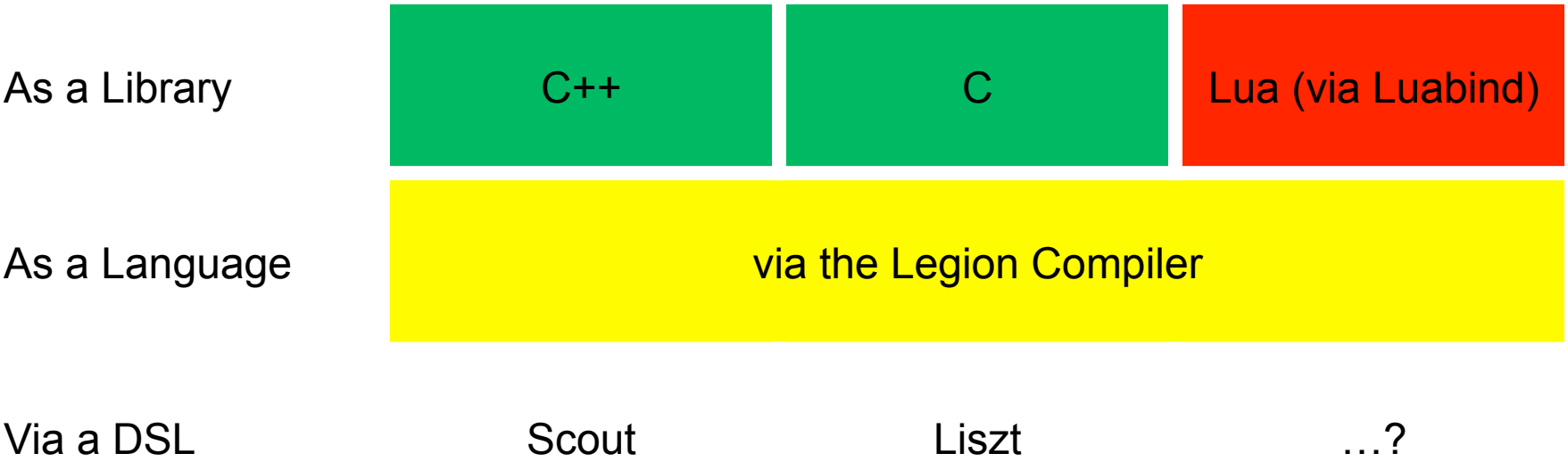



Legion Language and Compiler

Elliott Slaughter

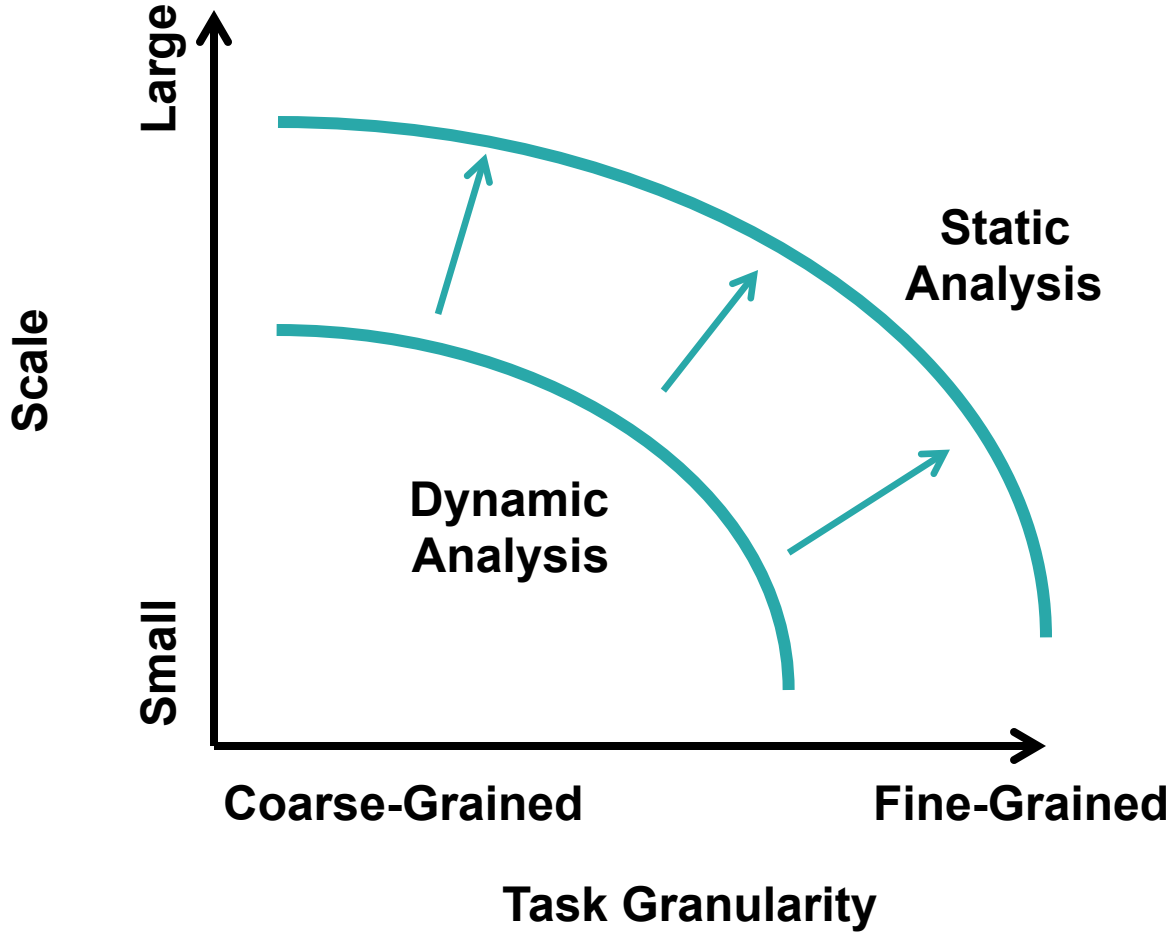
How Do You Program Legion?



Believe it or not…

- Legion is already a language
- With a real type system
  - Treichler, et al., OOPSLA 2013
- Legion, as implemented, is fully dynamic
  - Parallelism is discovered at runtime, by dynamic analysis
  - Currently we hide this cost with deferred execution
  - But if the runtime can't get ahead, performance suffers
- But what could we do with a Legion compiler?

Pushing the Performance Envelope with Static Analysis



Why Compilation?

- Expressiveness
  - The C++ API is verbose
- Safety
  - Type checker catches more errors at compile-time
- Optimization
  - Dynamic analysis limits task granularity and scale
  - Static optimizations allow us to push to envelope

Language Overview

Tasks

```
task fib(n : int)
  if n <= 1 then
    return 1
  end
  return fib(n-1) + fib(n-2)
end
```

Creating Regions

```
-- Field spaces are just structs
struct point {
  x : int,
  y : int,
  z : int,
}

-- Unstructured region containing 20 points
var r = region(point, 20)
```

Creating Partitions

```
var r = region(...)
var c : coloring = ...
var p = partition(disjoint, r, c)
for i = 0, n do
  var rn = p[i]
  ...
end
```

Iterating Regions

```
-- increment all elements in region
for x in r do
  @x += 1
end
```

Interoperability with C

```
cstdio = terralib.includec("stdio.h")

task main()
  cstdio.printf("hello world!\n")
end
```

Interoperability with Terra


```
-- This struct has an overloaded + operator
struct rbga {
  r : float, b : float, g : float, a : float
}
rgba.metamethods.__add = macro(
  function(x, y) ... end)

task sum(x : rbga, y : rgba)
  return x + y
end
```

Metaprogramming

```
function sum(type, zero)
  local task sum_(r : region(type))
    var total = zero
    for x in r do
      total += @x
    end
    return total
  end
  return sum_
end

-- later...
var r = region(int, 20)
[sum(int, 0)](r)
```

Safety: Permissions


```
task lookup(r : region(int), x : ptr(int, r)),
    reads(r)
  @x = 5 -- compile-time error
end
```

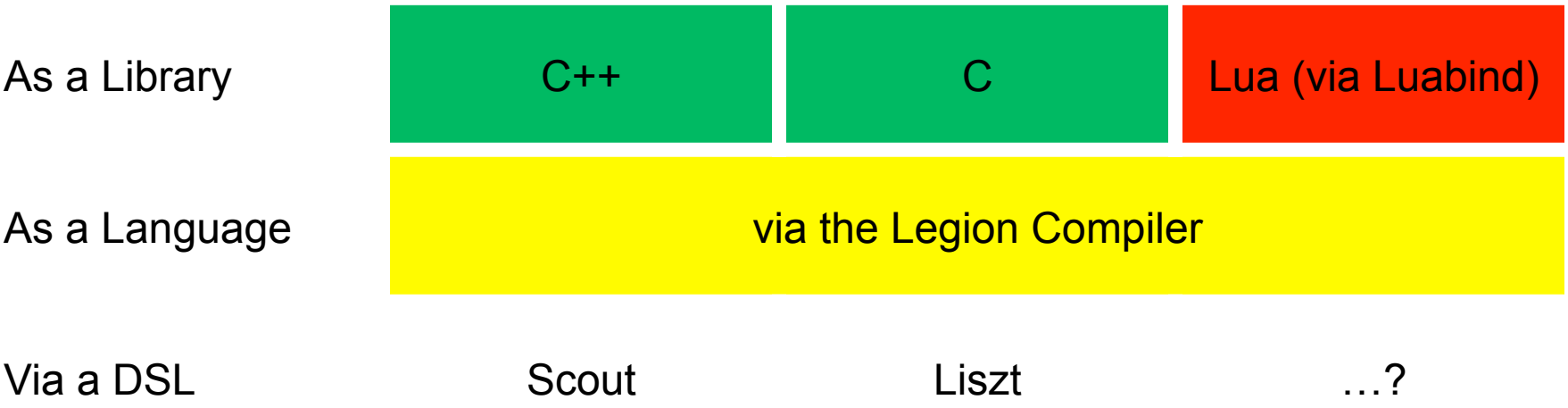
Safety: Pointers


```
task sparse_sum(r : region(int), s : region(ptr(int, r))),
    reads(r, s)
  var s = 0
  for x in s do
    s += @x -- OK: compiler knows x points to r
  end
  return s
end
```

Status

- Compiler is close to feature-complete
- But not yet optimized!
- Please tell me about your app so I can use it as a benchmark

How Do You Program Legion?



- **Mere Mortals: DSL of choice**
- **Advanced Application Developers: C++**
  - **Or if you're adventurous, try the compiler!**
- **Language and Library Authors: C, or Terra + C**

Optimization: Distribution