Dialects as a Dialect

Bringing native C++ registration to IRDL

UNIVERSITY OF CAMBRIDGE

Ivan Ho

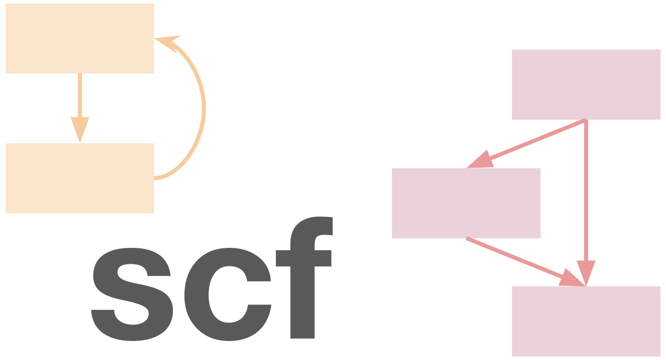
PhD Student

Supervised by Tobias Grosser

arith

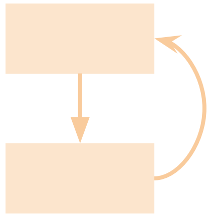
arith

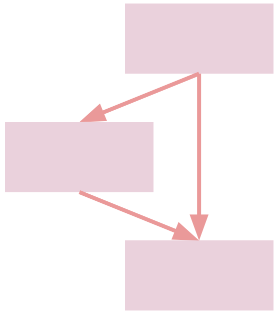
scf



arith

scf





func

arith

scf

func

transform

What if we had dialects as a dialect?

What if we had dialects as a dialect?

IRDL

＼＼\٩(๑`^´๑)۶//／／

Why a Dialect for Dialects?

```
irdl.dialect cmath {

  irdl.type complex {
    %0 = irdl.is f32
    %1 = irdl.is f64
    %2 = irdl.any_of(%0, %1)

    irdl.parameters (elem: %2)
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

Why a Dialect for Dialects?

```
irdl.dialect cmath {
















}
```

A ***Dialect*** definition is a single operation.

Why a Dialect for Dialects?

```
irdl.dialect cmath {

  irdl.type complex {
    %0 = irdl.is f32
    %1 = irdl.is f64
    %2 = irdl.any_of(%0, %1)

    irdl.parameters (elem: %2)
  }









}
```

A ***Dialect*** definition is a single operation.

Types may have constraints.

Why a Dialect for Dialects?

```
irdl.dialect cmath {

  irdl.type complex {
    %0 = irdl.is f32
    %1 = irdl.is f64
    %2 = irdl.any_of(%0, %1)

    irdl.parameters (elem: %2)
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

A ***Dialect*** definition is a single operation.

Types may have constraints.

An ***Operation*** is defined similarly.

```
irdl.dialect cmath {

  irdl.type complex {
    // ...
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

```
$ mlir-opt  --irdl-file=cmath.irdl.mlir  example.cmath.mlir
```

```
irdl.dialect cmath {

  irdl.type complex {
    // ...
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

IRDL dialects are loaded dynamically

```
$ mlir-opt  --irdl-file=cmath.irdl.mlir  example.cmath.mlir
```

```
irdl.dialect cmath {

  irdl.type complex {
    // ...
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

```
$ mlir-opt  --irdl-file=cmath.irdl.mlir  example.cmath.mlir
```

```
irdl.dialect cmath {

  irdl.type complex {
    // ...
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

```
%0 = "cmath.norm" (%arg0)
      : (!cmath.complex<f32>) -> f32
```

```
$ mlir-opt  --irdl-file=cmath.irdl.mlir  example.cmath.mlir
```

```
irdl.dialect cmath {

  irdl.type complex {
    // ...
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

```
%0 = "cmath.norm" (%arg0)
       : (!cmath.complex<f32>) -> f32
```

```
%0 = "cmath.norm" (%arg0)
       : (!cmath.complex<f32>) -> f64
```

❌ **unsatisfied constraint**

IRDL: IR Definition Language

Concise

Introspectable

Dynamic

Generatable

And it *just* works…

IRDL: IR Definition Language

Concise

Introspectable

Dynamic

Generatable

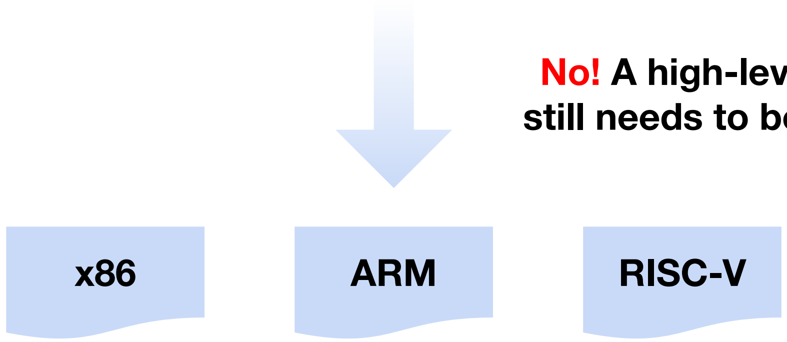
And it *just* works… kind of

Is parsing this program enough?

```
%0 = "cmath.norm" (%arg0) : (!cmath.complex<f32>) -> f32
```

Is parsing this program enough?

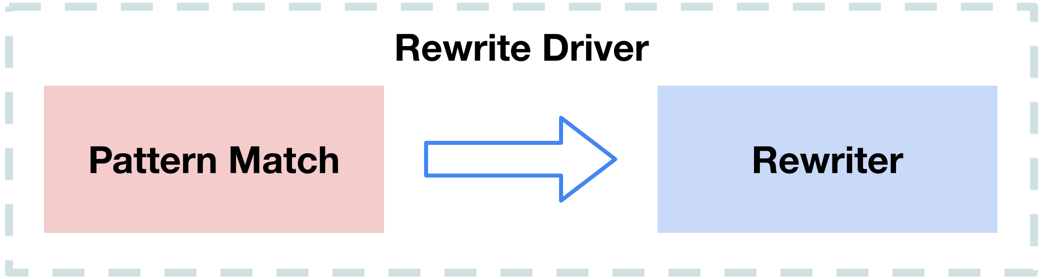
```
%0 = "cmath.norm" (%arg0) : (!cmath.complex<f32>) -> f32
```



No! A high-level dialect still needs to be lowered!

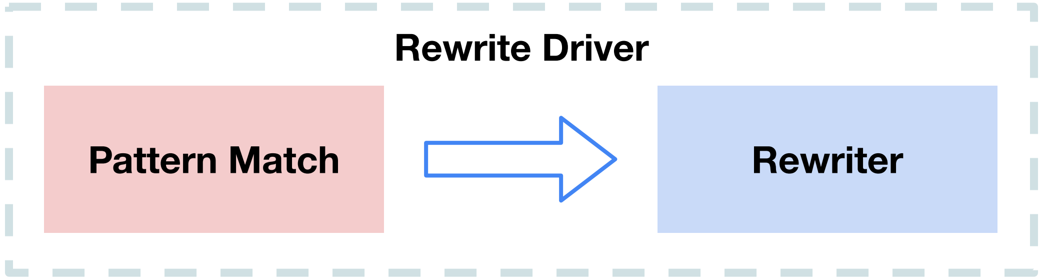
How to Rewrite IR in MLIR

(the abridged version)



How to Rewrite IR in MLIR

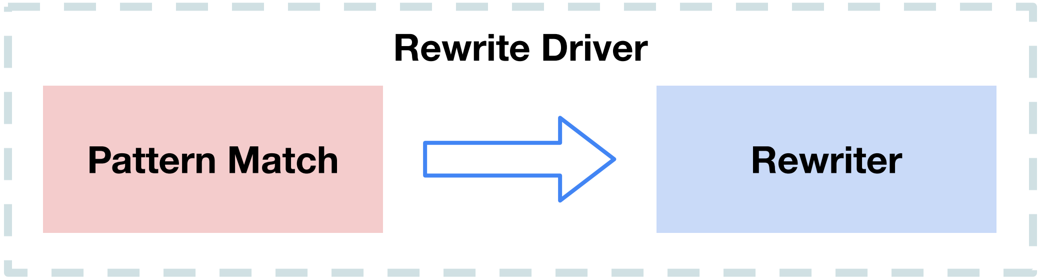
(the abridged version)



```
void matchAndRewrite(arith::AddFOp op, OpAdaptor op,
                     ConversionPatternRewriter& rewriter)
{
  // ...
  rewriter.create<arith::MulFOp>(loc, ...)
  rewriter.replaceWithNewOp<arith::MulFOp>(loc, op, ...)
}
```

How to Rewrite IR in MLIR

(the abridged version)



```
void matchAndRewrite(arith::AddFOp op, OpAdaptor op,
                     ConversionPatternRewriter& rewriter)
{
  // ...
  rewriter.create<arith::MulFOp>(loc, ...)
  rewriter.replaceWithNewOp<arith::MulFOp>(loc, op, ...)
}
```

MLIR infrastructure uses **static C++ types**!

```
mlir-opt  --irdl-file=cmath.irdl.mlir  example.cmath.mlir
```

IRDL dialects are loaded **dynamically**

MLIR infrastructure uses **static C++ types**!

```
mlir-opt  --irdl-file=cmath.irdl.mlir  example.cmath.mlir
```

IRDL dialects are loaded **dynamically**

IRDL doesn’t work with the existing MLIR C++ infrastructure

MLIR infrastructure uses **static C++ types**!

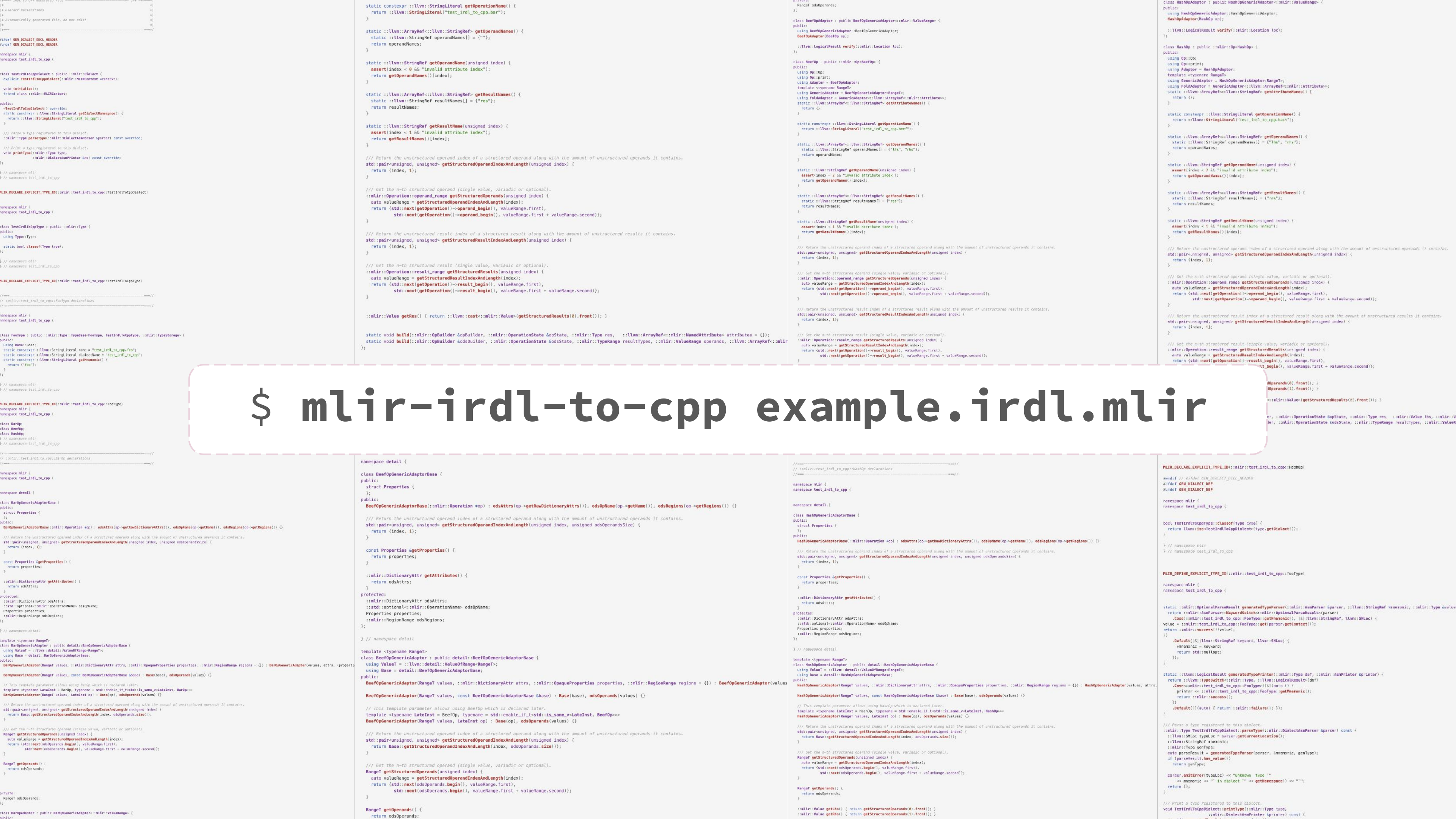
```
mlir-opt  --irdl-file=cmath.irdl.mlir  example.cmath.mlir
```

IRDL dialects are loaded **dynamically**

IRDL didn't work with the existing MLIR C++ infrastructure

MLIR infrastructure uses **static C++ types**!

```
$ mlir-irdl-to-cpp example.irdl.mlir
```



```
$ mlir-irdl-to-cpp example.irdl.mlir
```

```cpp
void matchAndRewrite(cmath::NormOp op, OpAdaptor op,
                     ConversionPatternRewriter& rewriter)
{
  // ...
  rewriter.create<cmath::RealOp>(loc, ...)
  rewriter.replaceWithNewOp<cmath::ImagOp>(loc, op, ...)
}
```

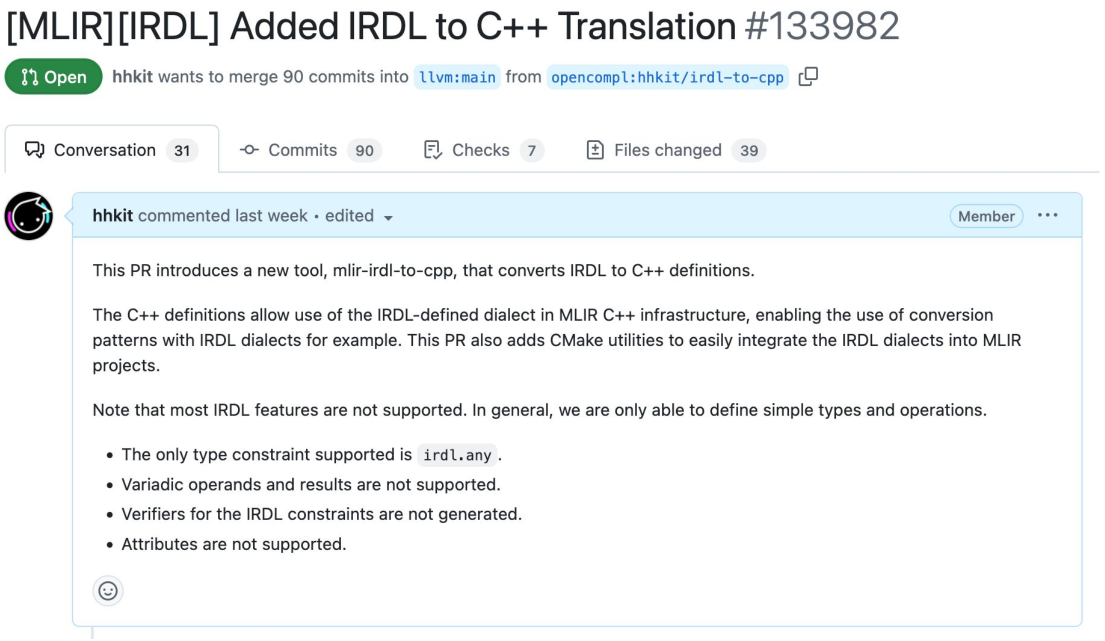
CMakeLists.txt

```
add_irdl_to_cpp_target(TestIRDLToCppGen test_irdl_to_cpp.irdl.mlir)
```

```
// CHECK: func.func @test() {
// CHECK: %[[v0:[^ ]*]] = "test_irdl_to_cpp.bar"() : () -> i32
// CHECK: %[[v1:[^ ]*]] = "test_irdl_to_cpp.bar"() : () -> i32
// CHECK: %[[v2:[^ ]*]] = "test_irdl_to_cpp.hash"(%[[v0]], %[[v0]]) : (i32, i32) -> i32
// CHECK: return
// CHECK: }
func.func @test() {
    %0 = "test_irdl_to_cpp.bar"() : () -> i32
    %1 = "test_irdl_to_cpp.beef"(%0, %0) : (i32, i32) -> i32
    return
}
```

test_conversion.testd.mlir

Merged?



[MLIR][IRDL] Added IRDL to C++ Translation #133982

Open
hhkit wants to merge 90 commits into llvm:main from opencompl:hhkit/irdl-to-cpp
Conversation 31
Commits 90
Checks 7
Files changed 39
hhkit commented last week • edited
Member

This PR introduces a new tool, mlir-irdl-to-cpp, that converts IRDL to C++ definitions.

The C++ definitions allow use of the IRDL-defined dialect in MLIR C++ infrastructure, enabling the use of conversion patterns with IRDL dialects for example. This PR also adds CMake utilities to easily integrate the IRDL dialects into MLIR projects.

Note that most IRDL features are not supported. In general, we are only able to define simple types and operations.

- The only type constraint supported is `irdl.any`.
- Variadic operands and results are not supported.
- Verifiers for the IRDL constraints are not generated.
- Attributes are not supported.

IRDL: IR Definition Language

Concise

Dynamic

```
irdl.dialect cmath {

  irdl.type complex {
    %0 = irdl.is f32,     %1 = irdl.is f64
    %2 = irdl.any_of(%0, %1)

    irdl.parameters (elem: %2)
  }

  irdl.operation norm {
    %0 = irdl.any
    %1 = irdl.parametric @cmath::@complex(%0)

    irdl.operands (in: %1)
    irdl.results (res: %0)
  }
}
```

Introspectable

Generatable

IRDL Devs

Mathieu Fehr
email: mathieufehr@gmail.com
github: math-fehr

Théo Degioanni
github: Moxinilian
discord: moxinilian

Ivan Ho
email: ivan@hhkit.dev
github: hhkit

Resources

https://godbolt.org/z/Ya54Whvd8