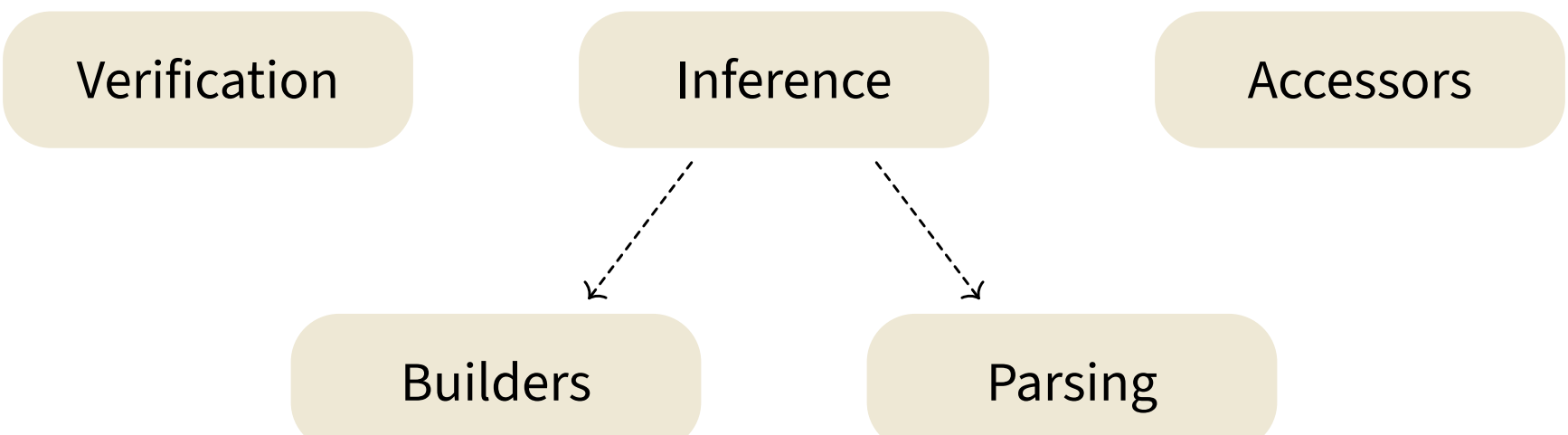
THE UNIVERSITY *of* EDINBURGH

Defining and Verifying MLIR Operations with Constraints

Alex Rice

EuroLLVM 2025

What do operation definitions do?



Example: `arith.addi`

```
%0 = arith.addi %1, %2 : i32
```

Example: `arith.addi`

```
%0 = arith.addi %1, %2 : i32
```

Constraints cannot be defined independently:

```
"arith.addi"(%1, %2) : (i32, i64) -> i8
```

Example: `arith.addi`

```
%0 = arith.addi %1, %2 : i32
```

Constraints cannot be defined independently:

```
"arith.addi"(%1, %2) : (i32, i64) -> i8
```

Constraints for `arith.addi`

```
class AddIOp:
    _T: ClassVar = VarConstraint("T", signlessIntegerLike)
    lhs    = operand_def(_T)
    rhs    = operand_def(_T)
    result = result_def(_T)

    assembly_format = "$lhs `,` $rhs attr-dict `:` type($result)"
```

Towards more complex constraints

```
class InsertOp:
    name = "vector.insert"

    _T: ClassVar = VarConstraint("T", AnyAttr())
    _V: ClassVar = VarConstraint("V", VectorType.constr(_T))

    source = operand_def(VectorType.constr(_T) | _T)
    dest   = operand_def(_V)
    result = result_def(_V)
    ...
```