Check-In

Review: Heap Management

What is the purpose of LLVM's opt tool?

Announcements

Administrivia

- Review Session in Learned 1136



Partial Evaluation

Today's Lecture

Partial Evaluation

Partial Evaluation

- What it is
- How to do it
- The Futamura projections



Advanced Topics

Partial Evaluation

Partial Evaluation - Background

Disclaimer: the fun-sized introduction to a complex topic



There are whole books on this technique!

(https://compilers.cool/materials/JonesPartialEvaluation.pdf)

Compiler Philosophy

About Partial Evaluation - Background

More *static* work means less *dynamic* work

- Optimize programs to run better
- Flag (potential) bugs before they bite



"Fortune favors the prepared"
- Louis Pastuer

Compiler Philosophy - Consequence

About Partial Evaluation - Background

We can only optimize what we can prepare for

```
int a = atoi(argv[1]);
int y = atoi(argv[2]);
int x;
if (y < 3){
   x = (4 * (7 * (2 + (3 * a))));
} else {
   x = 1;
}
```

A tasty target for optimization…



blocked by a pesky dynamic value!



Partial Evaluation: Concept

Partial Evaluation

We often use the same value for *some* of the arguments to the program

- What if we could take those values for granted?
- *Specialize* programs for "guaranteed" inputs

"Drew"

```
bool cool(bool age, char * name){
  size_t len = strlen(name);
                          4
  if (isPrime(len)){
    return false;
  } else {
    return age < 30;
  }
}
```

How I run this program:

cool(27, "Drew") -> true
cool(28, "Drew") -> true
cool(29, "Drew") -> true
cool(30, "Drew") -> false
cool(31, "Drew") -> false
cool(32, "Drew") -> false

Partial Evaluation: "Specialization"

Create special versions of a program

- Less general than the original program
- More efficient on what it does do

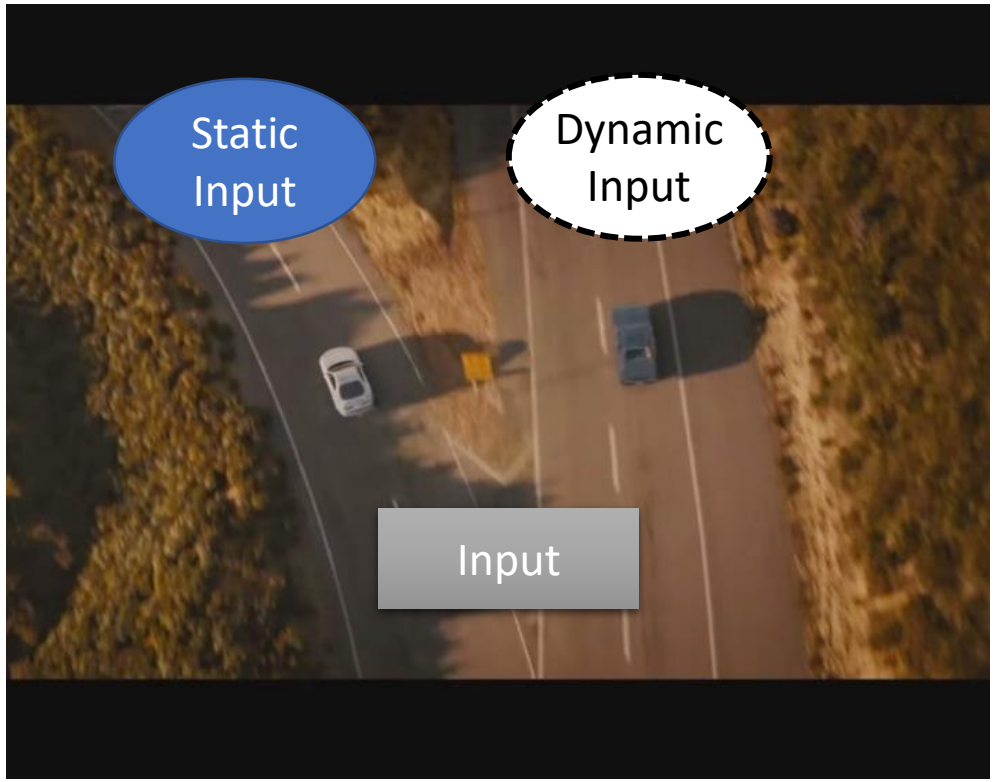


Dividing Input

Partial Evaluation: Concept

Split input into two groups

Some inputs constant, the rest dynamic



“This is Kinda Like Currying!”

Partial Evaluation: Concept

Yep, it is!



Haskell Curry, after whom Currying and the Haskell language are named

"This is Kinda Like Currying!"

Partial Evaluation: Concept

Yep, it is!

Some differences:

- PE not constrained to function level
- PE can perform arbitrary combinations of arguments

Uncurried function

$f : (x, y) \rightarrow z$

Curried function

$f : x \rightarrow (y \rightarrow z)$

```
int minus(int a, int b){
  a - b;
}
```

minus a as 5 →

```
int minus(int b){
  5 - b;
}
```

```
int minus(int a, int b){
  a - b
}
```

minus b as 5 →

```
int minus(int a){
  a - 5;
}
```

Today's Lecture

Partial Evaluation

Partial Evaluation

- What it is
- How to do it
- The Futamura projections



Simplistic Implementation Intuition

Partial Evaluation - Technique

Analyze every program path

- If dynamic data can influence it, don't alter the code
- If multiple values can touch it, account for all possibilities
- If value is static, replace with result (like constant folding)!

Dynamic

Static "Drew"

```
bool cool(bool age, char * name)
{
  size_t len = strlen("Drew");
  if (isPrime(len)){
    return false;
  } else {
    return age < 30;
  }
}
```

```
bool cool_nameIsDrew(bool age)
{
  size_t len = 4;
  if (isPrime(4)){
  } else {
    return age < 30;
  }
}
```

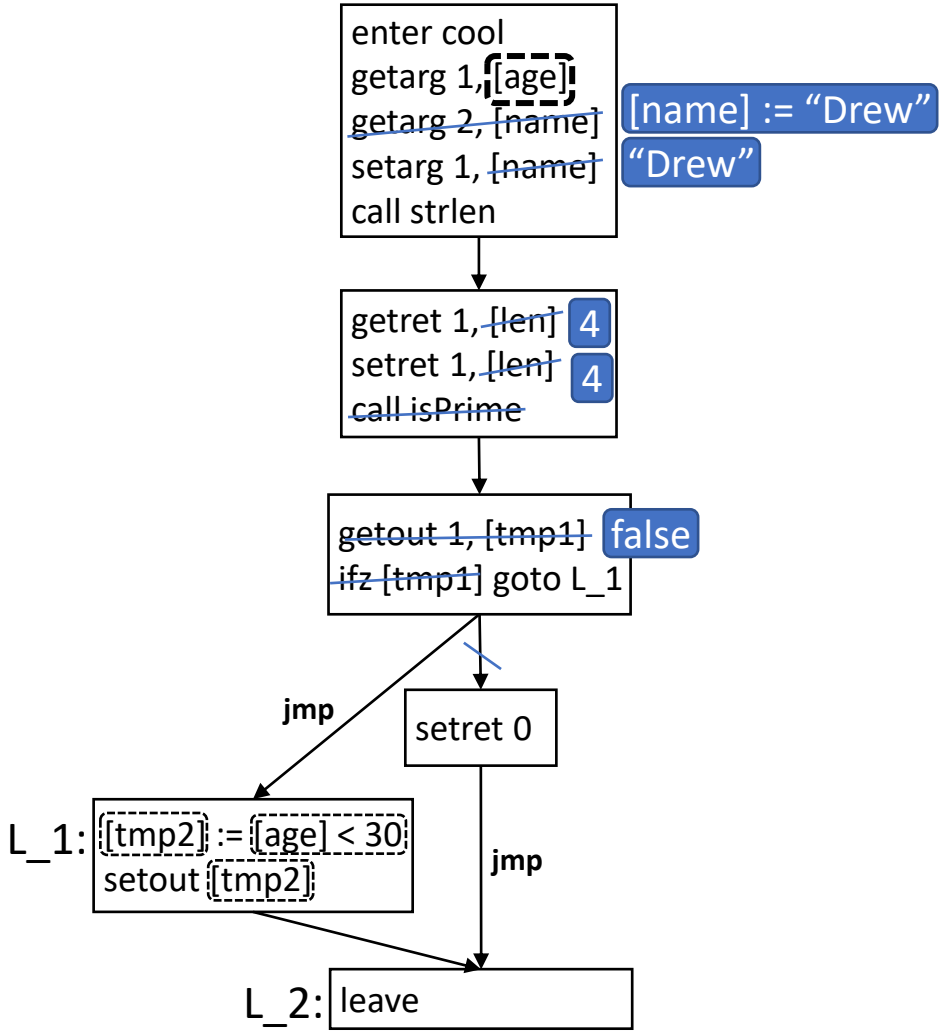
Implementation – Dataflow Approach

Partial Evaluation - Technique

Propagate dynamic values

- Leave dynamic-dependent code alone
- Evaluate purely-static code paths

```
bool cool(bool age, char * name)
{
  size_t len = strlen(name);
  if (isPrime(len)){
    return false;
  } else {
    return age < 30;
  }
}
```

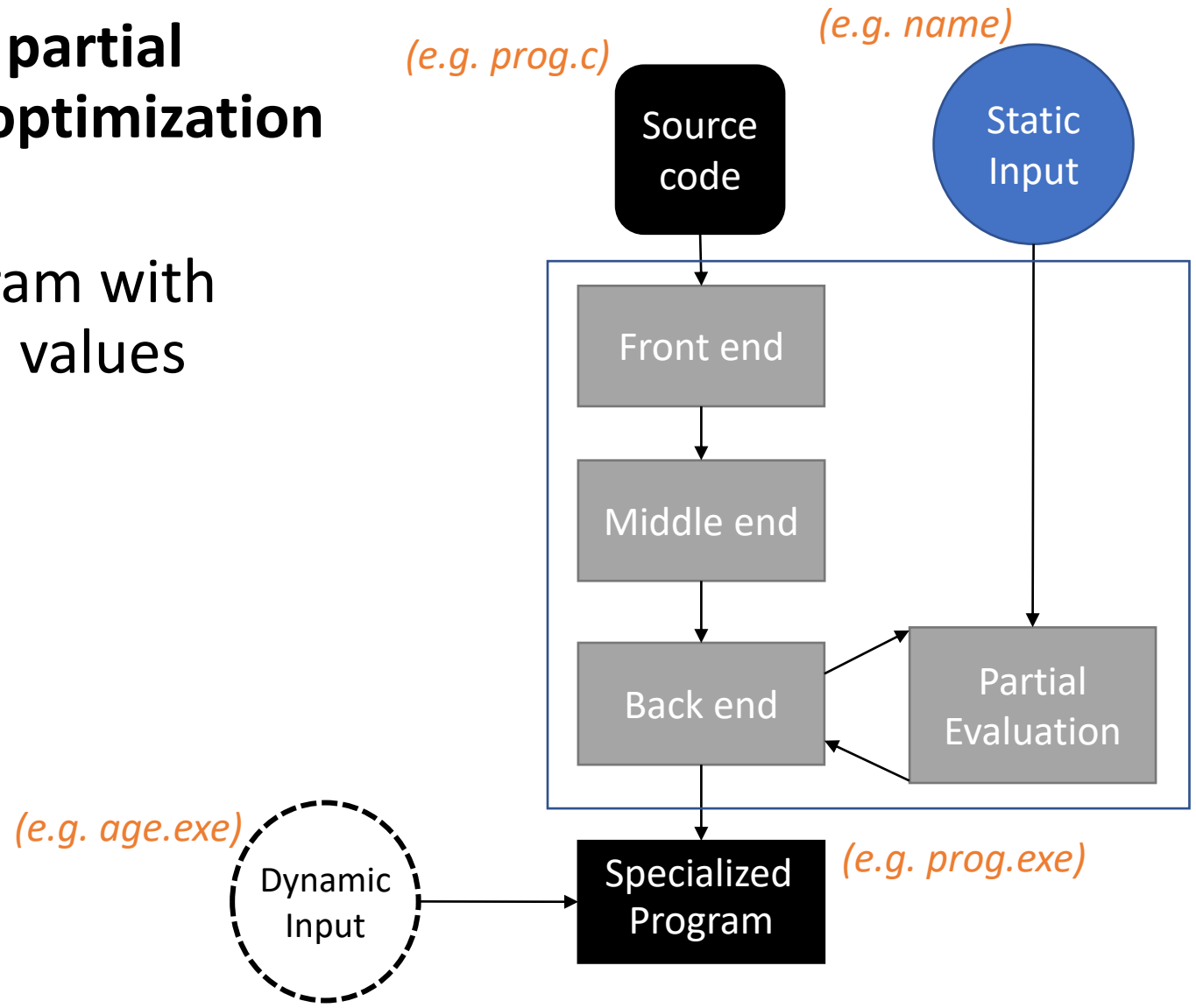


Partial Evaluation as Compiler Pass

Partial Evaluation - Technique

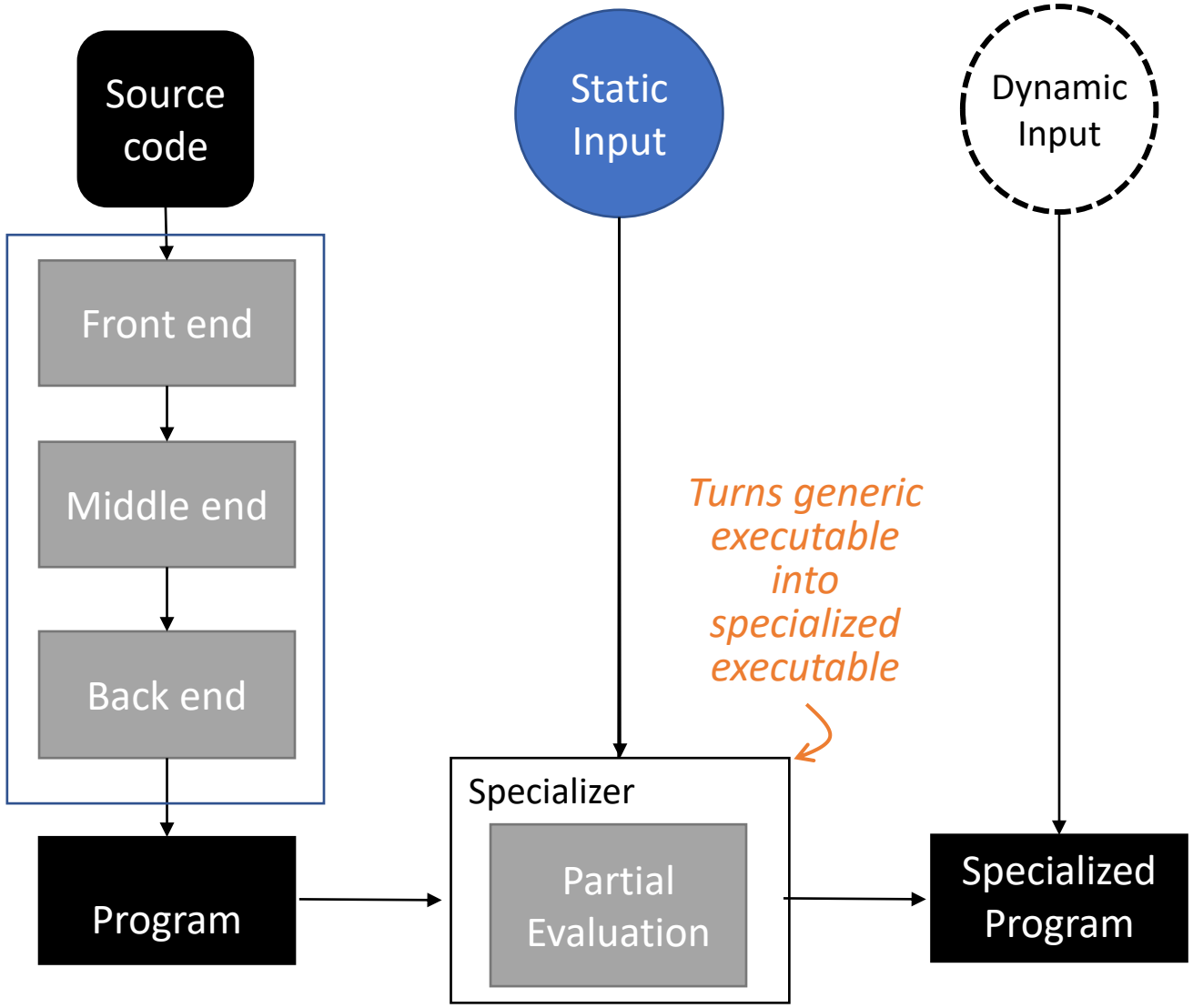
Could implement partial evaluation as an optimization module

- Recompile program with your guaranteed values



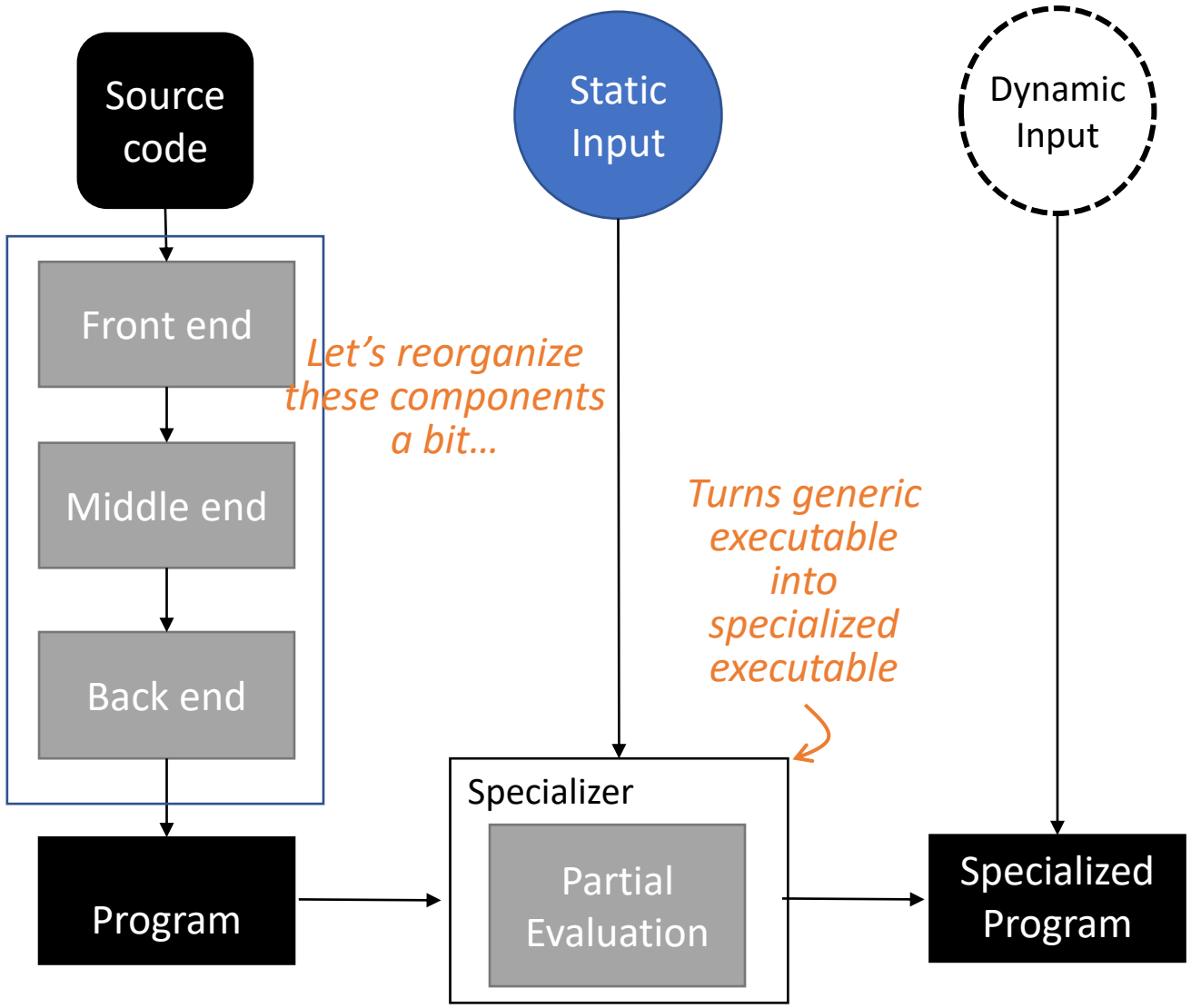
Partial Evaluation as Compiler Pass

Partial Evaluation - Technique



The Specializer

Partial Evaluation - Technique



The Specializer

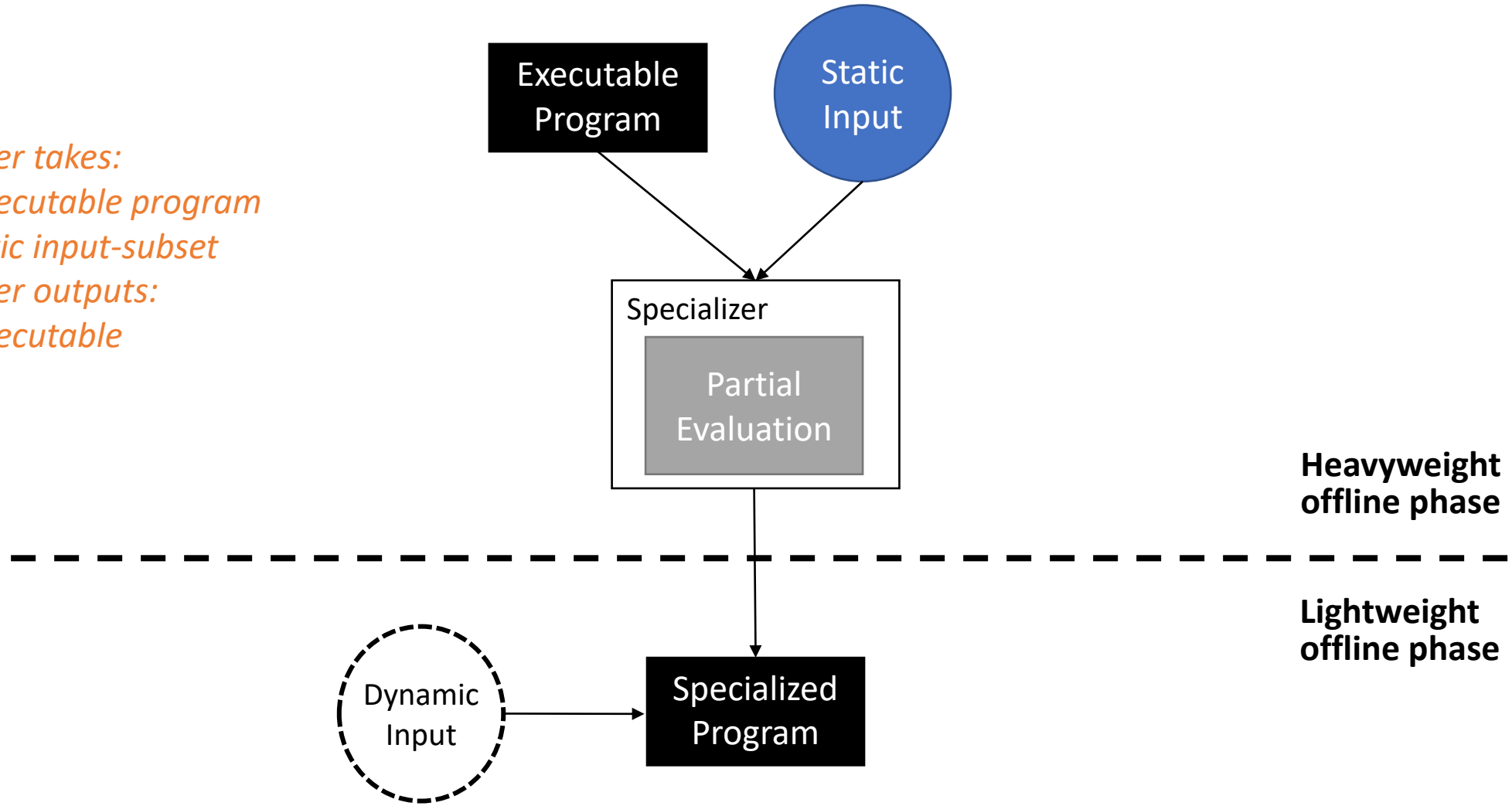
Partial Evaluation - Technique

Specializer takes:

- *An executable program*
- *A static input-subset*

Specializer outputs:

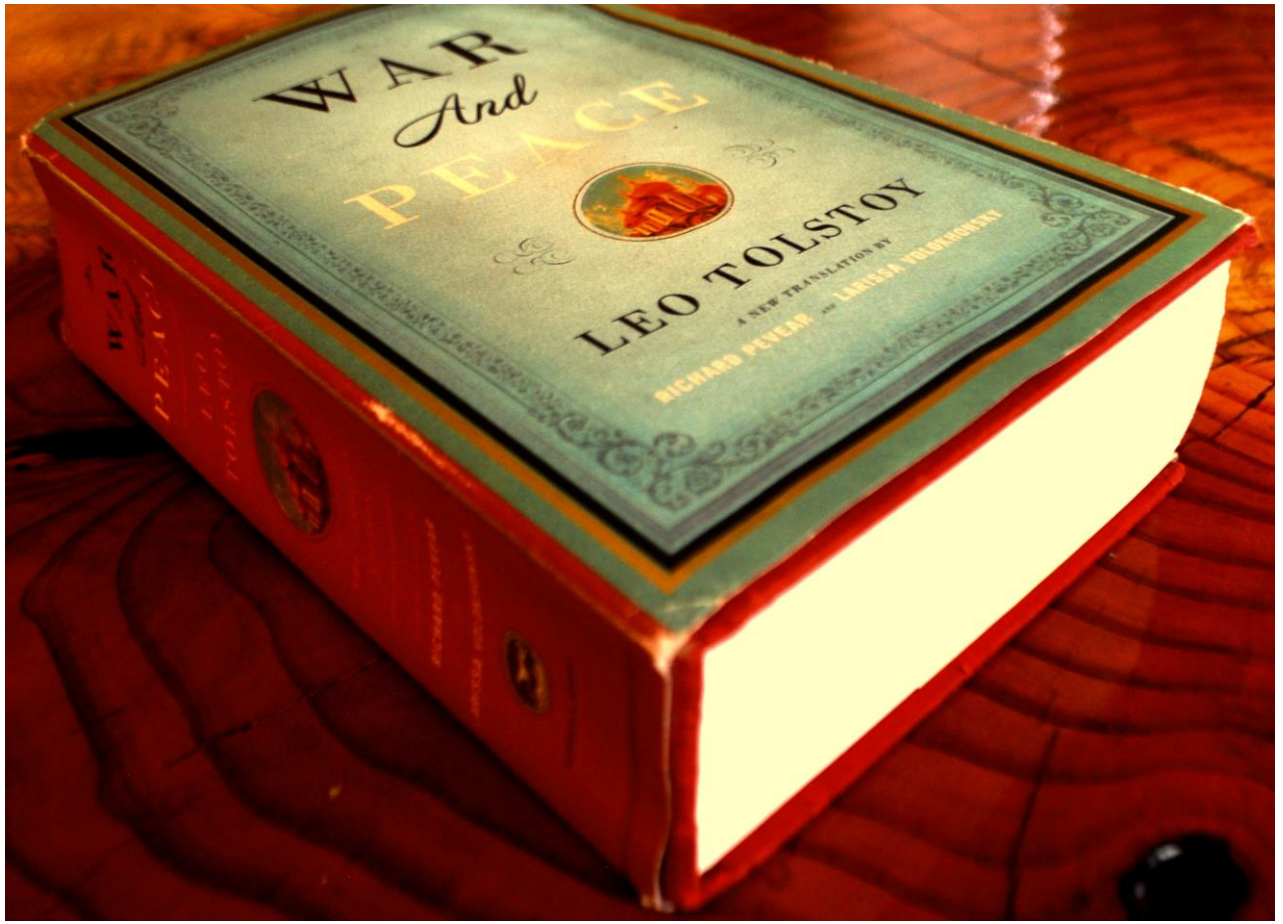
- *An executable*



The Specializer: Example

Partial Evaluation - Technique

Assume we frequently need to know text statistics of War and Peace



The Specializer: Example

Partial Evaluation - Technique

Assume we frequently need to know text statistics of War and Peace

- How many lines of text?
- How many words?
- How many characters?

Assume we run these commands a lot

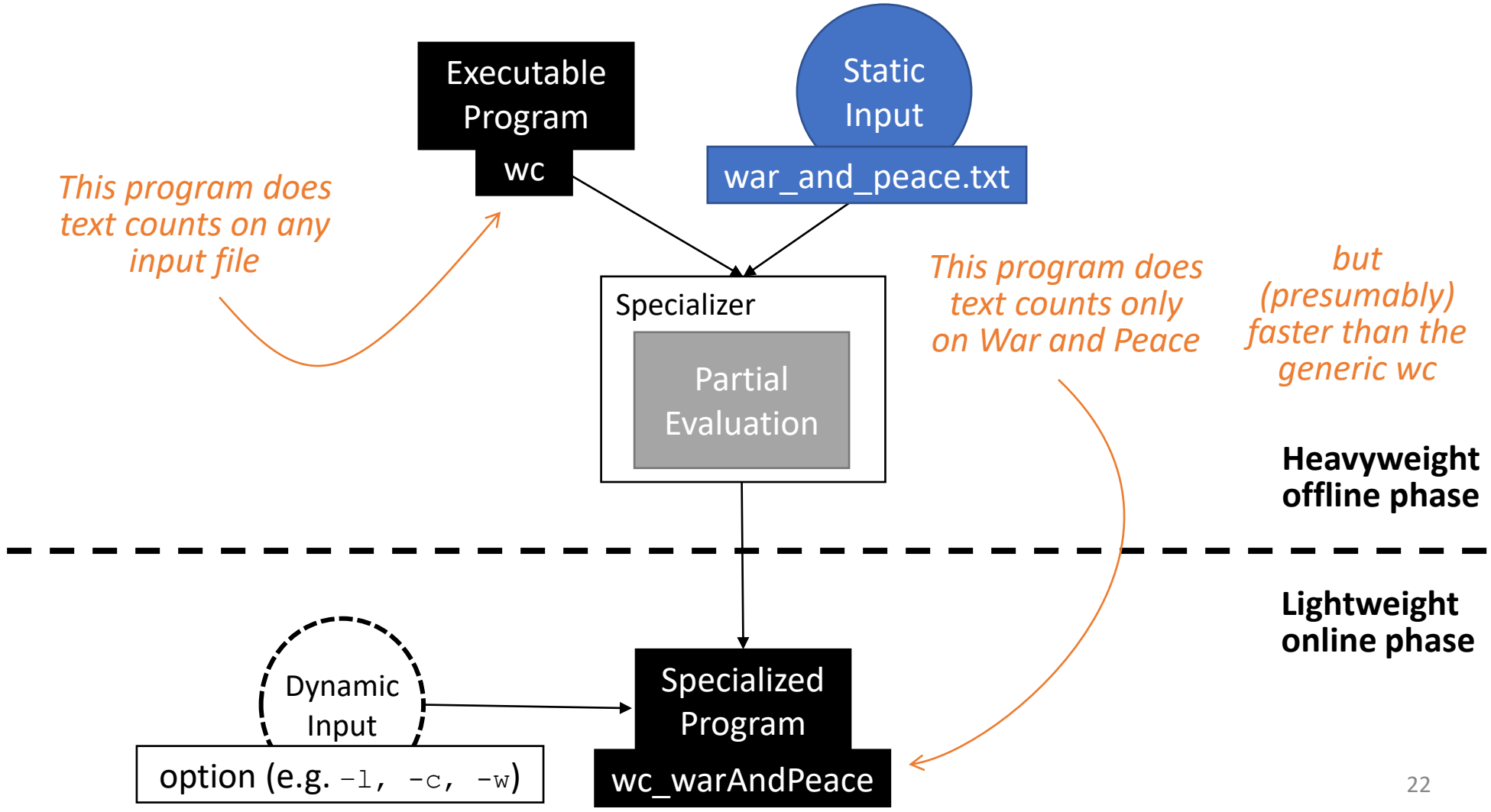
options change *text is static*

```
wc -l war_and_peace.txt
wc -w war_and_peace.txt
wc -c war_and_peace.txt
wc -w war_and_peace.txt
wc -l war_and_peace.txt
wc -w war_and_peace.txt
```

The Specializer: Example

Partial Evaluation - Technique

Assume we frequently need to know text statistics of War and Peace



Specializer Optimization Targets

Partial Evaluation

- Pattern recognition
- Ray tracing of solid models
- Neural network training
- Database queries
- Spreadsheet computations
- Scientific computing
- Discrete hardware simulation

Other Uses of Specialization

Partial Evaluation – Futamura Projections

A (perhaps obvious) observation:

- Some programs take other programs as input
- What if we used specialization as part of the program transformation process?

This observation leads to some startling results



Today's Lecture

Partial Evaluation

Partial Evaluation

- What it is
- How to do it
- The Futamura projections



Advanced Topics

The Futamura Projections

Partial Evaluation – Futamura Projections



Baseline specialization
(not a Futamura Projection)

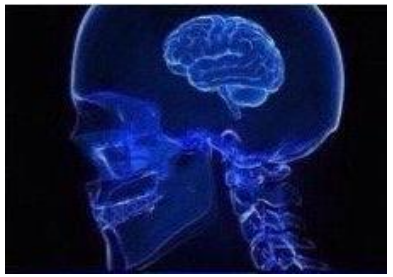
1st Futamura Projection

2nd Futamura Projection

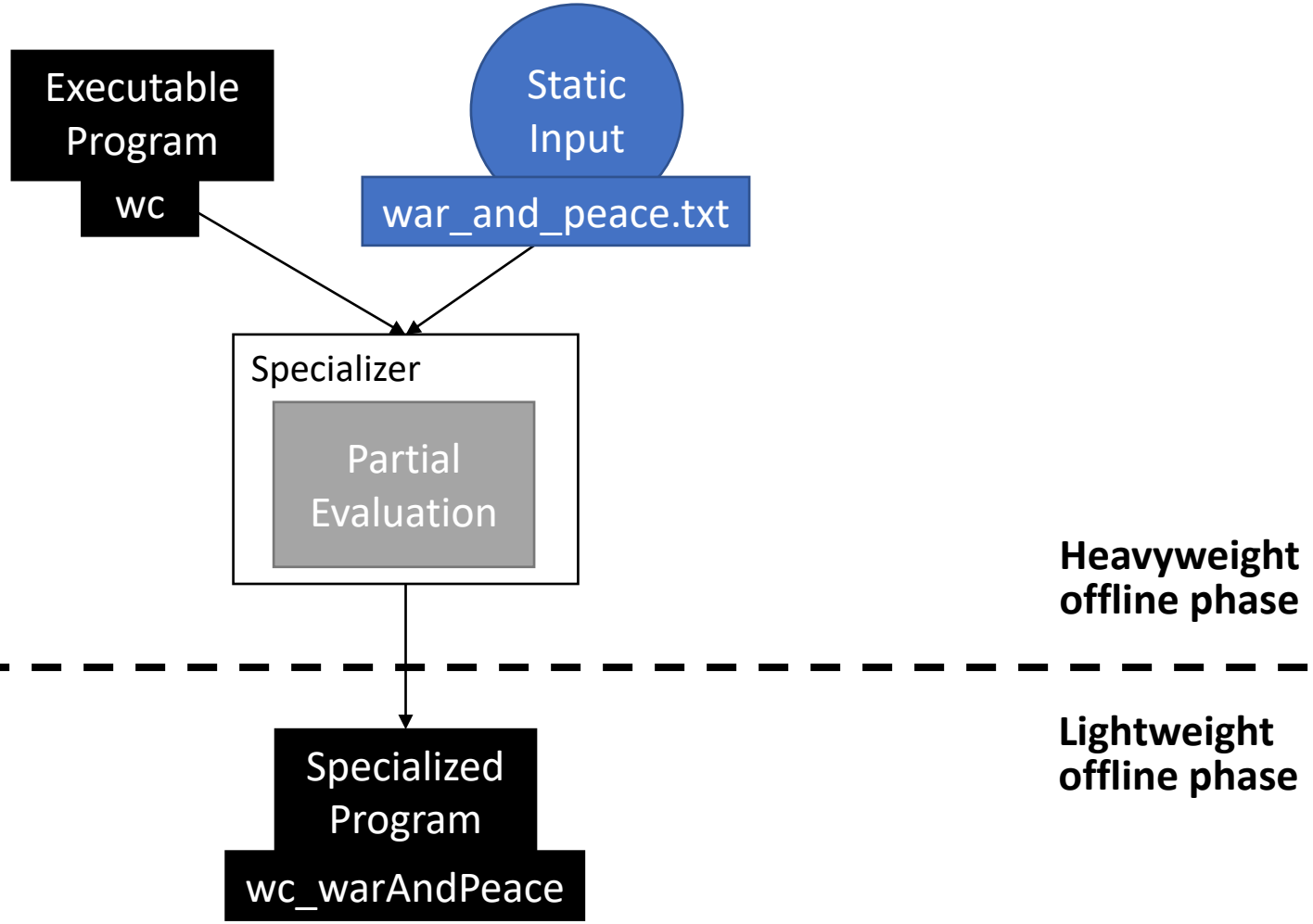
3rd Futamura Projection

Baseline Specialization

Partial Evaluation – Futamura Projections

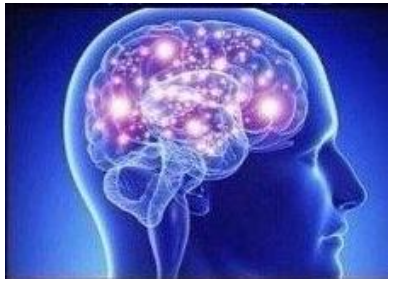


Specialize a program on some of its input

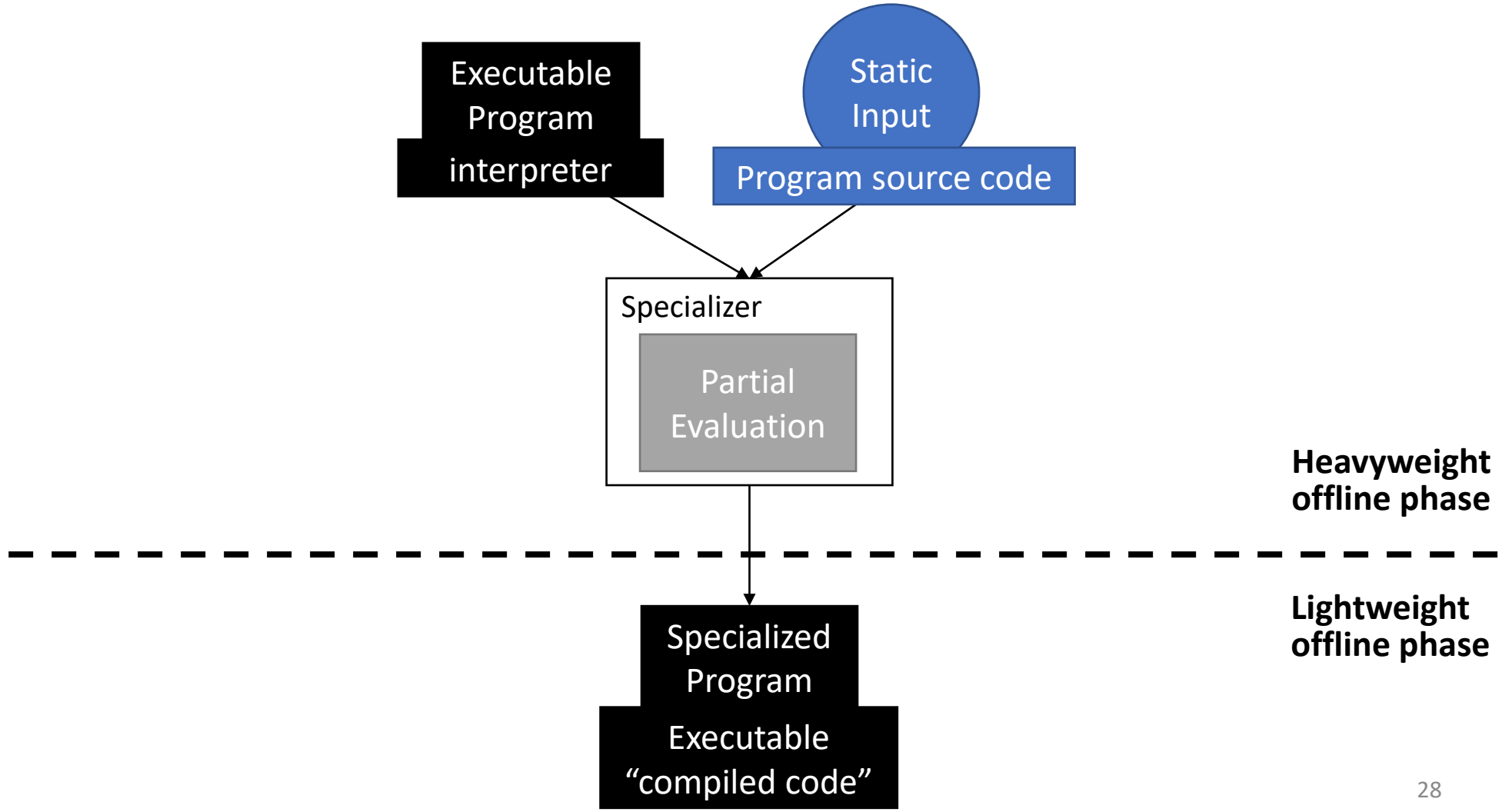


1st Futamura Projection

Partial Evaluation – Futamura Projections



Specialize an interpreter on program code

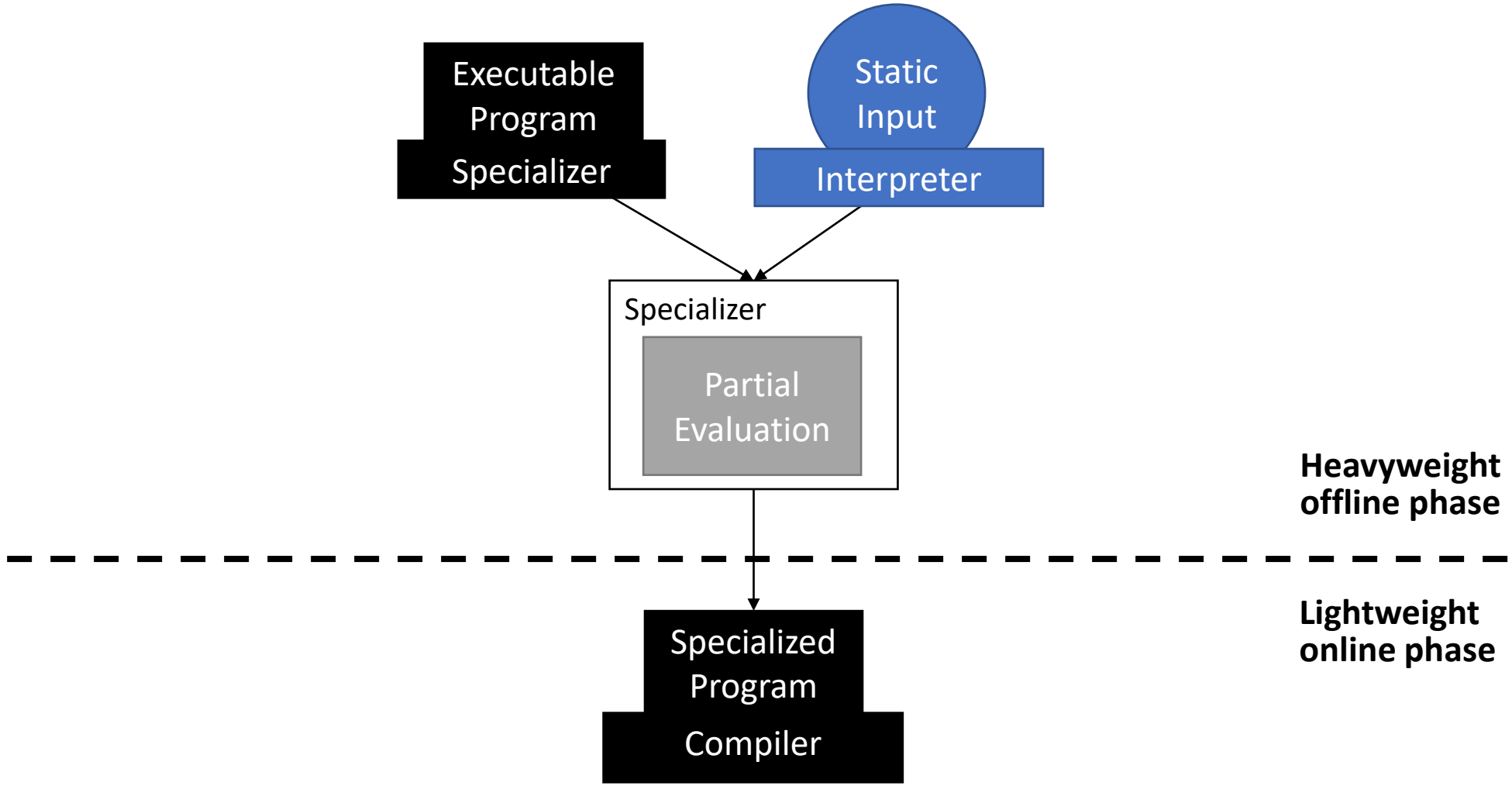


2nd Futamura Projection

Partial Evaluation – Futamura Projections



Specialize the specializer on the interpreter code

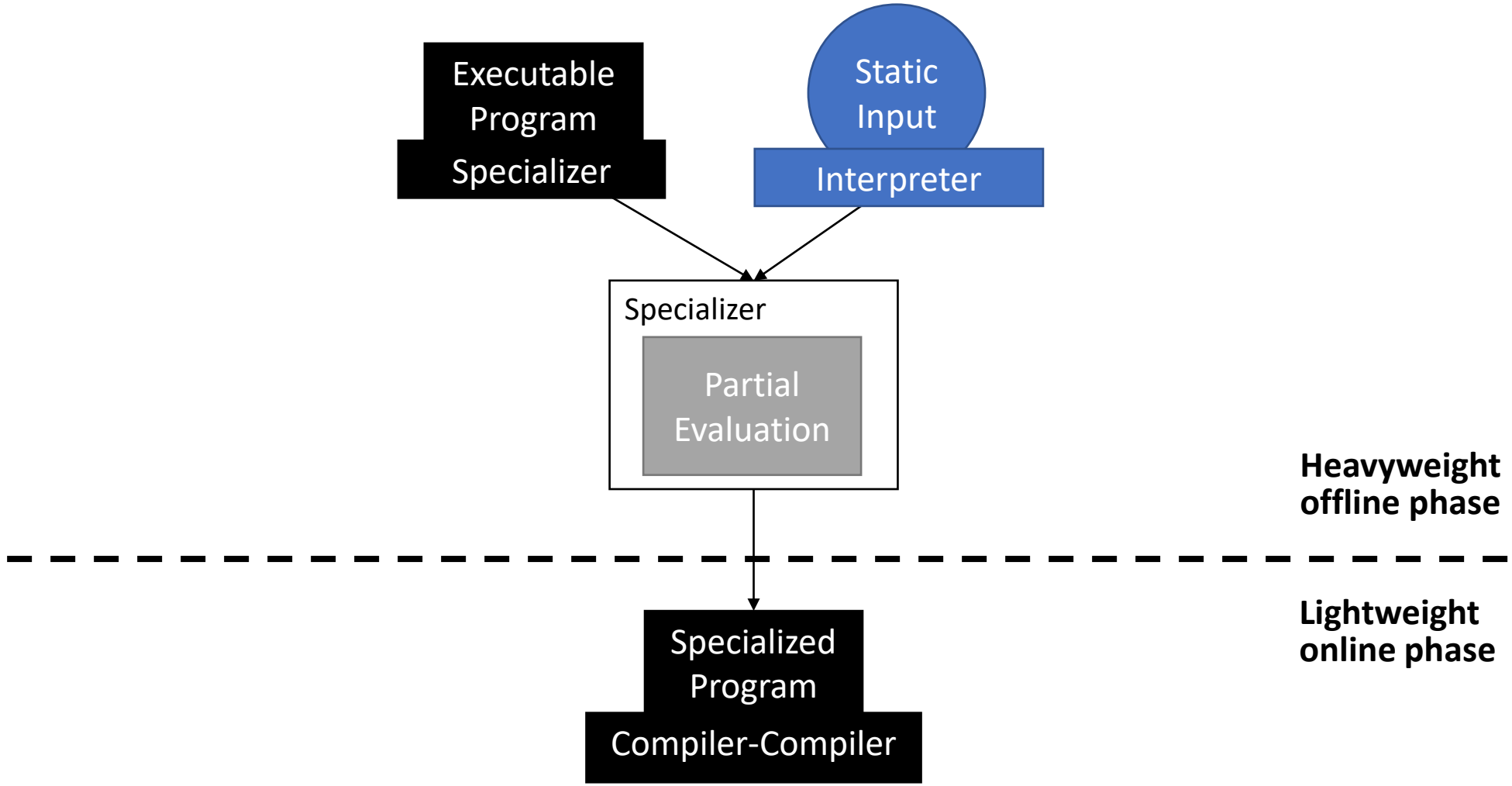


3rd Futamura Projection

Partial Evaluation – Futamura Projections



Specialize the specializer on the interpreter code



The Futamura Projections

Partial Evaluation – Futamura Projections



Baseline specialization
(not a Futamura Projection)

1st Futamura Projection

Specialize interpreter on a program

Use specializer as a (slow) compiler

2nd Futamura Projection

Specialize specializer on interpreter

Use specializer to (slowly) build a compiler

3rd Futamura Projection

Specialize specializer on specializer

Use specializer to build a program that builds compilers

The Futamura Projections

Partial Evaluation – Futamura Projections



Baseline specialization
(not a Futamura Projection)

1st Futamura Projection

Specialize interpreter on a program

Use specializer as a (slow) compiler

2nd Futamura Projection

Specialize specializer on interpreter

Use specializer to (slowly) build a compiler

3rd Futamura Projection

Specialize specializer on specializer

Use specializer to build a program that builds compilers

Futamura Projections: WTF?

Partial Evaluation – Futamura Projections

Why would you do this?

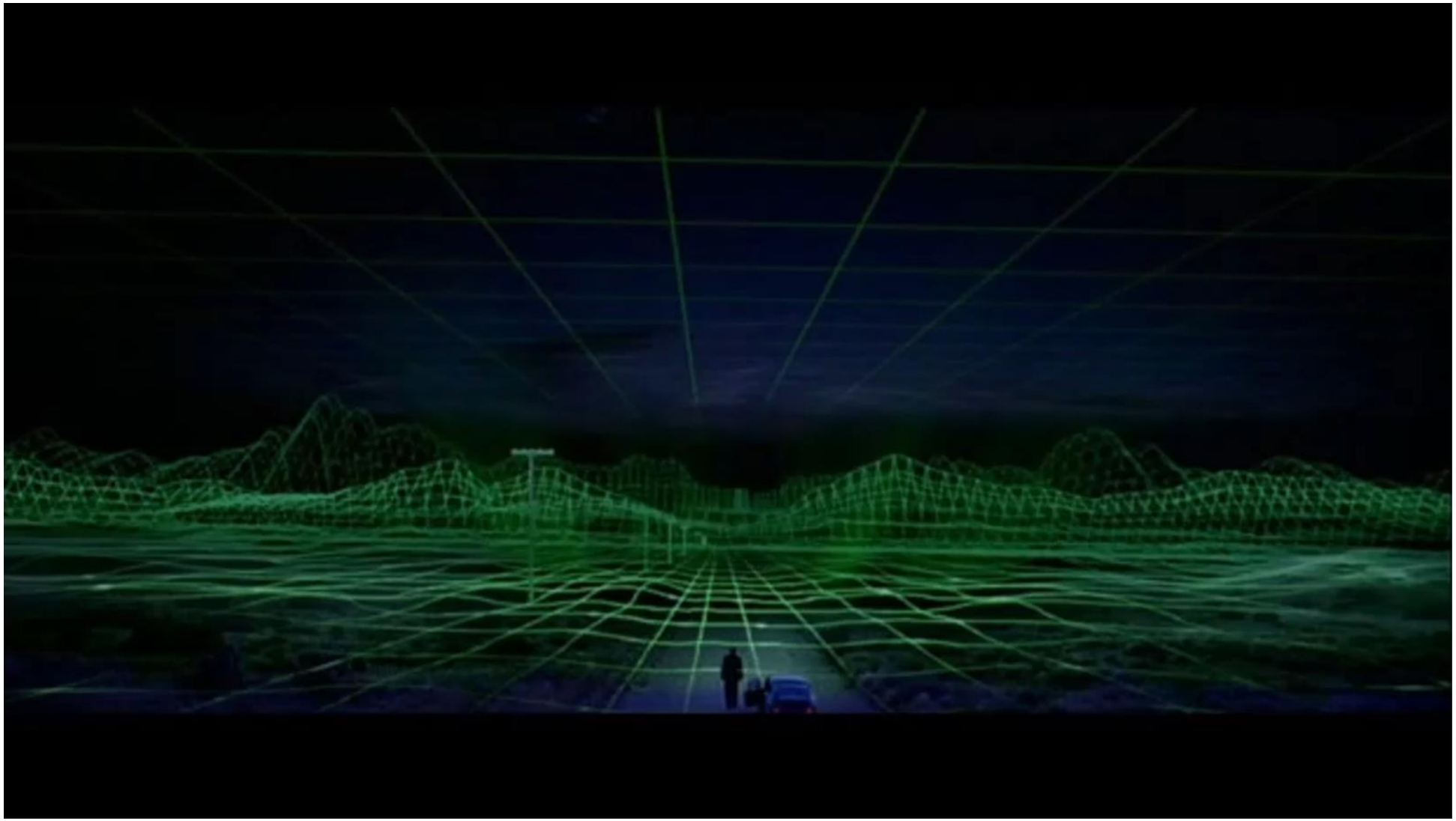
- Reduces Effort
  - Interpreters are nice! So are compilers!
  - You want both, you can get both by just building interpreters

Is this real?

- Satisfying the definitions is easy
- Making good specialized programs is not

Another Frontier in Computer Science

Partial Evaluation – Futamura Projections



End of Lecture: Summary

Partial Evaluation – Futamura Projections

Summary

- Specialize program to enable optimization
- Treat some input as static, some as dynamic
- Powerful technique with ability to repurpose compiler components

Next Time: Other Compilers

Lecture Preview

- Look at other types of compilers, some of the niches filled by other compilation tools