Recover Polyhedral Transformations From Polyhedral Schedule

Nelson Lossing, Walid Astaoui, Gianpietro Consolaro, Harenome Razanajato, Zhen Zhang, Denis Barthou
Huawei Technologies France

IMPACT 2024, 14th International Workshop on Polyhedral Compilation Techniques
January 17th, 2024
Munich, Germany

MindSpore HUAWEI

Polyhedral Representation

2D+1 Representation

Example with polyhedral schedule 2D+1 representation of seidel-2d kernel:

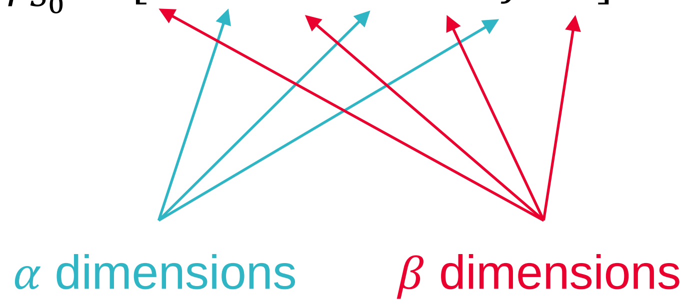
```c
static
void kernel_seidel_2d(int tsteps,
                      int n,
                      DATA_TYPE POLYBENCH_2D(A,N,N,n,n))
{
  int t, i, j;

#pragma scop
  for (t = 0; t <= _PB_TSTEPS - 1; t++)
    for (i = 1; i<= _PB_N - 2; i++)
      for (j = 1; j <= _PB_N - 2; j++)
        A[i][j] = (A[i-1][j-1] + A[i-1][j] + A[i-1][j+1]
                   + A[i][j-1] + A[i][j] + A[i][j+1]
                   + A[i+1][j-1] + A[i+1][j] + A[i+1][j+1])/SCALAR_VAL(9.0);
#pragma endscop

}
```

D corresponds to the number of iterators

$$\phi_{S_0} = [0 \quad t \quad 0 \quad i \quad 0 \quad j \quad 0]$$



HUAWEI

Motivating example

bicg

- Initial schedule

$$\phi_{S_0} = [0 \quad i \quad 0]$$
$$\phi_{S_1} = [1 \quad i \quad 0]$$
$$\phi_{S_2} = [1 \quad i \quad 1 \quad j \quad 0]$$
$$\phi_{S_3} = [1 \quad i \quad 1 \quad j \quad 1]$$

```
#pragma scop
  for (i = 0; i < _PB_M; i++) {
    s[i] = 0;
  }
  for (i = 0; i < _PB_N; i++) {
    q[i] = SCALAR_VAL(0.0);
    for (j = 0; j < _PB_M; j++) {
      s[j] = s[j] + r[i] * A[i][j];
      q[i] = q[i] + A[i][j] * p[j];
    }
  }
#pragma endscop
}
```

Chlore transformation:

reorder([1], [1,0]);
split([1,0], 1);
reorder([], [0,2,1]);
fuse([0]);
reorder([1,0], [1,0]);
split([1,0,0], 2);
split([1,0], 1);
reorder([], [0,2,1]);
fuse([0]);
fuse([0]);
fuse([0,2]);
interchange([0,2,0], 1, 2, 0);
embed([0,1]);
embed([0,0]);

Our recovery transformation:

Split([1, 1], 0);
Fuse([0]);
Fuse([0]);
Fuse([0, 0]);
Fuse([0, 0]);
Interchange([2], 1, 3);

- Compute schedule

$$\phi_{S_0} = [i \quad 0 \quad 0]$$
$$\phi_{S_1} = [i \quad 0 \quad 1]$$
$$\phi_{S_2} = [j \quad i \quad 2]$$
$$\phi_{S_3} = [i \quad j \quad 3]$$

```
#pragma scop
  /* Scattering iterators. */
  int t1, t2;
  int lbp, ubp;

  if ((_PB_M >= 1) && (_PB_M == _PB_N)) {
    lbp=0;
    ubp=_PB_M-1;
#pragma omp parallel for private(t2, )
     for (t1=lbp;t1<=ubp;t1++) {
      s[t1] = 0;
      q[t1] = SCALAR_VAL(0.0);
      for (t2=0;t2<=_PB_M-1;t2++) {
        s[t1] = s[t1] + r[t2] * A[t2][t1];
        q[t1] = q[t1] + A[t1][t2] * p[t2];
      }
    }
  }
#pragma endscop
}
```

*with CLooG, context: _PB_M==_PB_N

L. Bagnères et al, Opening polyhedral compiler's black box, CGO'16

Transformation Primitives

- Transformation primitives either modify the α dimension, either modify the β dimensions.
- All transformation primitives are invertible.

Name	Type	Description
$Reorder(\vec{\beta}, \vec{p})$	β-primitive	Reorder inner-loops or statements directly beneath the given outer-loop.
$Split(\vec{\beta})$	β-primitive	Split outer-loop just before given inner-loop or statement.
$Fuse(\vec{\beta})$	β-primitive	Fuse given loop with the next one on the same depth.
$Embed(\vec{L})$ $Unembed(\vec{L})$	α-primitive	Embed given statements beneath an innermost one-iteration extra loop. Unembed removes the added innermost extra loop.
$Reverse(\vec{L}, d)$	α-primitive	Reverse given output dimension for given statements.
$Grain(\vec{L}, d, c)$ $Densify(\vec{L}, d, c)$	α-primitive	Add some pad between consecutive iterations in given output dimension for given statements. Densify removes some/all of the pad.
$Shift(\vec{L}, d, c, \vec{C})$	α-primitive	Shift given output dimension by some (parametric) coefficient(s) for given statements.
$Interchange(\vec{L}, d_1, d_2)$	α-primitive	Interchange two given output dimensions for given statements.
$Skew(\vec{L}, d_1, d_2, c)$	α-primitive	Skew first output dimension by a coefficient of the second output dimension.
$Reshape(\vec{L}, d, d_{input}, c)$	α-primitive	Skew given output dimension by a coefficient of the given input dimension.

- $\vec{\beta}$: beta-vector targeting an entity
- $\vec{p}$: permutation vector
- $\vec{L}$: list of schedule IDs
- $\vec{C}$: list of parametric shift coefficients
- d, d_1, d_2, d_{input} : an output/input dimension
- c : a scalar coefficient

Recovery Algorithm

Recovery Algorithm

Normalization

$$\begin{aligned}
\phi_{S0} &: [\; 5 \quad i \quad j \quad 1 \quad 0 \quad 0 \quad k \;] \\
\phi_{S1} &: [\; 5 \quad i \quad j \quad 1 \quad 1 \quad 0 \quad k \;] \\
\phi_{S2} &: [\; 5 \quad -i \quad j \quad 1 \quad 1 \quad 0 \quad k \;] \\
\phi_{S3} &: [\; 5 \quad i \quad j \quad 0 \quad 0 \quad 0 \quad k \;] \\
\phi_{S4} &: [\; 5 \quad i \quad 0 \quad 0 \quad 0 \quad 0 \quad k \;]
\end{aligned}$$

(a) Initial

$$\begin{aligned}
&[\; 5 \quad i \quad j \quad 1 \quad k \;] \\
&[\; 5 \quad i \quad j \quad 2 \quad k \;] \\
&[\; 5 \quad -i \quad j \quad 2 \quad k \;] \\
&[\; 5 \quad i \quad j \quad 0 \quad k \;] \\
&[\; 5 \quad i \quad 0 \quad 0 \quad k \;]
\end{aligned}$$

(b) β-collapsing

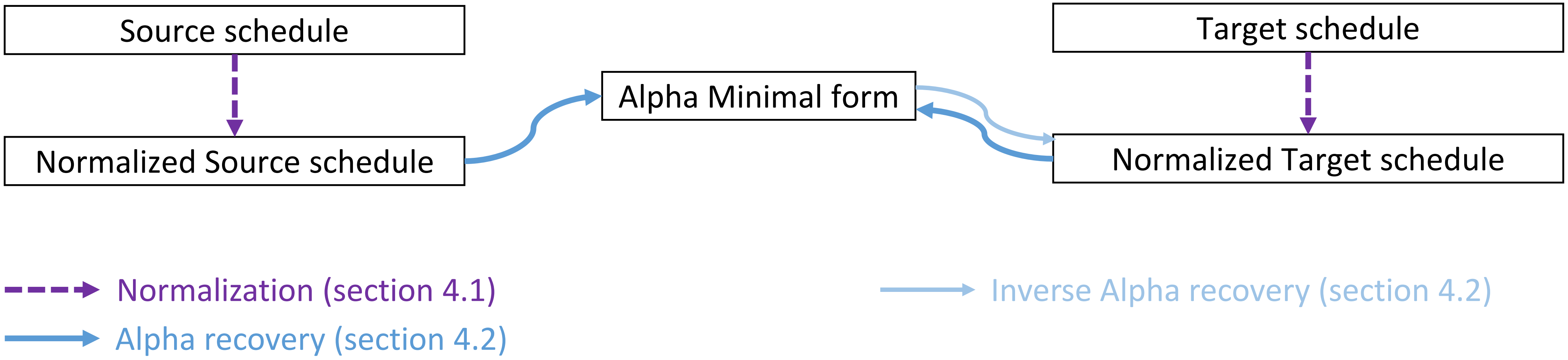
$$\begin{aligned}
&[\; 5 \quad i \quad 0 \quad j \quad 1 \quad k \quad 0 \;] \\
&[\; 5 \quad i \quad 0 \quad j \quad 2 \quad k \quad 0 \;] \\
&[\; 5 \quad -i \quad 0 \quad j \quad 2 \quad k \quad 0 \;] \\
&[\; 5 \quad i \quad 0 \quad j \quad 0 \quad k \quad 0 \;] \\
&[\; 5 \quad i \quad 0 \quad 0 \quad 0 \quad k \quad 0 \;]
\end{aligned}$$

(c) $2d + 1$ format

$$\begin{aligned}
&[\; 0 \quad i \quad 0 \quad j \quad 1 \quad k \quad 0 \;] \\
&[\; 0 \quad i \quad 0 \quad j \quad 2 \quad k \quad 0 \;] \\
&[\; 0 \quad -i \quad 0 \quad j \quad 2 \quad k \quad 1 \;] \\
&[\; 0 \quad i \quad 0 \quad j \quad 0 \quad k \quad 0 \;] \\
&[\; 0 \quad i \quad 0 \quad 0 \quad 0 \quad k \quad 1 \;]
\end{aligned}$$

(d) β-normalization

Recovery Algorithm



Recovery Algorithm

α-recovery

- The α-recovery algorithm search which α transformations are required to achieve Alpha Minimal form
- Alpha Minimal form correspond to the identity between the input domain and the output schedule without the β dimensions:

$$\begin{aligned}
\phi_{S_0} &= [0 \quad i \quad 0 \quad j \quad 1 \quad k \quad 0] \\
\phi_{S_1} &= [0 \quad i \quad 0 \quad j \quad 2 \quad k \quad 0] \\
\phi_{S_2} &= [0 \quad -i \quad 0 \quad j \quad 2 \quad k \quad 1] \\
\phi_{S_3} &= [0 \quad i \quad 0 \quad j \quad 0 \quad k \quad 0] \\
\phi_{S_4} &= [0 \quad i \quad 0 \quad 0 \quad 0 \quad k \quad 1]
\end{aligned}$$

Initial (Normalize) Schedule

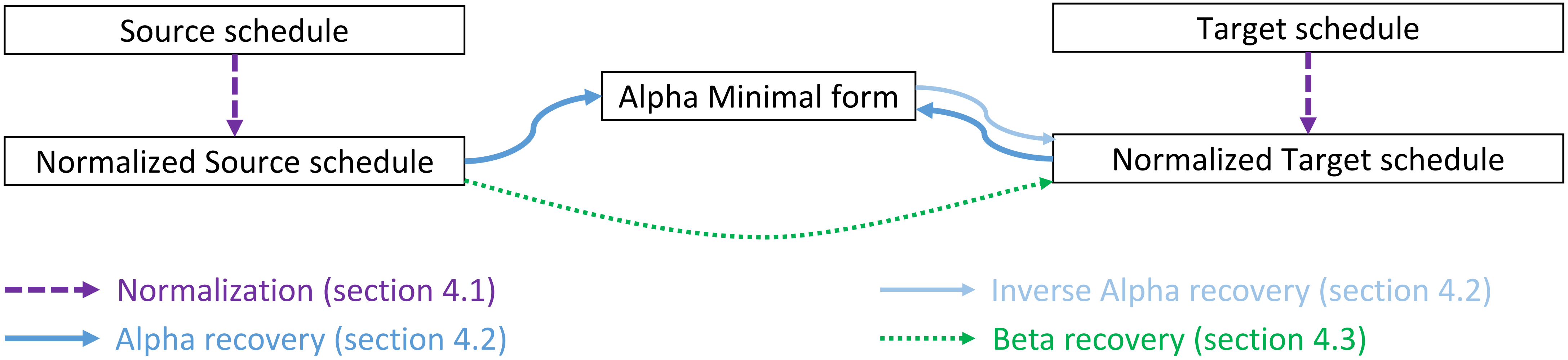
$$\begin{aligned}
\phi_{S_0} &= [i \quad j \quad k] \\
\phi_{S_1} &= [i \quad j \quad k] \\
\phi_{S_2} &= [i \quad j \quad k] \\
\phi_{S_3} &= [i \quad j \quad k] \\
\phi_{S_4} &= [i \quad k \quad]
\end{aligned}$$

Alpha Minimal form

- The α transformations are always considers in the same order:
  {Embed, Unembed} – {Densify, Reverse} – Shift – {Skew, Interchange} – Reshape
  - {Embed, Unembed}: modify number of schedule elements/dimensions
  - {Densify, Reverse}: modify the coefficient of an output dimension
  - Shift: add scalar coefficient of an output dimension
  - {Skew, Interchange} - Reshape: reorder of an output dimension, combine output dimensions together
- NB: The normalize source schedule is often already in the Alpha Minimal form

HUAWEI

Recovery Algorithm



Recovery Algorithm

β-recovery

- The β-recovery algorithm search which β transformations are required to transform source schedule to target schedule only considering the β dimensions
- The β transformations are repeatly consider in the same order:

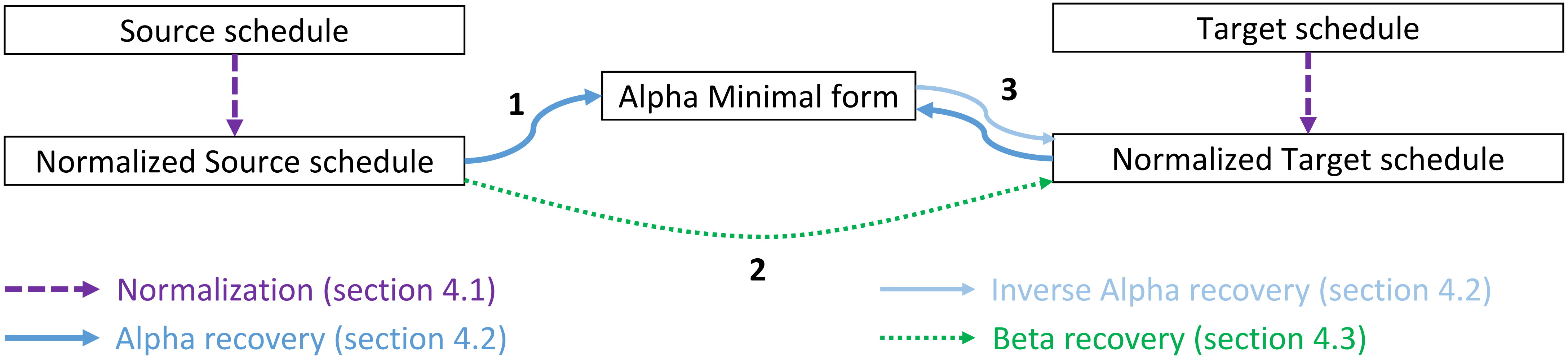
  Reorder – Split – Fuse

  > Reorder: modify the order of the statements on the same depth level

  > Split: separate a statement from the current group of fused statements

  > Fuse: fuse the next statement in the same depth level

Recovery Algorithm



Experiments

seidel-2d (example)

- Source schedule: $\phi_{S_0}^{S} = [0 \quad t \quad 0 \quad i \quad 0 \quad j \quad 0]$
- Target schedule: $\phi_{S_0}^{T} = [t \quad t+i \quad 2*t+i+j]$
- Target Normalized schedule : $\phi_{S_0}^{NT} = [0 \quad \underline{t} \quad 0 \quad \underline{t+i} \quad 0 \quad \underline{2*t+i+j} \quad 0]$

- Chlore (12 transformations)

```
reshape([0,0,0,0], 2, 3, 1);
grain([0,0,0,0], 1, 2);
reshape([0,0,0,0], 1, 3, -1);      find 2*t+i+j
reshape([0,0,0,0], 1, 2, -1);
interchange([0,0,0,0], 2, 3, 0);
skew([0,0,0,0], 2, 3, -1);
reverse([0,0,0,0], 2);
interchange([0,0,0,0], 1, 3, 0);   place 2*t+i+j at right position
skew([0,0,0,0], 1, 3, -1);
densify([0,0,0,0], 1);
interchange([0,0,0,0], 1, 2, 0);   place t+i at right position
skew([0,0,0,0], 1, 2, -1);         find t
```

- Our recovery (3 transformations)

```
Reshape([0], 5, 1, 1);
Reshape([0], 5, 0, 2);
Reshape([0], 3, 0, 1);
```

- Reshape([0], 5, 1, 1): $\phi_{S_0}^{0} = [0 \quad t \quad 0 \quad i \quad 0 \quad i+j \quad 0]$
- Reshape([0], 5, 0, 2): $\phi_{S_0}^{1} = [0 \quad t \quad 0 \quad i \quad 0 \quad 2*t+i+j \quad 0]$
- Reshape([0], 3, 0, 1): $\phi_{S_0}^{2} = [0 \quad t \quad 0 \quad t+i \quad 0 \quad 2*t+i+j \quad 0]$

- Chlore Result schedule:

$$\phi_{S_0}^{chlore} = [0 \quad \phi_{S_0}^{chlore}[4]-i \quad 0 \quad t+i \quad 0 \quad 2*t+i+j \quad 0]$$

HUAWEI

Experiments

Result: Number of transformations recover by Chlore and our tool

- Polybench cases
- Source schedule = initial code
- Target schedule = schedule found with PolyTOPS schedule

Name	Chlore	Our approach
correlation	NA	45
covariance	NA	21
2mm	16	7
3mm	22	13
bicg	14	10
cholesky	13	9
doitgen	1	5
durbin	40	28
gemm	3	3
gemver	7	6
gesummv	7	5
gramschmidt*	29	13
lu	2	4

Name	Chlore	Our approach
mvt	2	2
symm	3	5
syr2k	3	3
syrk	3	3
trisolv	9	5
trmm	3	2
floyd-warshall	0	0
fdtd-2d	41	15
heat-3d	38	12
jacobi-1d	16	4
jacobi-2d	26	8
seidel-2d	12	3

G. Consolaro et al, PolyTOPS: Reconfigurable and Flexible Polyhedral Scheduler, *CGO'24 [Accepted]*

Conclusion and Future Work

- Describe an algorithm to recover transformation primitive that are applied between two polyhedral schedules

- Show that a restricted numbers of primitives is required for recovery

- Show that the set of recover primitive can be heavily reduce comparing with existing tool

- Extend α transformation primitive and α recovery with *stripmine* transformation

- Convert our transformation primitives to primitives for others tools

Thank you.

Bring digital to every person, home and organization for a fully connected, intelligent world.