Manipulate visualizations, not codes!

Oleksandr Zinenko, Cédric Bastoul, Stéphane Huot
<firstname.lastname@inria.fr>

5th International Workshop on Polyhedral Compilation Techniques
Amsterdam, January 19, 2015

Polyhedral Optimization *with** Polyhedra

*and that is the point of this talk.

What is easier to manipulate?

```
#pragma omp parallel for private(lbv,ubv,t3,t4)
 for (t2=lbp;t2<=ubp;t2++) {
    a_r[t2] = 0;;
  }
  if (M >= 1) {
    lbp=0;
    ubp=N-1;
#pragma omp parallel for private(lbv,ubv,t3,t4)
 for (t2=lbp;t2<=ubp;t2++) {
      for (t3=0;t3<=M-1;t3++) {
        a_r[t2] = s_r[t3] * m_r[t2][t3] - s_i[t3] * m_i[t2][t3];;
      }
    }
  }
  lbp=0;
  ubp=N-1;
#pragma omp parallel for private(lbv,ubv,t3,t4)
 for (t2=lbp;t2<=ubp;t2++) {
    a_i[t2] = 0;;
  }
  if (M >= 1) {
    lbp=0;
    ubp=N-1;
#pragma omp parallel for private(lbv,ubv,t3,t4)
 for (t2=lbp;t2<=ubp;t2++) {
      for (t3=0;t3<=M-1;t3++) {
        a_i[t2] = s_i[t3] * m_r[t2][t3] + s_r[t3] * m_i[t2][t3];;
      }
    }
  }
  for (t2=0;t2<=N-1;t2++) {
    val = a_r[t2] * a_r[t2] + a_i[t2] * a_i[t2];;
    t = (val >= t_val)? (t_val = val, t2) : t;;
  }
}
```

$$\mathcal{D}_S(N) = \left\{ () \to \begin{pmatrix} i \\ j \end{pmatrix} \in \mathbb{Z}^2 \,\middle|\, \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \\ N \\ 1 \end{pmatrix} \geq \vec{0} \right\}$$

$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \to \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \end{pmatrix} \in \mathbb{Z}^2 \times \mathbb{Z}^5 \,\middle|\, \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \\ i \\ j \\ N \\ 1 \end{pmatrix} = \vec{0} \right\}$$

$$\mathcal{A}_{S,1}(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \to \begin{pmatrix} a_{S,1} \end{pmatrix} \in \mathbb{Z}^2 \times \mathbb{Z} \,\middle|\, \begin{bmatrix} -1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} a_{S,1} \\ i \\ j \\ N \\ 1 \end{pmatrix} = \vec{0} \right\}$$

Neither?

Challenges in Loop Optimization

- Find and apply pertinent transformation.
- Specify transformation target.
- Combine transformations.

Polyhedral Model

- Exact representation for precise analysis.
- Sacrifices code understanding for analysis power.
- The mathematics involved is far from being trivial.

Challenges in Loop Optimization

- Find and apply pertinent transformation —
  hard in code, easy in the polyhedral model.

- Specify transformation target —
  easy in code, tricky in the polyhedral model.

- Combine transformations —
  difficult in both.

Directive-Based Manipulation

Use higher-level abstraction to describe polyhedral loop transformations.

$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \to \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \end{pmatrix} \in \mathbb{Z}^2 \times \mathbb{Z}^5 \;\middle|\; \left[\begin{array}{ccccc|cc|c|c} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{array}\right] \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \\ \hline i \\ j \\ \hline N \\ \hline 1 \end{pmatrix} = \vec{0} \right\}$$

shift(S, i, -3)

$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \to \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \end{pmatrix} \in \mathbb{Z}^2 \times \mathbb{Z}^5 \;\middle|\; \left[\begin{array}{ccccc|cc|c|c} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{3} \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{array}\right] \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \\ \hline i \\ j \\ \hline N \\ \hline 1 \end{pmatrix} = \vec{0} \right\}$$

Challenges

- Find and apply pertinent transformation —
  application is done by polyhedral tools;
  hard to find the transformation.

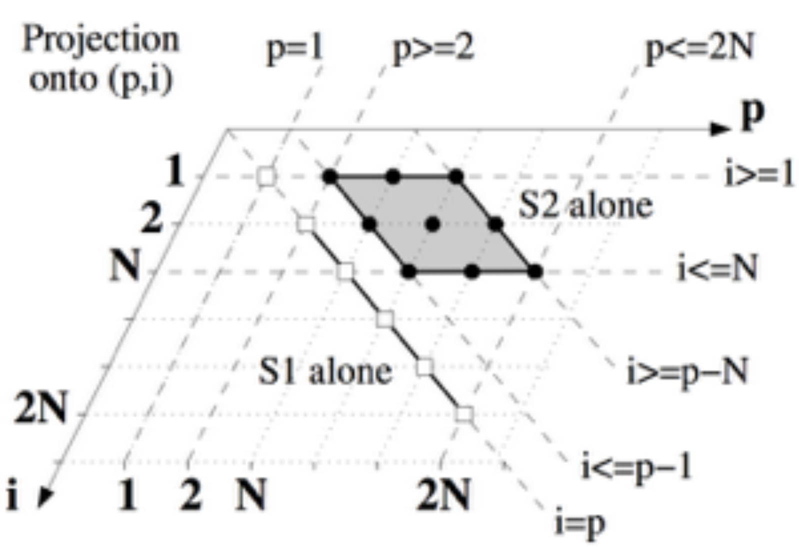
- Specify transformation target —
  non-trivial.

- combine transformations —
  requires support from polyhedral backend.

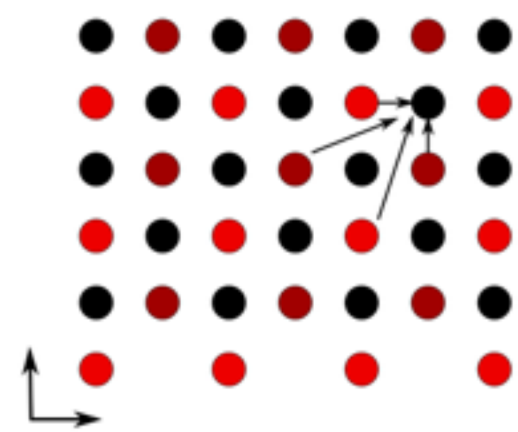
Multiple papers on the polyhedral model resort to visualizations in order to explain it.

Various Polyhedral Visualizations

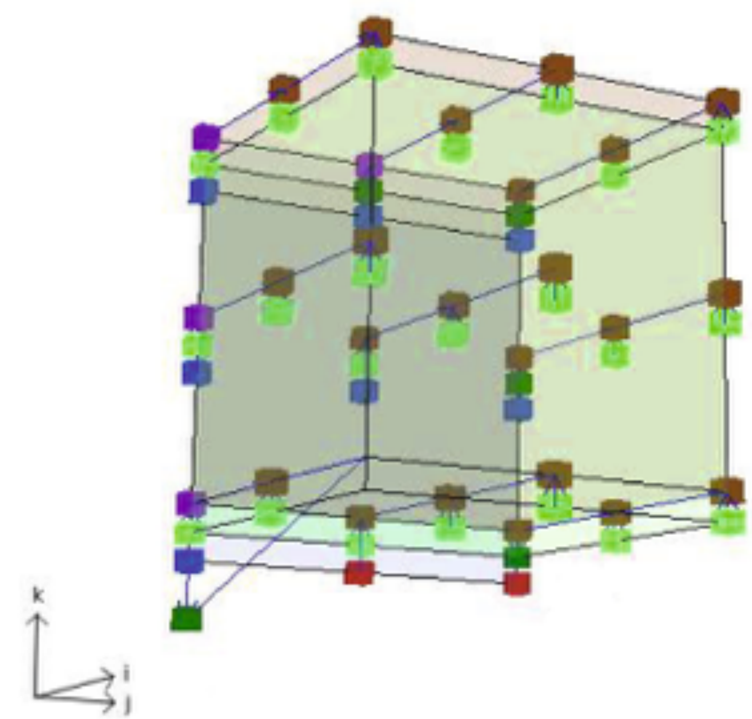
Polyhedral model has a geometrical representation that is extensively used in compilation-related papers.



[Girbal *et.al*, 2006]

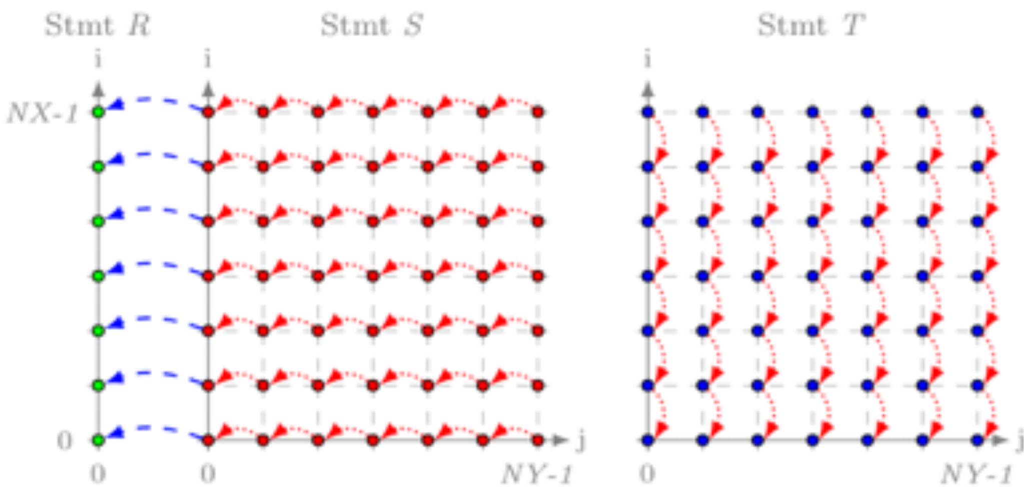


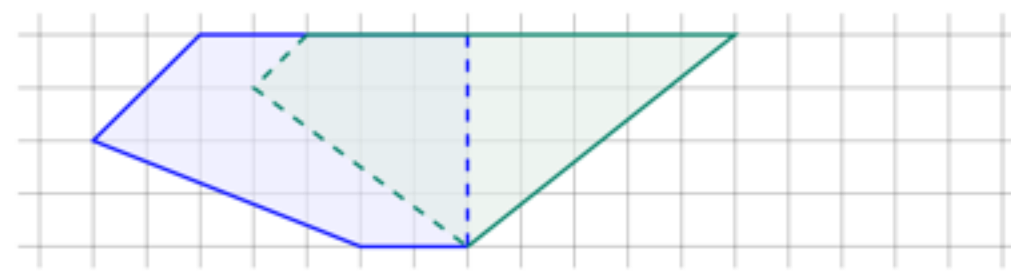
[Fassi *et.al*, 2013]

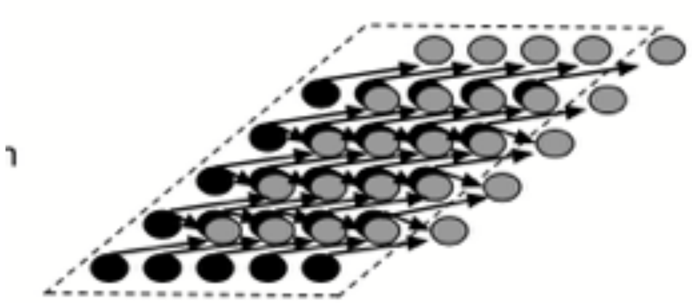


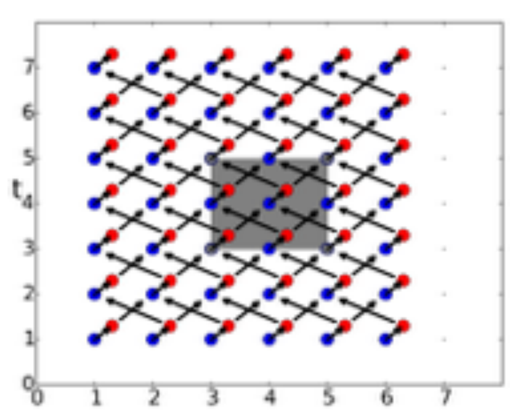
[Wong *et.al*, 2012]

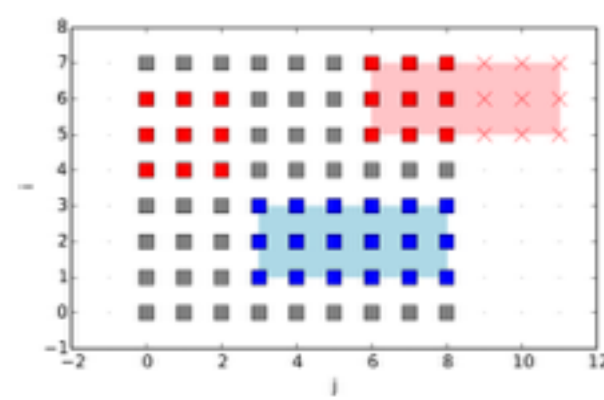
Polyhedral visualizations at IMPACT 2015

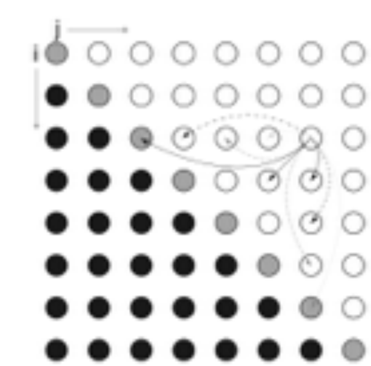












Integer Points Enumeration

$$\mathcal{D}_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \middle| \begin{array}{l} 0 \leq i < N \\ 0 \leq j < N \end{array} \right\}$$

+

$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow \begin{pmatrix} t1 \\ t2 \end{pmatrix} \middle| \begin{array}{l} t1 = i + j \\ t2 = i \end{array} \right\}$$

↓

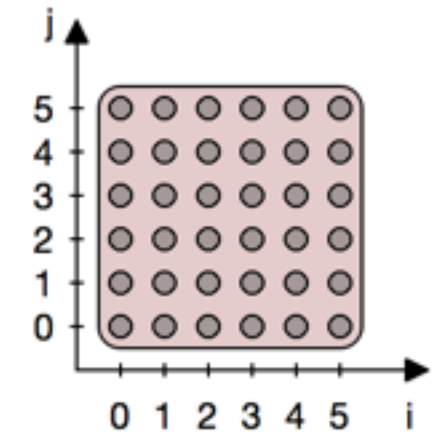
$$\mathcal{T}_S(N) = \left\{ \begin{pmatrix} t1 \\ t2 \end{pmatrix} \middle| \begin{array}{l} 0 \leq t1 - t2 < N \\ 0 \leq t2 < N \end{array} \right\}$$

```
(0, 0),   (0, 1),   (0, 2), ...,  (0, N-1)
(1, 0),   (1, 1),   (1, 2), ...,  (1, N-1)
(2, 0),   (2, 1),   (2, 2), ...,  (2, N-1)
...
(N-1, 0), (N-1, 1), (N-1, 2), ...,(N, N-1)
```

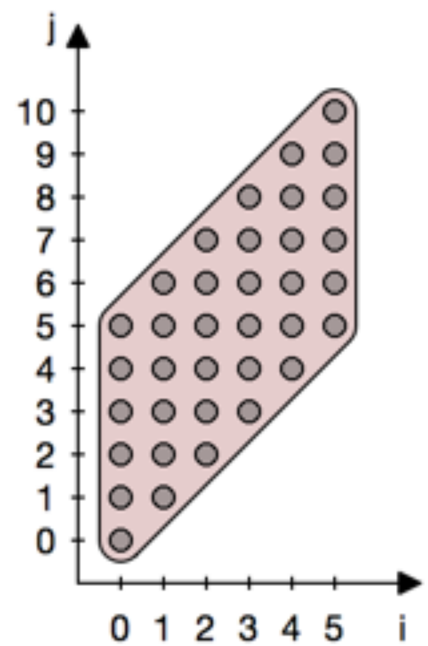
```
(0, 0),   (0, 1),   (0, 2), ...,  (0, N-1)
(1, 1),   (1, 2),   (1, 3), ...,  (1, N)
(2, 2),   (2, 3),   (2, 4), ...,  (2, N+1)
...,
(N-1, 3), (N-1, 4), (N-1, 5), ...,(N, N+N-2)
```

Scheduled iteration domain +
parameters replaced with constants.

Visualizing Domains




```
(0, 0),   (0, 1),   (0, 2), ...,  (0, N-1)
(1, 0),   (1, 1),   (1, 2), ...,  (1, N-1)
(2, 0),   (2, 1),   (2, 2), ...,  (2, N-1)
...
(N-1, 0), (N-1, 1), (N-1, 2), ...,(N, N-1)
```

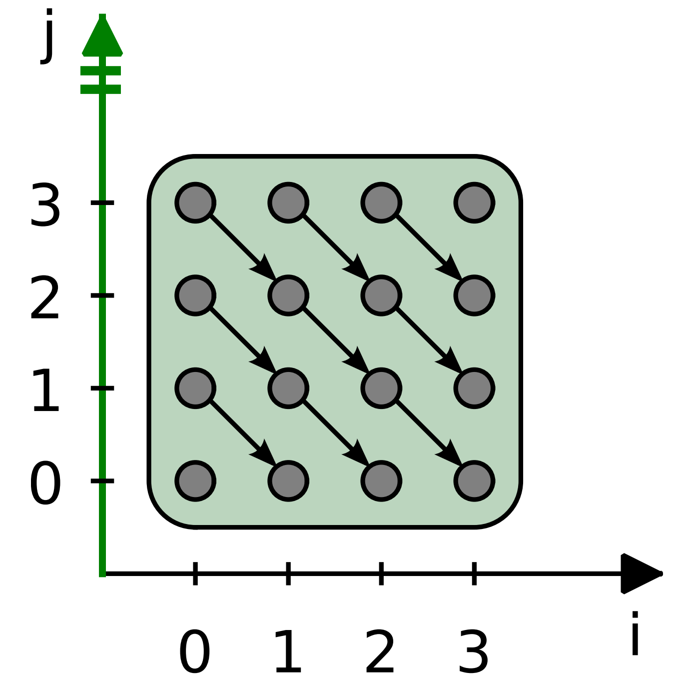



```
(0, 0),   (0, 1),   (0, 2), ...,  (0, N-1)
(1, 1),   (1, 2),   (1, 3), ...,  (1, N)
(2, 2),   (2, 3),   (2, 4), ...,  (2, N+1)
...,
(N-1, 3), (N-1, 4), (N-1, 5), ...,(N, N+N-2)
```

Visualizing Dependences

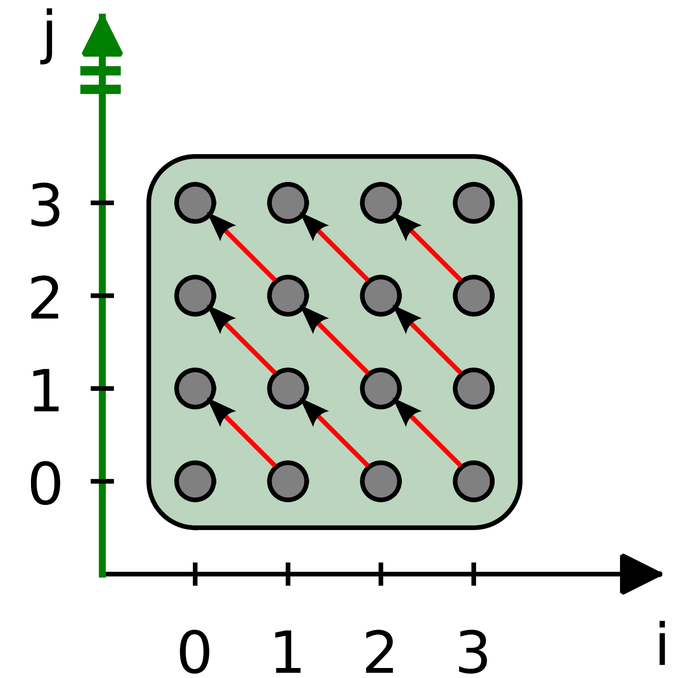
```
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++)
    Z[i+j] += X[i] * Y[j];
```

```
(0, 0) -> (0)
(0, 1); (1, 0) -> (1)
(0, 2); (1, 1); (2, 0) -> (2)
(0, 3); (1, 2); (2, 1); (3, 0) -> (3)
```



Oops… Dependence violation

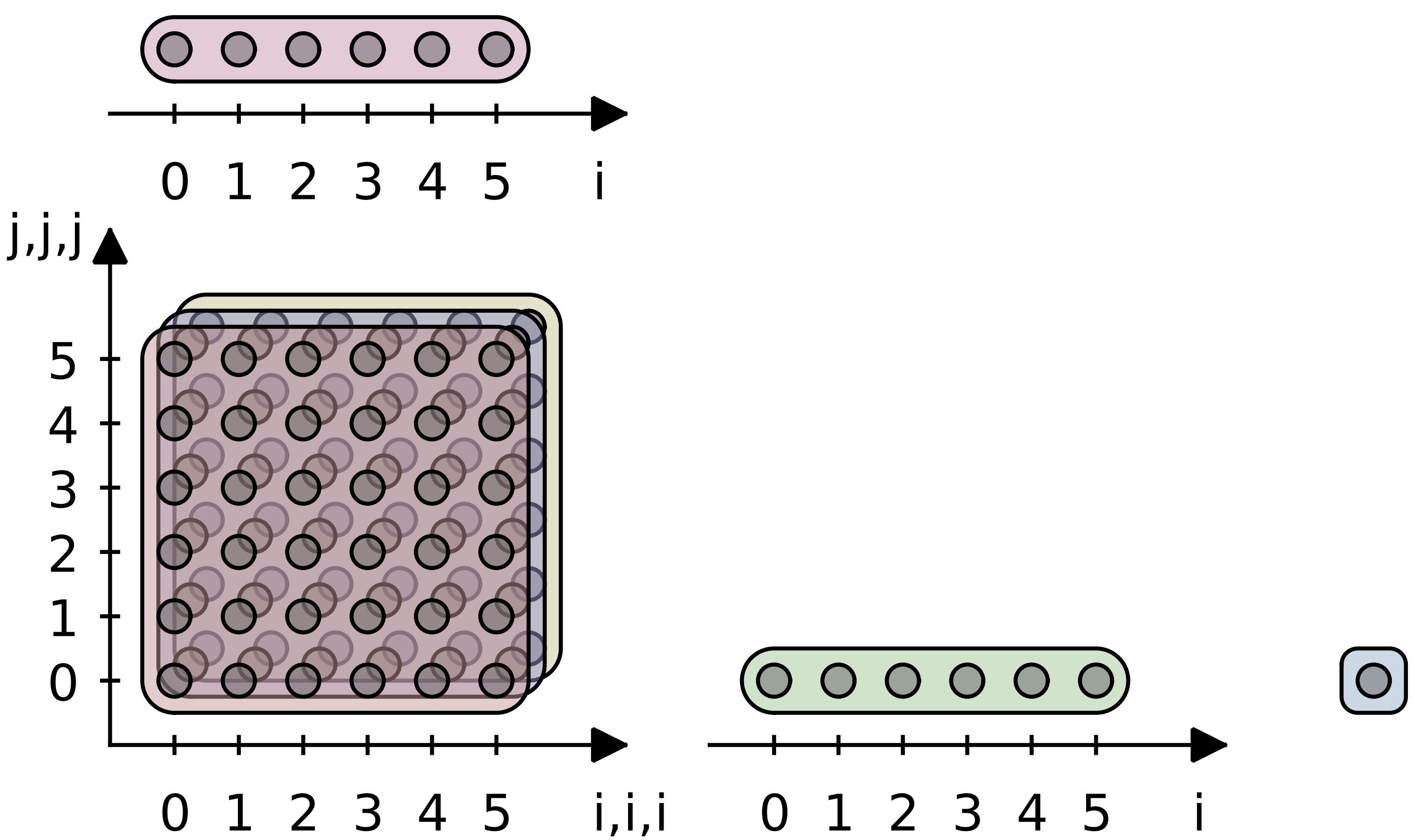
```
for (i = N-1; i >= 0; i--)
  for (j = N-1; j >= 0; j--)
    Z[i+j] += X[i] * Y[j];
```




```
(0, 0) -> (0)
(0, 1); (1, 0) -> (1)
(0, 2); (1, 1); (2, 0) -> (2)
(0, 3); (1, 2); (2, 1); (3, 0) -> (3)
```

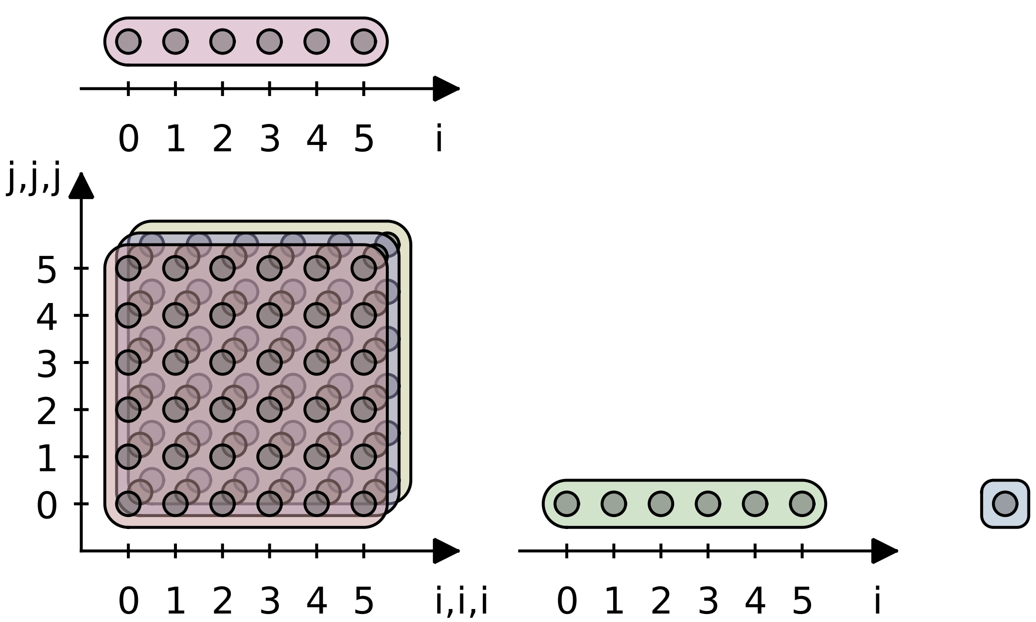
Visualizing Polyhedra

Statements share loop => polyhedra share axis.



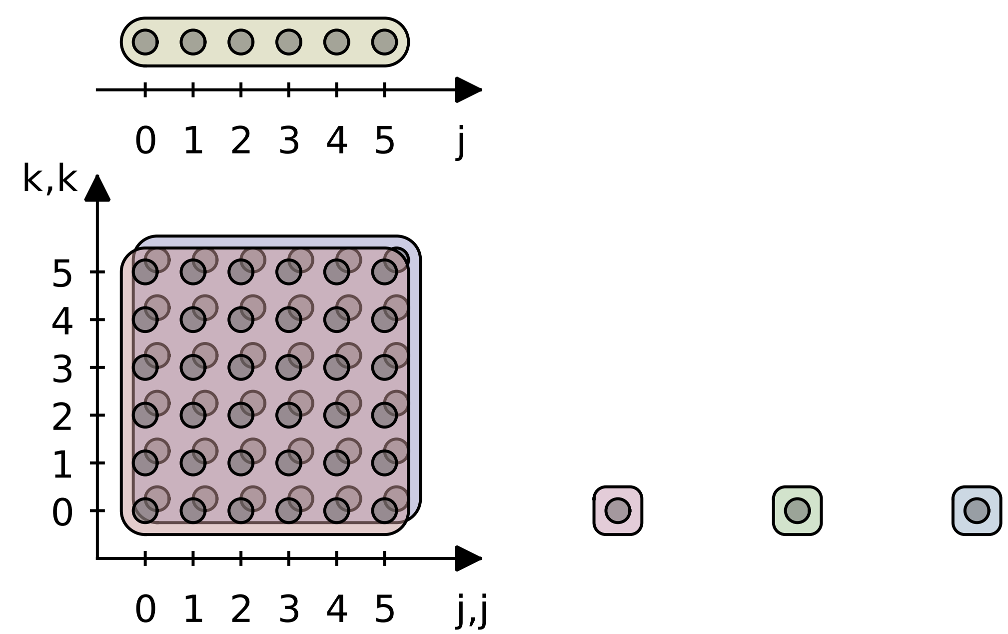

```
for (i=0;i<=N-1;i++) {
  for (j=0;j<=N-1;j++) {
    for (k=0;k<=N-1;k++) {
      S1(i,j,k);
      S2(i,j,k);
    }
    S3(i,j);
  }
  S4(i);
}
for (i=0;i<=N-1;i++) {
  S5(i);
}
S6;
```

Visualizing Polyhedra



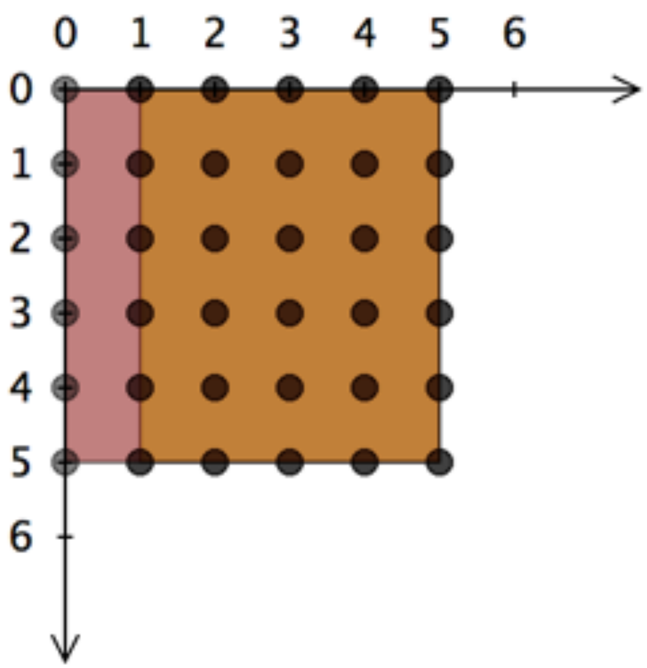
Projection on (i,j)

```
for (i=0;i<=N-1;i++) {
  for (j=0;j<=N-1;j++) {
    for (k=0;k<=N-1;k++) {
      S1(i,j,k);
      S2(i,j,k);
    }
    S3(i,j);
  }
  S4(i);
}
for (i=0;i<=N-1;i++) {
  S5(i);
}
S6;
```

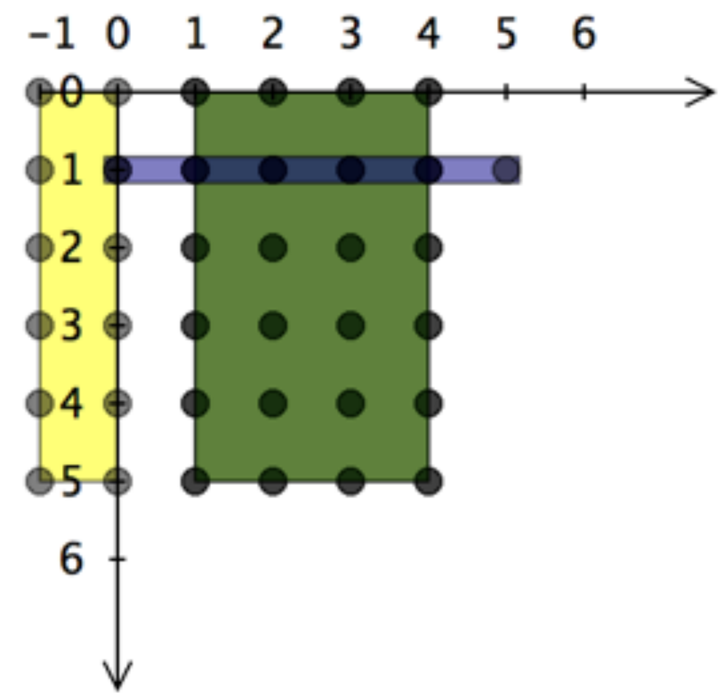


Projection on (j,k)

Visualizing Polyhedra



shift({S2}, left, 1)




```
for (i = 0; i < 6; i++)
  for (j = 0; j < 6; j++) {
    if (i >= 1) S1(i,j);
    S2(i,j);
  }
```

```
for (i = -1; i < 1; i++)
  for (j = 0; j < 6; j++)
    S2(i+1,j);
for (i = 1; i < 5; i++)
  for (j = 0; j < 6; j++) {
    S1(i, j)
    S2(i, j)
  }
for (i = 5; i < 6; i++)
  for (j = 0; j < 6; j++)
    S1(i, j);
```

Challenges

- Find and apply pertinent transformation —
  easier to find; still hard to apply.

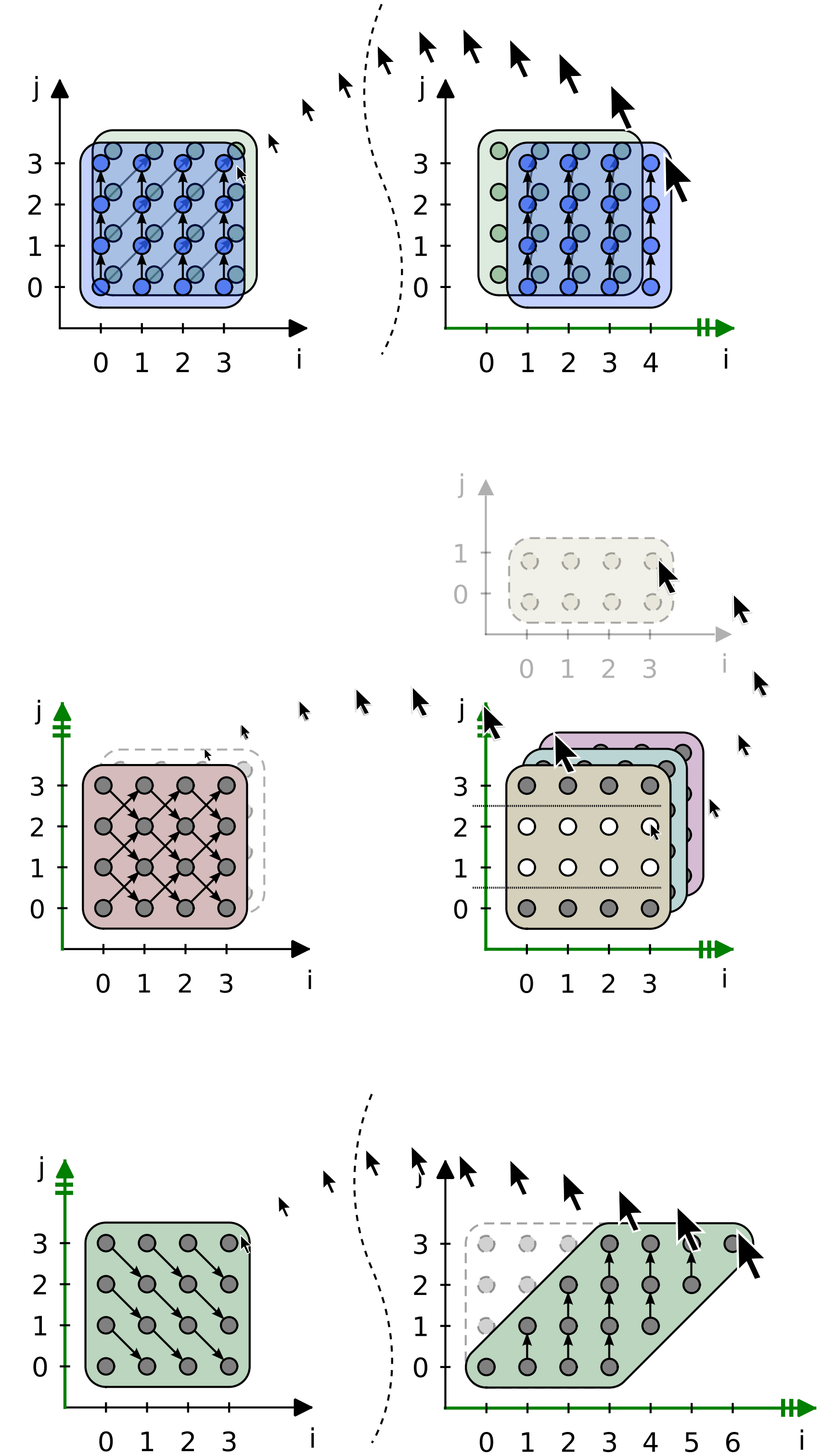
- Specify transformation target —
  may be visualized.

- Combine transformations —
  even harder than in the code.

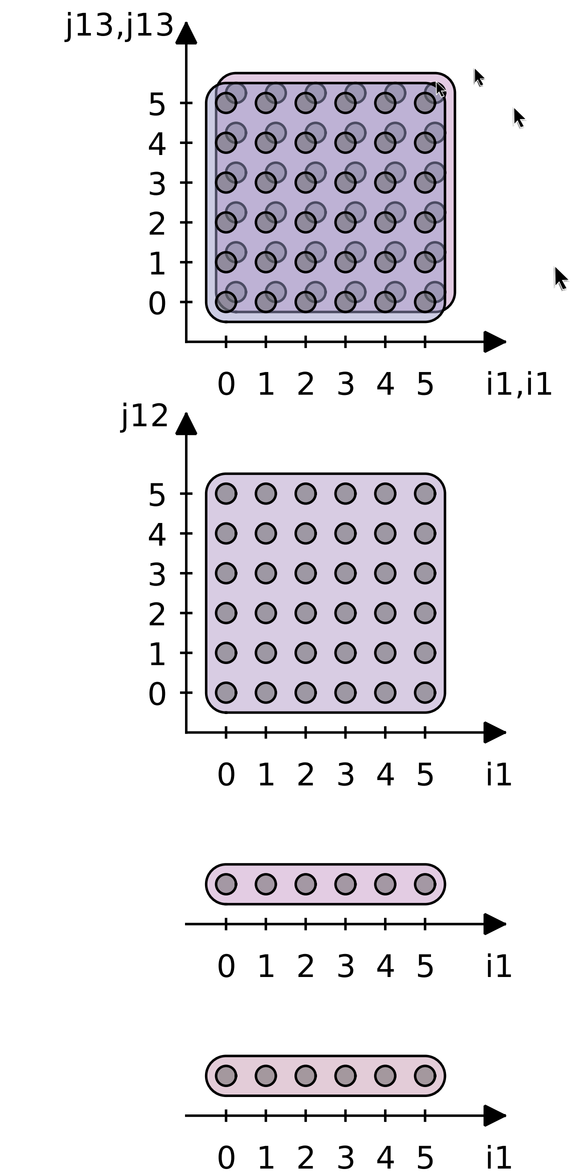
If we already have polyhedra visualized,
we can manipulate visualizations instead of the code!

Available Transformations

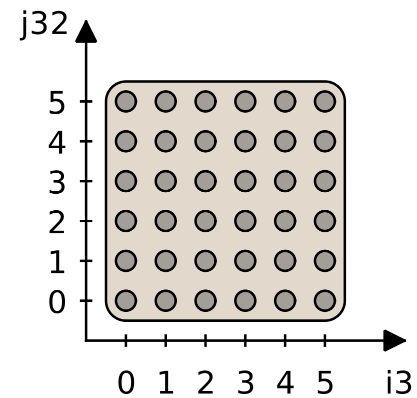
- Loop shifting.
- Loop splitting.
- Loop fusion.
- Statement reordering.
- Loop peeling.
- Loop skewing.

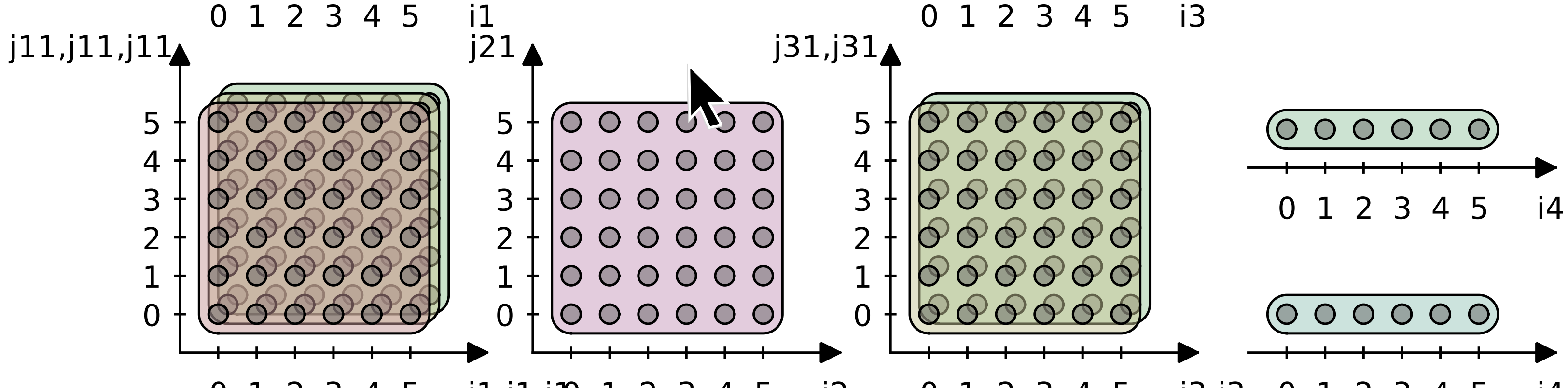


Combining Transformations




```
reorder([0, 4], [1, 0]);
split([0, 4, 0], 2);
split([0, 4], 1);
reorder([], [0, 4, 1, 2, 3, 5, 6]);
fuse([3]);
fuse([3, 0]);
```

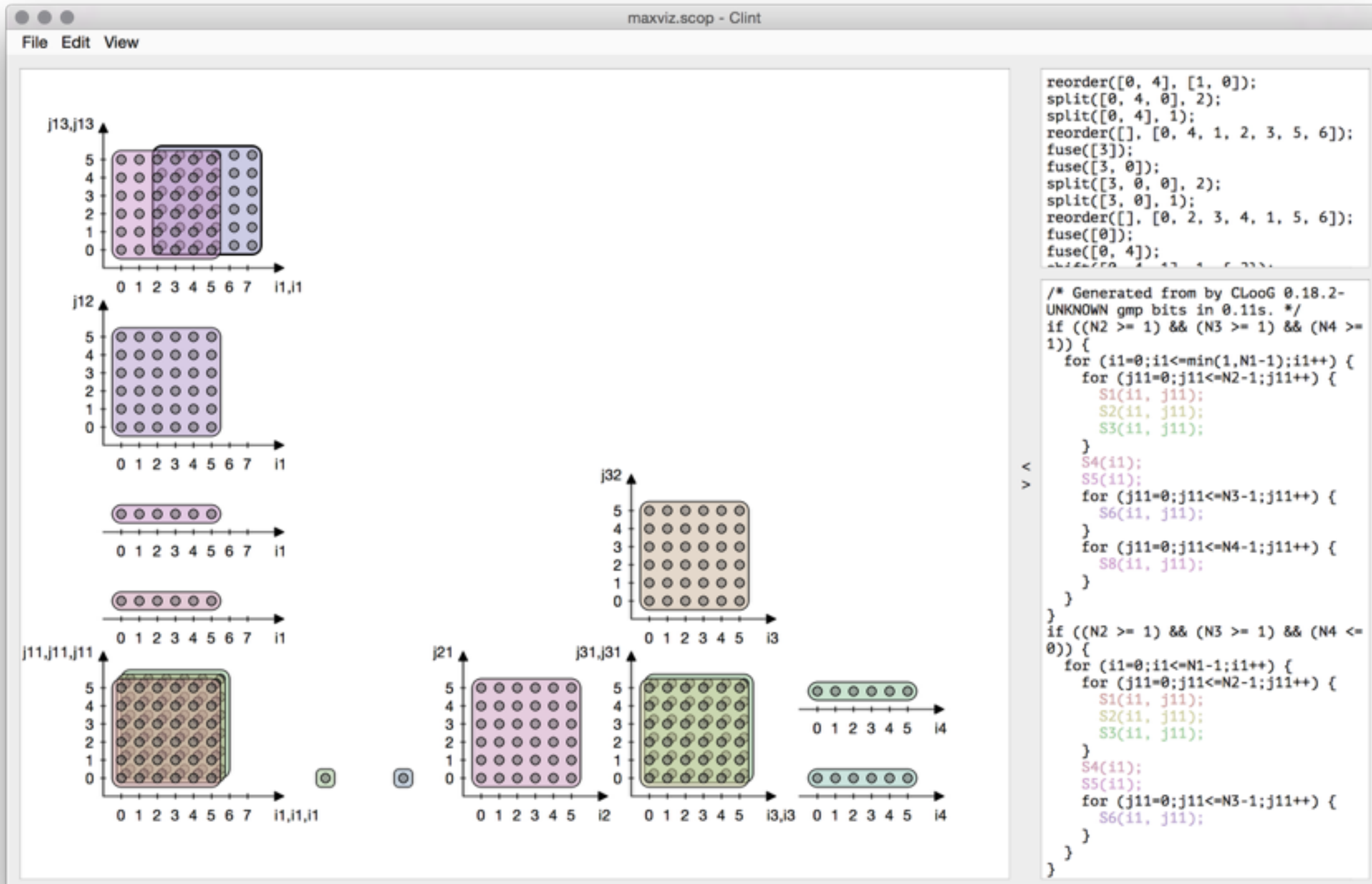




Solutions to Challenges

- Find and apply pertinent transformation —
  as easy as manipulating visual objects.
- Specify transformation target —
  directly select it on the screen.
- Combine transformations —
  as long as the structure is managed.

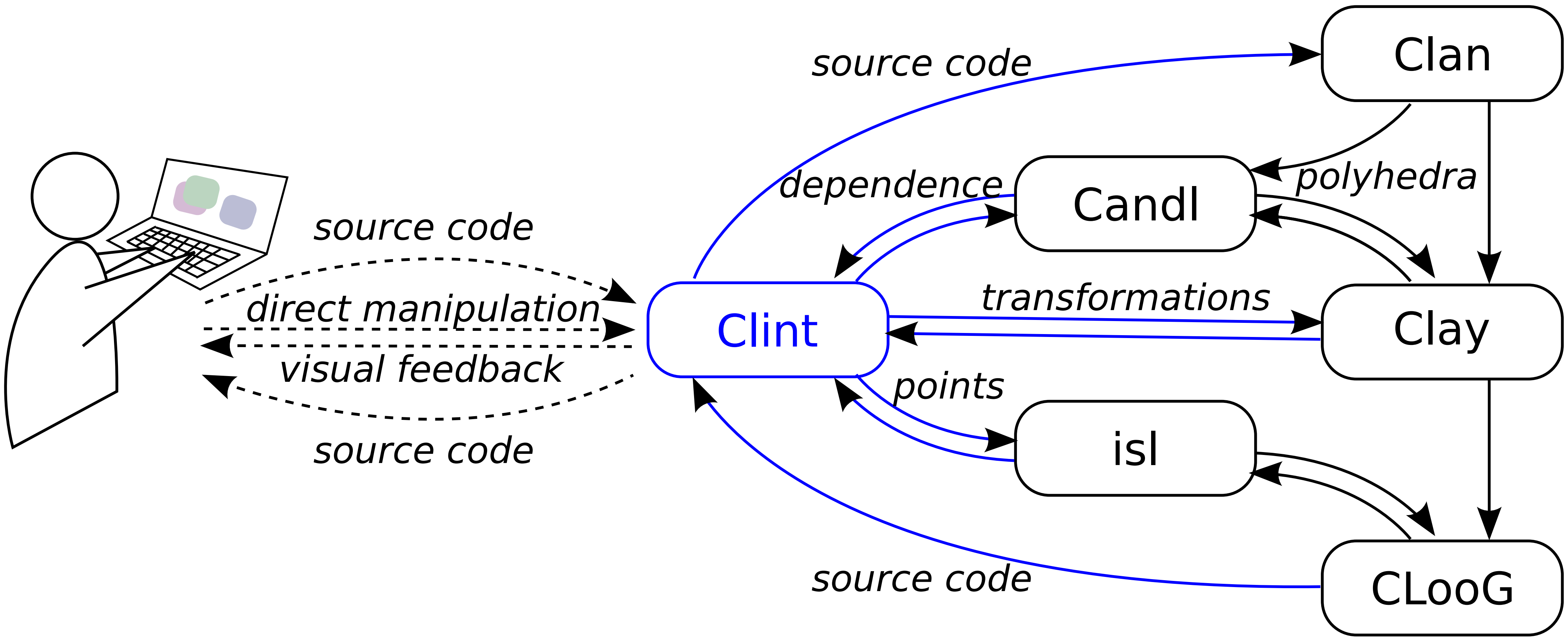
Clint Interactive Visualization



Clint Live Demonstration

./clint --nosegfault,please impact_demo.scop

Clint Architecture



Union of Domain Relations

```
for (i = 0; i < N; i++)
  for (j = 0; j < i; j++)
    S1(i, j);
```

$$\mathcal{D}_{S1}(\mathrm{N}) = \left\{ () \to \begin{pmatrix} i \\ j \end{pmatrix} \middle| \begin{array}{l} 0 \leq i < \mathrm{N} \\ 0 \leq j < i \end{array} \right\}$$

```
for (i = 0; i < N; i++)
  for (j = 0; j < M; j++)
    if (j < 21 || j > 42)
      S0(i, j);
```

$$\mathcal{D}_{S0}(\mathrm{N}, \mathrm{M}) = \left\{ () \to \begin{pmatrix} i \\ j \end{pmatrix} \middle| \begin{array}{l} 0 \leq i < \mathrm{N} \\ 0 \leq j < \mathrm{M} \\ 0 \leq j < 21 \end{array} \right\} \cup \left\{ () \to \begin{pmatrix} i \\ j \end{pmatrix} \middle| \begin{array}{l} 0 \leq i < \mathrm{N} \\ 0 \leq j < \mathrm{M} \\ 0 \leq j > 42 \end{array} \right\}$$

Each relation in the union defines a convex polyhedron.

Union of Scheduling Relations

```
for (i = 0; i < N; i++)
  S0(i);
```

Transformation

$$\theta_{S0}(\mathrm{N},\mathrm{M}) = \left\{ \left(\; i \; \right) \to \begin{pmatrix} \beta_0 \\ \alpha_1 \\ \beta_1 \end{pmatrix} \middle| \begin{array}{ll} \beta_0 & = 0 \\ \alpha_1 & = i \\ \beta_1 & = 0 \end{array} \right\}$$

Change of
Scheduling Relation Union

```
for (i = 0; i < N; i++) {
  if (i <= M + 19)
    S0(i);
  if (i >= M + 20)
    S0(i);
}
```

$$\theta_{S0}(\mathrm{N},\mathrm{M}) =$$
$$\left\{ \left(\; i \; \right) \to \begin{pmatrix} \beta_0 \\ \alpha_1 \\ \beta_1 \end{pmatrix} \middle| \begin{array}{ll} \beta_0 & = 0 \\ \alpha_1 & = i \\ \beta_1 & = 0 \\ \alpha_1 & \leq M + 19 \end{array} \right\} \cup$$
$$\left\{ \left(\; i \; \right) \to \begin{pmatrix} \beta_0 \\ \alpha_1 \\ \beta_1 \end{pmatrix} \middle| \begin{array}{ll} \beta_0 & = 0 \\ \alpha_1 & = i \\ \beta_1 & = 1 \\ \alpha_1 & \geq M + 20 \end{array} \right\}$$

Clay Transformation Set

Transformation = Change of Scheduling Relation Union

Clay encodes high-level loop transformations as changes to the scheduling relation union.

Clay Transformation Example

$$\theta_{S0}(\mathrm{N}, \mathrm{M}) = \left\{ \left(\; i \; \right) \to \begin{pmatrix} \beta_0 \\ \alpha_1 \\ \beta_1 \end{pmatrix} \middle| \begin{array}{ll} \beta_0 & = 0 \\ \alpha_1 & = i \\ \beta_1 & = 0 \end{array} \right\}$$

$$\textsc{IndexSetSplit}(\vec{\rho},\ \textit{constraint})$$

$$\textit{offset} \leftarrow \max_{m \in \mathcal{T}_{\vec{\rho}}} \left(\beta_m^{\dim(\vec{\rho})} \right);$$

$$\forall \mathcal{T} \in \mathcal{T}_{\vec{\rho}},\quad \mathcal{T} \leftarrow \begin{array}{l} \left(\mathrm{copy}(\mathcal{T}) \cap \textit{constraint} \right) \;\bigcup \\ \left(\begin{array}{l} \mathrm{copy}(\mathcal{T}) \cap \overline{\textit{constraint}}; \\ \beta_{\mathcal{T}}^{\dim(\vec{\rho})} \leftarrow \beta_{\mathcal{T}}^{\dim(\vec{\rho})} + \textit{offset} + 1 \end{array} \right) \end{array}$$

$$\theta_{S0}(\mathrm{N}, \mathrm{M}) =$$

$$\left\{ \left(\; i \; \right) \to \begin{pmatrix} \beta_0 \\ \alpha_1 \\ \beta_1 \end{pmatrix} \middle| \begin{array}{ll} \beta_0 & = 0 \\ \alpha_1 & = i \\ \beta_1 & = 0 \\ \alpha_1 & \leq M + 19 \end{array} \right\} \cup$$

$$\left\{ \left(\; i \; \right) \to \begin{pmatrix} \beta_0 \\ \alpha_1 \\ \beta_1 \end{pmatrix} \middle| \begin{array}{ll} \beta_0 & = 0 \\ \alpha_1 & = i \\ \beta_1 & = 1 \\ \alpha_1 & \geq M + 20 \end{array} \right\}$$

Statement Selection in *Clay*

Odd output dimensions represent
order of statement and loops

$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \end{pmatrix} \in \mathbb{Z}^2 \times \mathbb{Z}^5 \;\middle|\; \left[\begin{array}{ccccc|cc|c|c} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{array}\right] \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \\ \hline i \\ j \\ \hline N \\ \hline 1 \end{pmatrix} = \vec{0} \right\}$$

$t_1=1$

$t_3=0$ → $[1,0,0]$

$t_5=0$

β-vector

Statements that share a loop, have a common β-prefix.

Maintain Structure

- Reorder: change particular elements of the beta-vector without changing their number: [0,0,0];[0,0,1];[0,0,2] -> [0,0,1];[0,0,0];[0,0,2].
- Split / fuse: move elements between positions: [0,0,0];[0,0,1];[0,0,2] -> [0,0,0];[0,0,1];[0,1,0].
- Stripmine: increase the number of elements: [0,0,0];[0,1,0] -> [0,0,0,0];[0,1,0].

Maintain Structure

- Reorder: change order of polygons or groups.
- Split / fuse: group or ungroup polygons.
- Stripmine: create extra projection.

Evaluation on PolyBench

has no clear sense for this tool :

- *Clint* is able to visualize all benches.
- But this gives no information about usefulness, usability or understandability of the approach.

Clint Visualization Evaluated

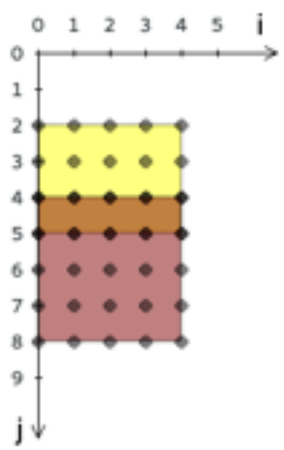
- Goal: verify that the visualization represents all the necessary information in an understandable form.
- Potential users are able to reliably map from the source code to visualization and back.

Clint Visualization Evaluated

- Experiment: make real users perform a specific task in a controlled environment.
- Participants: 6 experts in polyhedral optimization, 10 non-experts (students).
- Varying factors in task: difficulty (easy, medium, hard); mapping direction (code ↔ visualization).

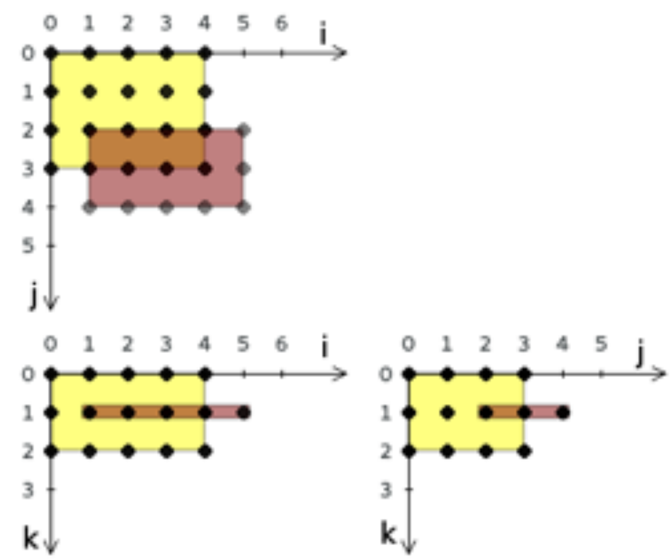
Clint Visualization Evaluated

Easy



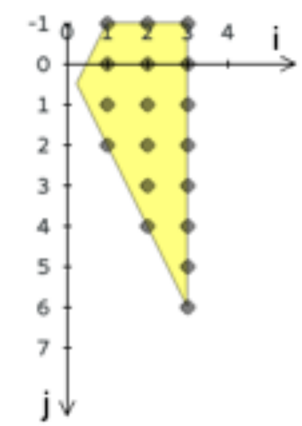

```
for (i = 2; i < 9; i++)
  for (j = 0; j < 5; j++) {
    if (i < 6) S1();
    if (i > 3) S2();
  }
```

Medium



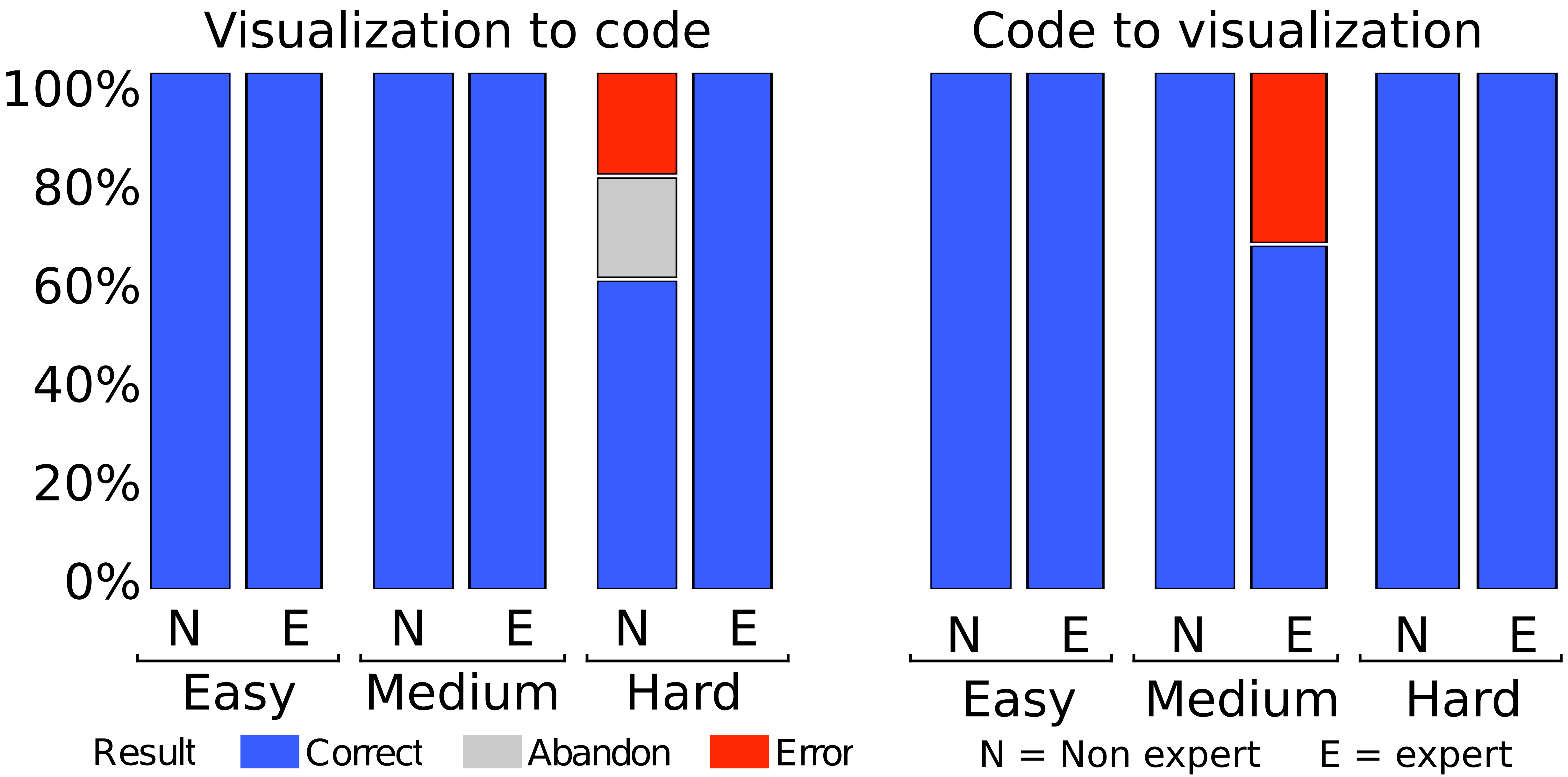

```
for (i = 0; i < 6; i++)
  for (j = 0; j < 5; j++) {
    if (i < 5)
      if (j < 4)
        for (k = 0; k < 3; k++)
          S1(i, j, k);
    if (i > 0)
      if (j > 1)
        S2(i, j);
  }
```

Hard




```
for (i = 0; i <= 3; i++)
  for (j = 1; j <= 2*i; j++)
    if (2*i + j >= 1)
      S(i, j);
```

Clint Visualization Evaluated

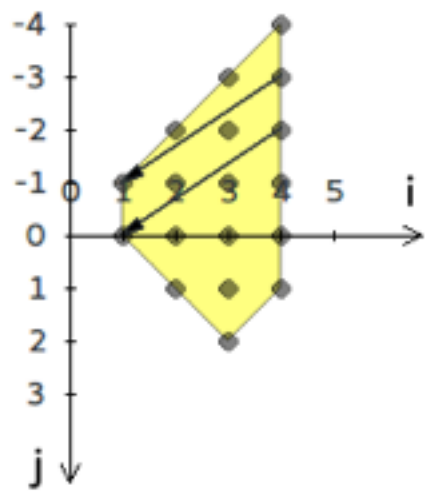


Clint Manipulation Evaluated

- Goal: explore the benefits and drawbacks of interactive visualization.
- Task: restructure code to expose parallelism.
- Participants: 8 participants from the previous experiment.
- Factors: available representations (code, visualization, both); difficulty (3 levels).

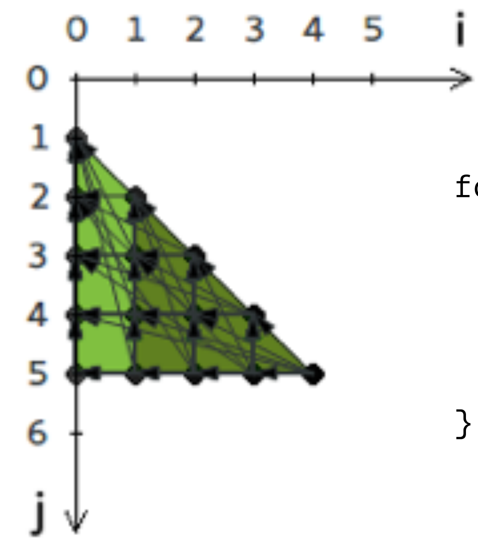
Clint Manipulation Evaluated

Easy

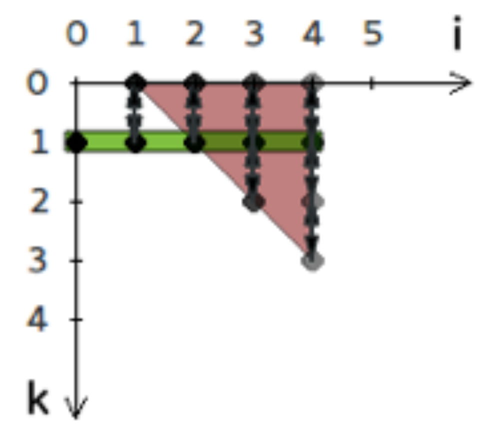


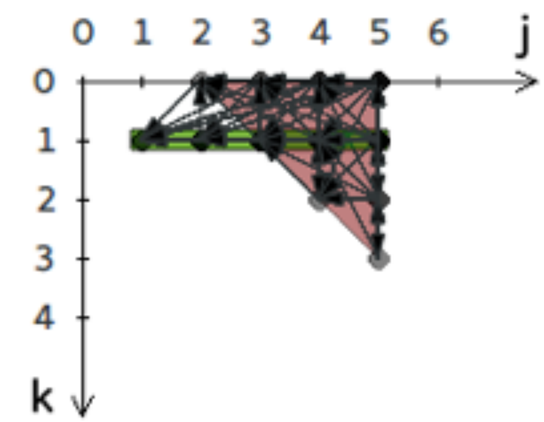

```
for (i = 0; i < N + 1; i++)
  for (j =-i; j < i; j++)
    if (i+j-N-1<=0)
      z[2*i + 3*j] +=
      A[i][j] *B[i][i] *m[j];
```

Hard

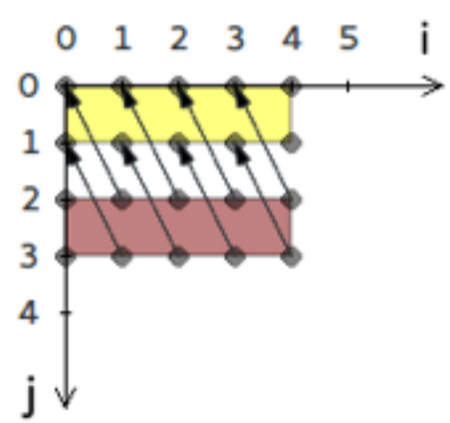



```
for (i = 0; i < N ; i++)
  for (j = i + 1; j < N; j++) {
    s[i]=0;
    for (k = 0; k < i; k++)
      s[i] += L[i][k] * L[j][k];
    L[j][k] = L[i][i] * A[j][i] - s[i];
}
```



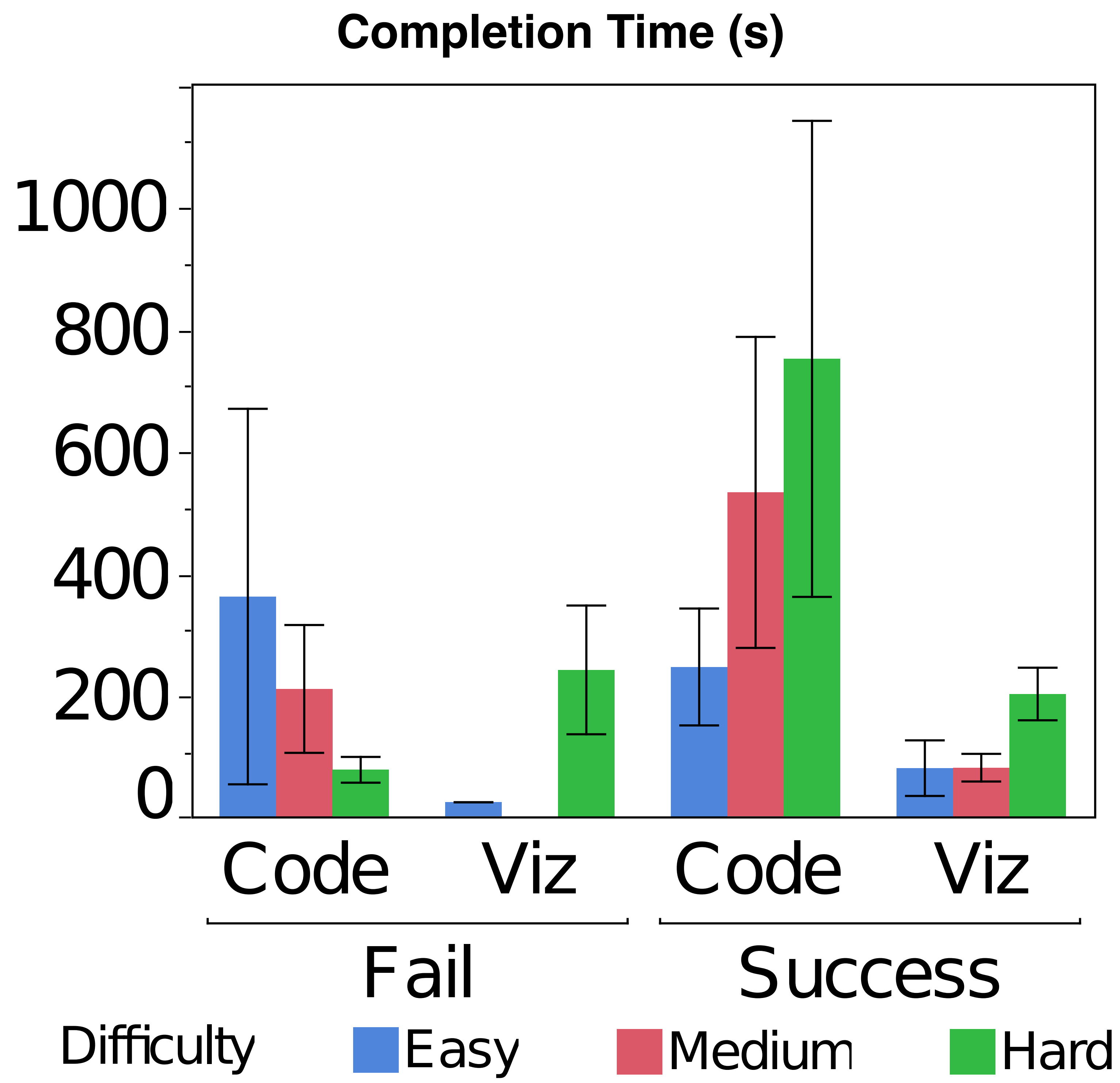


Medium

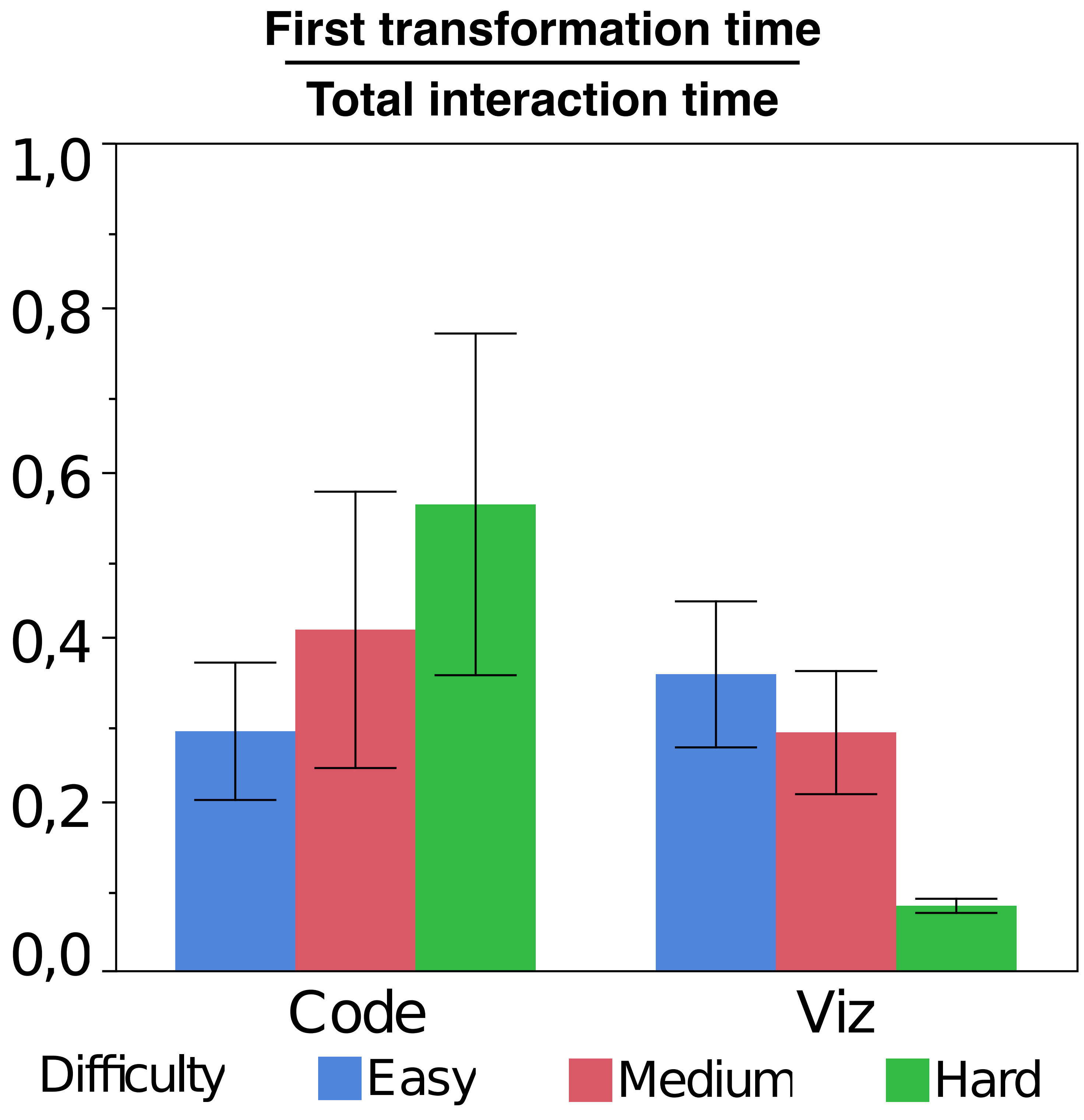



```
for (i = 0; i < 5; i++) {
  for (j = 0; j < 2; j++)
    A[i][j] = init(i, j);
  if (i > 0)
    for (j = 2; j < N; j++)
      A[i - 1][j - 2] +=
      func(P[i][j]);
}
```

Clint Manipulation Evaluated



Clint Manipulation Evaluated



Future Directions

- Integration with performance estimation tools and code editors *via* in-place visualizations.
- Support for data locality visualization and data layout-related transformations.
- User-guided optimization approaches for optimization and refinement.

Conclusion

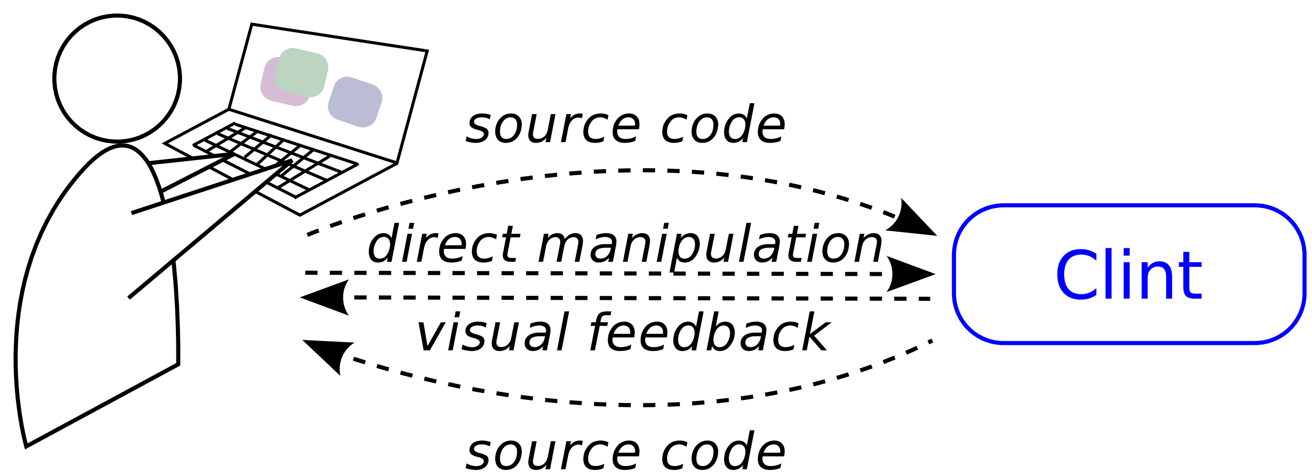
- Visual approach may favor reasoning in terms of instances and dependences rather than statements and loops.

- Expose power and complexity of the polyhedral model in a manageable way.

Questions?

Manipulate polyhedra, not codes!

Oleksandr Zinenko, Cédric Bastoul, Stéphane Huot
<firstname.lastname@inria.fr>



Clint available soon on *https://www.lri.fr/~zinenko/clint*

Clint: why the name?

- Tribute to polyhedral libraries that have CL in their names for a historical reason (CLooG, Clan, Clay).
- Chunky Loop INTeraction.

Why not Compare Against *Pluto*?

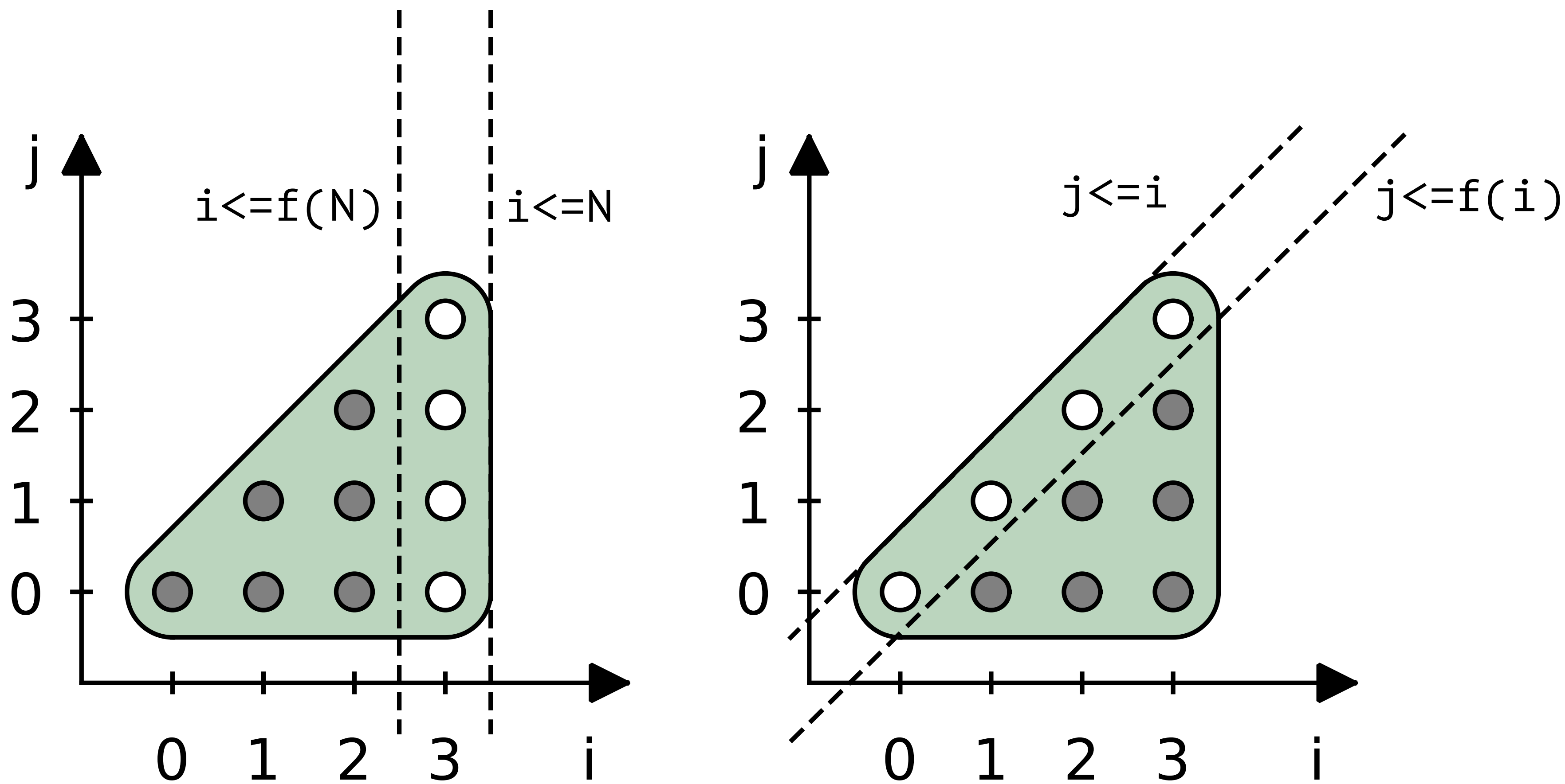
Clint has a different application area, complementary to that of an automatic optimizer.

For example helping to develop and debug optimizations and optimizers.

Why not a Tool for Teaching?

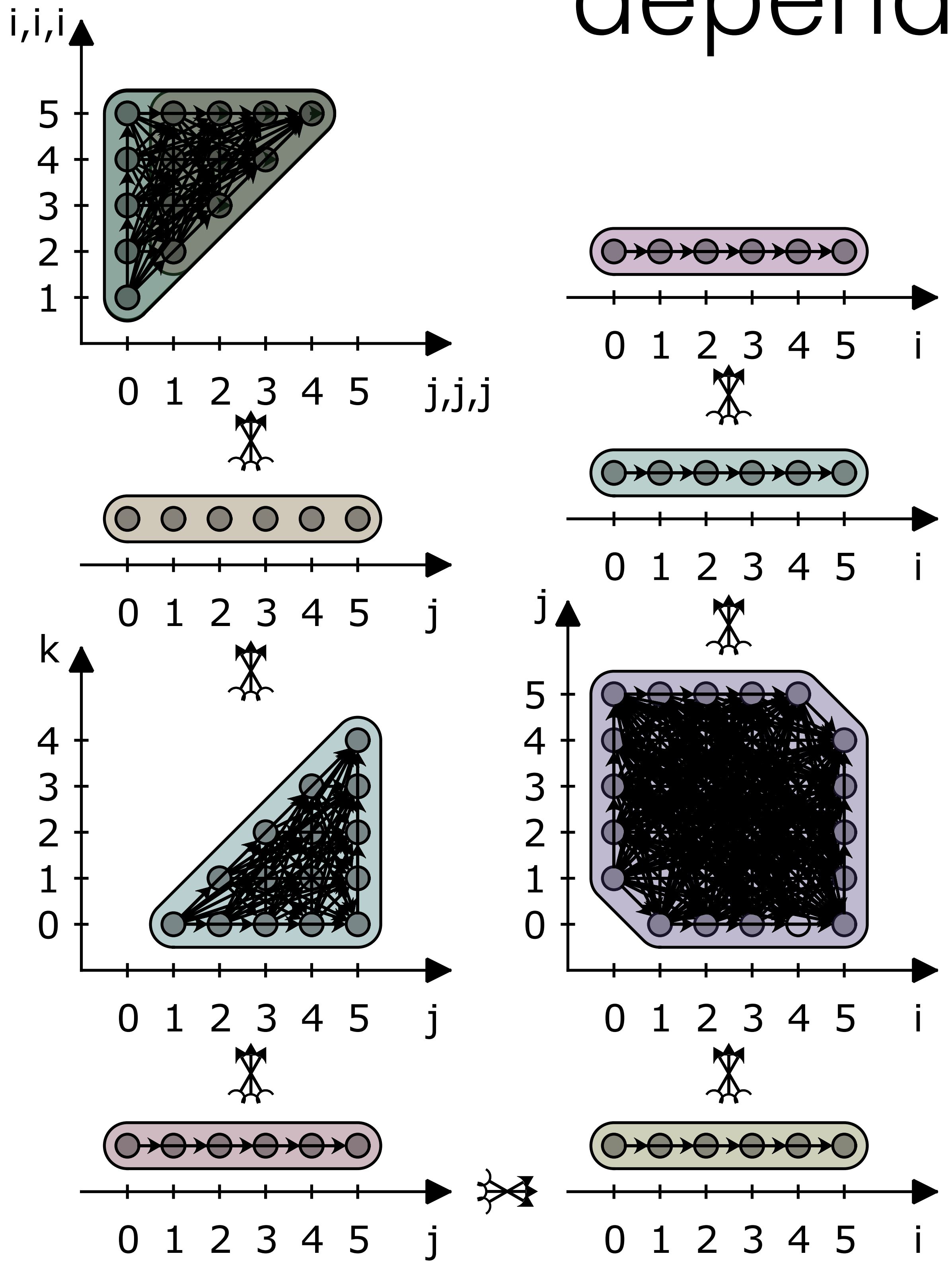
```
A++;
B++;
for (int i = 0; i < 4; i++)
    C[i]++;
for (int i = 0; i < 4; i++)
    for (int j = 0; j < 6; j++)
        D[i][j]++;
```

How does *Clint* manage parametric transformations?



Find linear condition in the same form as the closest boundary.

What if there are many dependences?




```
for (int j = 0; j < n; j++) {
  s = 0;
  for (int k = 0; k < j; k++) {
    s += L[j][k] * L[j][k];
  }
  L[j][j] = sqrt(A[j][j] - s);
  for (int i = j+1; i < n; i++) {
    s = 0;
    for (int k = 0; k < j; k++) {
      s += L[i][k] * L[j][k];
    }
    L[i][j] = (1.0 / L[j][j] * (A[i][j] - s));
  }
}

for (int i = 0; i < n; i++) {
  m = 0;
  for (int j = 0; j < n; j++) {
    if (i > j || i < j) {
      m = m + L[i][j] * L[i][j];
    }
  }
  m = m - L[i][i] * L[i][i];
  t = (m > mmax) ?
       (mmax = m, i) : t;
}
```

What if there are many dependences?