Polyhedral Model and MLIR

Guillaume Iooss (INRIA Grenoble)

MLIR Summer School

September 2025

Goals of this panel

Polyhedral model and integration inside MLIR?

1. What is the polyhedral model? (quick intro if needed)
2. Polyhedral model ecosystem:
    - What are the existing tools/compilers?
    - Current status of polyhedral compilation in MLIR?
3. What are the current/next steps? (Open discussion)

Goals of this panel

Polyhedral model and integration inside MLIR?

1. What is the polyhedral model? (quick intro if needed)
2. Polyhedral model ecosystem:
   - What are the existing tools/compilers?
   - Current status of polyhedral compilation in MLIR?
3. What are the current/next steps? (Open discussion)

These slides are only support/context for discussion

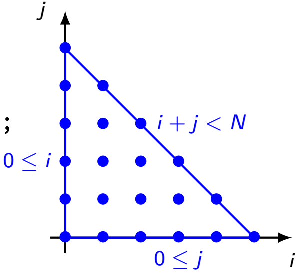
⇒ Feel free to interrupt/branch out on other topics

Polyhedral Model

- Exploit the regularity of structure/accesses of kernels

```
for (i=0; i<N; i++)
  for (j=0; j<N-i; j++)
S: A[i,j] = A[i,j-1] + B[j,i+2j];
```

$$\mathcal{D}_S = \{i, j \mid 0 \leq i, j \;\&\; i + j < N\}$$



- Concise mathematical representation of programs.
- Mathematical objects used to represent properties:
  - Set of integer points ⇒ $\mathcal{Z}$-polyhedron (Presburger set)
  - Relation between points ⇒ Affine function/relation
- Symbolic constants (parameters), usually for the problem size.

Polyhedral Model

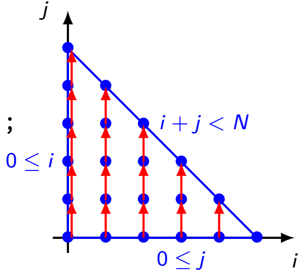
- Exploit the regularity of structure/accesses of kernels

```
for (i=0; i<N; i++)
  for (j=0; j<N-i; j++)
S: A[i,j] = A[i,j-1] + B[j,i+2j];
```

used by

$$\mathcal{D}_S = \{i,j \mid 0 \leq i,j \ \& \ i+j < N\}$$

$$f_{dep} = (i,j \ \mapsto \ i,j+1)$$



- Concise mathematical representation of programs.
- Mathematical objects used to represent properties:
  - Set of integer points ⇒ $\mathcal{Z}$-polyhedron (Presburger set)
  - Relation between points ⇒ Affine function/relation
- Symbolic constants (parameters), usually for the problem size.

Why do we want polyhedral compilation?

- **Application domain:** Focus on affine computations
  - Linear algebra, stencil operations, dynamic programming, . . .

  ⇒ Machine learning, physical simulations, solver, . . .

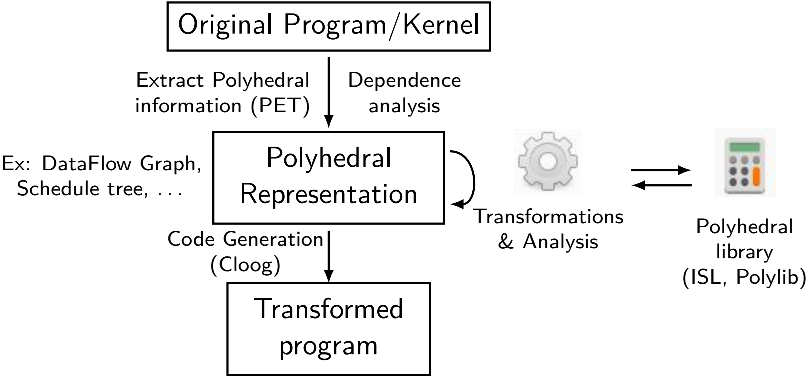
Why do we want polyhedral compilation?

- **Application domain:** Focus on affine computations
  - Linear algebra, stencil operations, dynamic programming, . . .

  ⇒ Machine learning, physical simulations, solver, . . .

- (Non-exhaustive) **List of contributions:**
  - Many polyhedral loop transformation + static analysis:
    - Loop tiling, skewing, fusion, . . .
    - Automatic vectorization, parallelization, . . .
    - Dependence analysis, scheduling, cache modeling, . . .
    - Program verification, termination, . . .

Typical workflow of a polyhedral compiler



Original Program/Kernel

Extract Polyhedral information (PET)

Dependence analysis

Ex: DataFlow Graph, Schedule tree, . . .

Polyhedral Representation

Transformations & Analysis

Polyhedral library (ISL, Polylib)

Code Generation (Cloog)

Transformed program

Mathematical “toolbox” (library)

Library that provides representation/operations (ISL, Polylib)

- Basic operations:
  - Intersection, difference, emptiness check, . . .
  - Image, preimage, composition, . . .
- More complicated operations:
  - Finding a (parametric) lexicographic maximum (PipLib)
  - Counting integer points (Barvinok)
  - Transitive closure
- Careful with scalability ! (some algo have exponential cost)
  - It usually goes well, but not a guaranty (sometimes explode)

Resources about the polyhedral model (for newcomers)

- Website with bibliography list: `https://polyhedral.info/`
- IMPACT workshop (collocated with HiPeac conference)
- A few tutorial resources:
  - Tutorial by Sven Verdoolaege (barvinok library)
  - OpenScop: good introduction
  - Louis-Noël Pouchet: class material
  - ...
- Just grab one expert and start asking questions.

Polyhedral tools ecosystem - outside of MLIR

- Most of the tools are written in C/C++.
  Usually designed to operate on C/Fortran programs (or DSL)

Polyhedral tools ecosystem - outside of MLIR

- Most of the tools are written in C/C++.
  Usually designed to operate on C/Fortran programs (or DSL)
- Useful tools for specific tasks:
  - Polyhedral extractor: PET
  - Maths: Piplib (for lexmax), ISL/barvinok, Polylib, ...
  - Code generation: Cloog
  - Benchmark suite: Polybench
  - And many more (lots of prototypes)...
- Src-to-src compiler: Pluto, PPCG, POCC, Rose/PolyOpt

Polyhedral tools ecosystem - outside of MLIR

- Most of the tools are written in C/C++.
  Usually designed to operate on C/Fortran programs (or DSL)
- Useful tools for specific tasks:
  - Polyhedral extractor: PET
  - Maths: Piplib (for lexmax), ISL/barvinok, Polylib, . . .
  - Code generation: Cloog
  - Benchmark suite: Polybench
  - And many more (lots of prototypes). . .
- Src-to-src compiler: Pluto, PPCG, POCC, Rose/PolyOpt
- Integration in mainstream compilers:
  - GCC ⇒ Graphite
  - LLVM ⇒ Polly

  Lessons on how successful (or not) they are/were?

MLIR dialects - `affine`

MLIR `affine` dialect: (+ `memref`)

- Good entry point for polyhedral analysis:
  - Express polyhedral program/fragments of program
  - Affine mapping manipulation (ex: `apply`)
  - `-lower-affine`: lower to `arith`/`scf`

- Affine dialect passes:
  - Some loop transformations (tiling, fusion, . . .)
  - Did not see any analysis (ex: dependence analysis)
  - ⇒ No elaborate analysis/heuristic/transformation.

Other pertinent parts of MLIR

Analysis/Presburger - "ISL/barvinok" in MLIR:

- `llvm-project/mlir/include/mlir/Analysis/Presburger`
- `IntegerRelation.h`: similar to ISL BasicMap/BasicSet
- `Barvinok.h` : Re-implem of Barvinok algorithm

⇒ Any idea of the current status?
(Done? Maintained? How much from ISL is here?)

Other pertinent parts of MLIR

Analysis/Presburger - "ISL/barvinok" in MLIR:

- `llvm-project/mlir/include/mlir/Analysis/Presburger`
- `IntegerRelation.h`: similar to ISL BasicMap/BasicSet
- `Barvinok.h` : Re-implem of Barvinok algorithm

⇒ Any idea of the current status?
(Done? Maintained? How much from ISL is here?)

MLIR `linalg` dialect:

- Tensor operations
- Can be lowered to `affine`, raising available from `affine`

Polygeist (Polymer)

Polygeist:

- Interfacing between MLIR and external polyhedral tools.
  - C code to `scf` then raise to `affine`.
  - `affine` Dialect ⇔ OpenScop
  - OpenScop: interface available for Pluto, Cloog, ISL

Discussions (if not already done)

- Any other efforts that was not listed?
  (Is there some unpublished efforts currently being made?)
- What form/shape should be the best for integrating polyhedral techniques in this ecosystem?
- What are the next steps?
- Further coordination/discussion