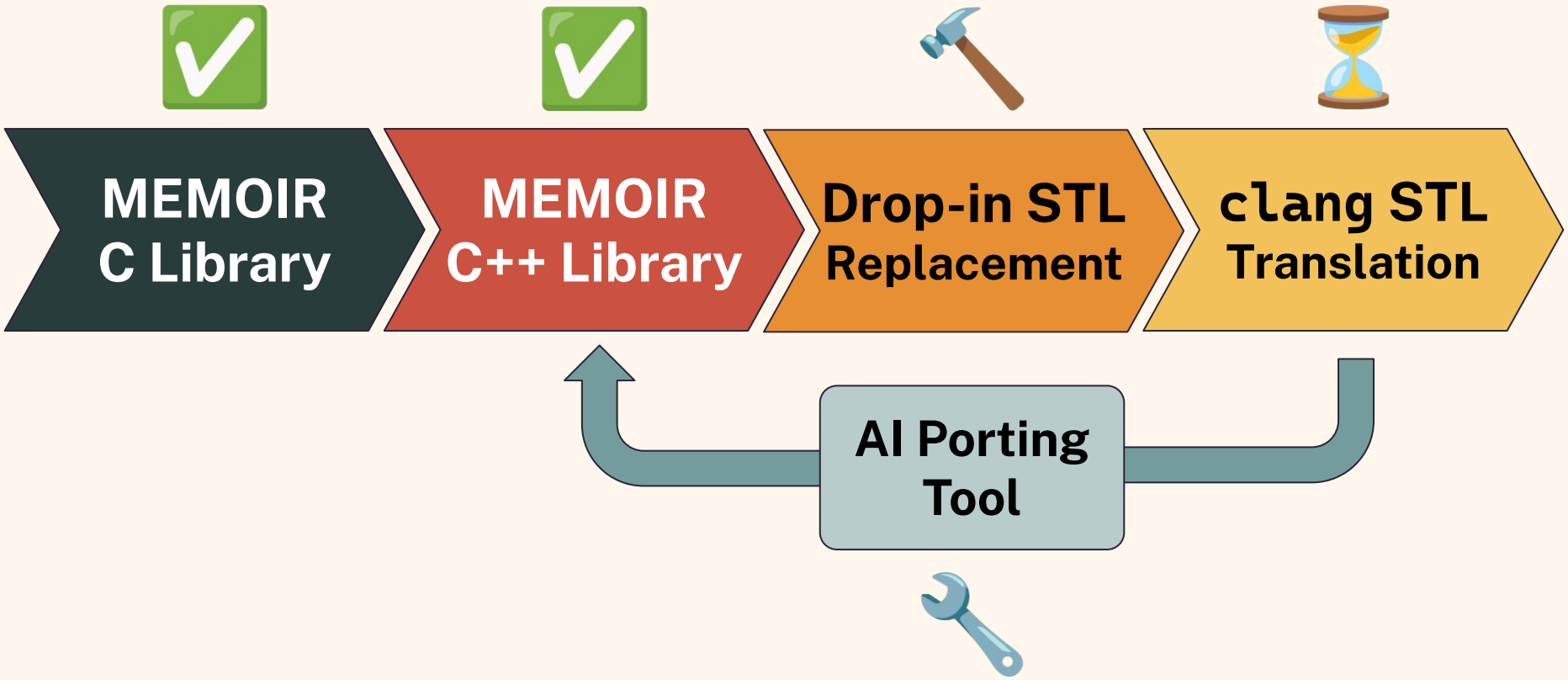


Next Steps

Towards Fully Transparent Raising



Towards Fully Transparent Raising



Learn more, get involved!

github.com/arcana-lab/memoir

CGO'24 Paper

MEMOIR Wiki

Learn more, get involved!

github.com/arcana-lab/memoir

CGO'24 Paper

MEMOIR Wiki

Poster

Tomorrow, 3:15-4:15

Learn more, get involved!

github.com/arcana-lab/memoir

CGO'24 Paper

MEMOIR Wiki

Poster

Tomorrow, 3:15-4:15

On the market

Tommy McMichen

mcmichen@u.northwestern.edu

www.mcmichen.cc

ARCANA
Rethinks Compiler Abstractions for New Applications

memoir