Google

PDLL

Frontend Pattern Language

River Riddle

Background: PDL, Pattern Descriptor Language

This presentation assumes some background on PDL:

- 2021-04-15: Pattern Descriptor Language ; slides - recording
- 2019-12-19: Interpreted Pattern Match Execution ; slides - recording

Introduction

What is PDLL?

- Frontend DSL for writing MLIR pattern rewrites

What is PDLL?

- Frontend DSL for writing MLIR pattern rewrites
  - Codegens to PDL patterns (i.e. no C++ patterns, but allows C++/native code injection)
    - PDLL patterns can be compiled at runtime

What is PDLL?

- Frontend DSL for writing MLIR pattern rewrites
  - Codegens to PDL patterns (i.e. no C++ patterns, but allows C++/native code injection)
    - PDLL patterns can be compiled at runtime
- Designed as a high-level representation of MLIR and PatternRewriter constructs
  - Support for all MLIR constructs; Optional, Variadic, Region, Successor, etc.

What is PDLL?

- Frontend DSL for writing MLIR pattern rewrites
  - Codegens to PDL patterns (i.e. no C++ patterns, but allows C++/native code injection)
    - PDLL patterns can be compiled at runtime
- Designed as a high-level representation of MLIR and PatternRewriter constructs
  - Support for all MLIR constructs; Optional, Variadic, Region, Successor, etc.
- Built with modern language tooling in mind
  - Code completion, go-to-definition, formatting, etc.

Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use

Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?

Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?
  - Tablegen is not an ideal host language for MLIR constructs
    - Problems have arose with multi-result operations, replacing multi operations, constraint application, …

Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?
  - Tablegen is not an ideal host language for MLIR constructs
    - Problems have arose with multi-result operations, replacing multi operations, constraint application, …

```
// Suppose we have a three result operation, defined as seen below.
def ThreeResultOp : Op<"three_result_op"> {
    let arguments = (ins ...);

    let results = (outs
      AnyTensor:$output1,
      AnyTensor:$output2,
      AnyTensor:$output3
    );
}

// To bind the results of `ThreeResultOp` in a TDRR pattern, we bind all results
// to a single name and use a special naming convention: `__N`, where `N` is the
// N-th result.
def : Pattern<(ThreeResultOp:$results ...),
              [(... $results__0), ..., (... $results__2), ...]>;
```

Why PDLL?

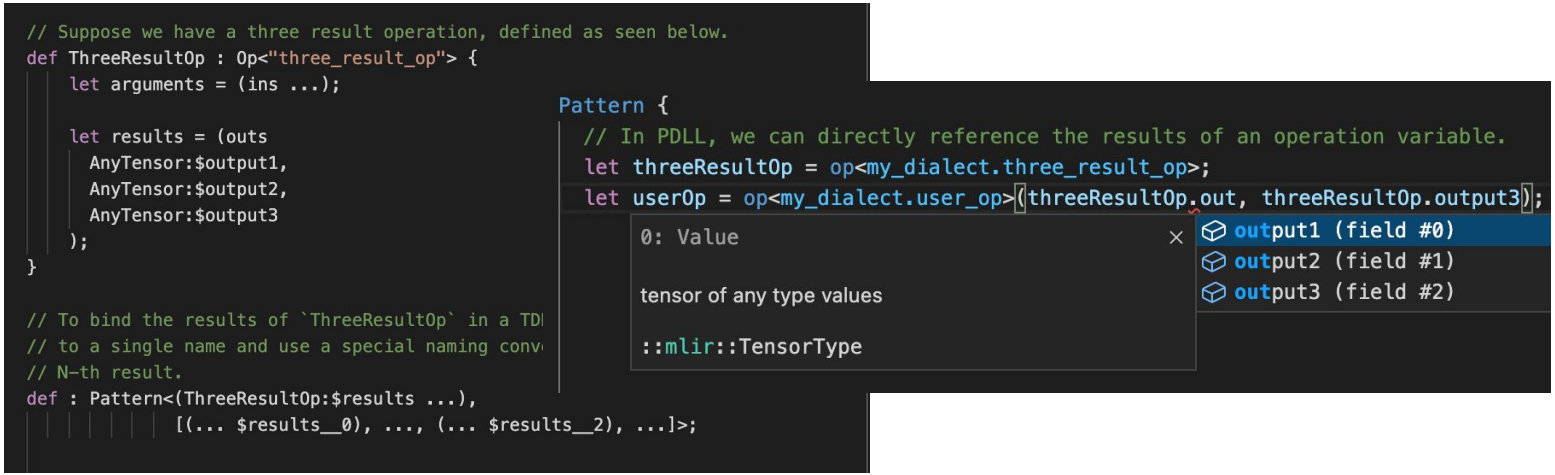
- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?
  - Tablegen is not an ideal host language for MLIR constructs
    - Problems have arose with multi-result operations, replacing multi operations, constraint application, …

```
// Suppose we have a three result operation, defined as seen below.
def ThreeResultOp : Op<"three_result_op"> {
    let arguments = (ins ...);

    let results = (outs
      AnyTensor:$output1,
      AnyTensor:$output2,
      AnyTensor:$output3
    );
}

// To bind the results of `ThreeResultOp` in a TDRR pattern, we bind all results
// to a single name and use a special naming convention: `__N`, where `N` is the
// N-th result.
def : Pattern<(ThreeResultOp:$results ...),
              [(... $results__0), ..., (... $results__2), ...]>;
```

```
Pattern {
  // In PDLL, we can directly reference the results of an operation variable.
  let threeResultOp = op<my_dialect.three_result_op>;
  let userOp = op<my_dialect.user_op>(threeResultOp.output1, threeResultOp.output3);
```

Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?
  - Tablegen is not an ideal host language for MLIR constructs
    - Problems have arose with multi-result operations, replacing multi operations, constraint application, …



Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?
  - Tablegen is not an ideal host language for MLIR constructs
    - Problems have arose with multi-result operations, replacing multi operations, constraint application, …
- Why not X language?
  - Yes and No

Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?
  - Tablegen is not an ideal host language for MLIR constructs
    - Problems have arose with multi-result operations, replacing multi operations, constraint application, …
- Why not X language?
  - No
    - Extra dependencies, environment, performance, tooling, subset of language to support, etc. considerations

Why PDLL?

- Provide a declarative rewrite infrastructure that is easy and pleasant to use
- What about Tablegen-DRR?
  - Tablegen is not an ideal host language for MLIR constructs
    - Problems have arose with multi-result operations, replacing multi operations, constraint application, …
- Why not X language?
  - No
    - Extra dependencies, environment, performance, tooling, subset of language to support, etc. considerations
  - Yes
    - PDL (the underlying infra) was designed with multiple frontends in mind
    - Different users have different constraints

Language Demo

Status

Roadmap

- Current Status
  - Everything in the demo
  - Focusing on useful initial feature set (e.g. what is needed to delete TDRR)

Roadmap

- Current Status
  - Everything in the demo
  - Focusing on useful initial feature set (e.g. what is needed to delete TDRR)
- Next steps
  - Support Blocks and Regions, DialectConversion type conversions, etc.
  - Performance improvements and optimizations for PDL

Roadmap

- Current Status
  - Everything in the demo
  - Focusing on useful initial feature set (e.g. what is needed to delete TDRR)
- Next steps
  - Support Blocks and Regions, DialectConversion type conversions, etc.
  - Performance improvements and optimizations for PDL
  - Formalize RFC and develop upstream

Thanks