Extending Pluto-Style Polyhedral Scheduling with Consecutivity

Sven Verdoolaege\textsuperscript{1}  Alexandre Isoard\textsuperscript{2}

\textsuperscript{1}KU Leuven and Polly Labs
\textsuperscript{2}Xilinx

January 23, 2018
Outline

1 Introduction
   - Consecutivity Concept
   - Pluto-Style Polyhedral Scheduling
   - Consecutivity Criterion
   - Related Work

2 Intra-Statement Consecutivity
   - Consecutivity Criterion
   - Specifying Schedule Constraints
   - Transformation to Constraints on Schedule Coefficients
   - Solving Constraints on Schedule Coefficients (isl)

3 Inter-Statement Consecutivity

4 Local Rescheduling

5 Conclusions and Future Work
Outline

1. Introduction
   - Consecutivity Concept
   - Pluto-Style Polyhedral Scheduling
   - Consecutivity Criterion
   - Related Work

2. Intra-Statement Consecutivity
   - Consecutivity Criterion
   - Specifying Schedule Constraints
   - Transformation to Constraints on Schedule Coefficients
   - Solving Constraints on Schedule Coefficients (isl)

3. Inter-Statement Consecutivity

4. Local Rescheduling

5. Conclusions and Future Work
Consecutivity Concept

- **Temporal Locality**
  Consecutive operations access the same memory element
  \[\Rightarrow\] reuse of data in cache or registers
Consecutivity Concept

Spatial Locality
Consecutive operations access neighboring memory elements
⇒ reuse of cache lines

Temporal Locality
Consecutive operations access the same memory element
⇒ reuse of data in cache or registers
Consecutivity Concept

Spatial Locality
Consecutive operations access neighboring memory elements
⇒ reuse of cache lines

Temporal Locality
Consecutive operations access the same memory element
⇒ reuse of data in cache or registers

Consecutivity
Consecutive operations access consecutive memory elements
⇒ vectorization
⇒ hardware cache prefetcher
⇒ burst accesses, e.g., on FPGA (Xilinx)
Consecutivity Concept

Spatial Locality
Consecutive operations access neighboring memory elements
⇒ reuse of cache lines

Temporal Locality
Consecutive operations access the same memory element
⇒ reuse of data in cache or registers

Consecutivity
Consecutive operations access consecutive memory elements
⇒ vectorization
⇒ hardware cache prefetcher
⇒ burst accesses, e.g., on FPGA (Xilinx)
Burst Accesses (Sketch)

```c
for (int i = 0; i < N; ++i) {
    for (int j = 0; j < M; ++j) {
        C[j][i] = A[i] * B[j];
    }
}
```
Burst Accesses (Sketch)

```c
AA = burst_read_start(A, N);
for (int i = 0; i < N; ++i) {
    BB = burst_read_start(B, M);
    for (int j = 0; j < M; ++j) {
        C[j][i] = burst_read_iter(AA, &A[i]) * 
                   burst_read_iter(BB, &B[j]);
    }
    burst_read_end(BB, M);
}
burst_read_end(AA, N);
```
Burst Accesses (Sketch)

\[
\begin{align*}
\text{AA} & = \text{burst\_read\_start}(A, N); \\
\text{for } (\text{int } i = 0; i < N; + + i) \{ \\
\text{BB} & = \text{burst\_read\_start}(B, M); \\
\text{for } (\text{int } j = 0; j < M; + + j) \{ \\
\text{C}[j][i] & = \text{burst\_read\_iter}(\text{AA}, &A[i]) \ast \\
& \text{burst\_read\_iter}(\text{BB}, &B[j]); \\
\} \\
\text{burst\_read\_end}(\text{BB}, M); \\
\} \\
\text{burst\_read\_end}(\text{AA}, N); \\
\end{align*}
\]

No burst accesses on C
Burst Accesses (Sketch)

```c
for (int i = 0; i < N; ++i) {
    for (int j = 0; j < M; ++j) {
        C[j][i] = A[i] * B[j];
    }
}
```

No burst accesses on C
Burst Accesses (Sketch)

```c
for (int j = 0; j < M; ++j) {
    for (int i = 0; i < N; ++i) {
        C[j][i] = A[i] * B[j];
    }
}
```
Burst Accesses (Sketch)

CC = burst_write_start(C, M * N);
BB = burst_read_start(B, M);
for (int j = 0; j < M; ++j) {
    AA = burst_read_start(A, N);
    for (int i = 0; i < N; ++i) {
        burst_write_iter(CC, &C[j][i]) =
        burst_read_iter(AA, &A[i]) *
        burst_read_iter(BB, &B[j]);
    }
    burst_read_end(AA, N);
}
burst_read_end(BB, M);
burst_write_end(CC, M * N);
Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances
- original schedule (if any) derived from input
- target schedule computed by scheduler
Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances
- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model
- instance set: set of schedulable statement instances
- access relations: map instances to memory locations
- dependence relations:
  - pairs of instances that need to be executed in order
  - derived from access relations and original schedule
Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances
- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model

Result (typically):
- multiple (quasi) affine functions on instance set
- hierarchically organized (sequence, tree)

Types:
- Farkas based schedulers (Feautrier 1992)
  ⇒ use Farkas to transform dependences into constraints on schedule coefficients
  - Pluto-style schedulers, e.g., Pluto, isl
    ⇒ compute affine functions one by one

...
Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances
- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model

Result (typically):
- multiple (quasi) affine functions on instance set
- hierarchically organized (sequence, tree)

Types:
- Farkas based schedulers (Feautrier 1992)
  - use Farkas to transform dependences
  into constraints on schedule coefficients
  - Plato-style schedulers, e.g., Pluto, isl
    - compute affine functions one by one
  - one-shot schedulers (Vasilache 2007)
    - compute entire schedule as a whole

...
Pluto-Style Polyhedral Scheduling

A schedule assigns an execution order to statement instances

- original schedule (if any) derived from input
- target schedule computed by scheduler

A polyhedral scheduler computes schedule using polyhedral model

Result (typically):

- multiple (quasi) affine functions on instance set
- hierarchically organized (sequence, tree)

Types:

- Farkas based schedulers (Feautrier 1992)
  - use Farkas to transform dependences into constraints on schedule coefficients
    - Pluto-style schedulers, e.g., Pluto, isl
      - compute affine functions one by one
    - one-shot schedulers (Vasilache 2007)
      - compute entire schedule as a whole

...
Pluto-Style Polyhedral Scheduling

Main optimization criteria:

- parallelism
- temporal locality
- permutability $\Rightarrow$ tiling
Pluto-Style Polyhedral Scheduling

Main optimization criteria:
- parallelism
- temporal locality
- permutability $\Rightarrow$ tiling

Remaining freedom (if any)
$\Rightarrow$ isl scheduler tends towards lexicographic ordering of instances

Extreme example:

```
for (i=0; i<M; ++i)
  for (j=0; j<N; ++j)
S:  A[i][j] = 0;

S[i,j] → [i,j]
consecutive (by chance)
```

```
for (i=0; i<M; ++i)
  for (j=0; j<N; ++j)
T:  B[j][i] = 0;

T[i,j] → [i,j]
not consecutive
```
Pluto-Style Polyhedral Scheduling

Main optimization criteria:
- parallelism
- temporal locality
- permutability $\Rightarrow$ tiling

Remaining freedom (if any)
$\Rightarrow$ isl scheduler tends towards lexicographic ordering of instances

Extreme example:

```c
for (i=0; i<M; ++i)
    for (j=0; j<N; ++j)
S:   A[i][j] = 0;
    S[i,j] → [i,j]
    consecutive (by chance)

for (i=0; i<M; ++i)
    for (j=0; j<N; ++j)
T:   B[j][i] = 0;
    T[i,j] → [i,j]
    not consecutive
```

Goal: steer towards consecutivity in case of sufficient freedom

Current implementation in isl (roughly):
permutability $>$ parallelism $>$ consecutivity $>$ temporal locality
Consecutivity Criterion

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)
- intra-statement consecutivity (⇒ per statement)
- row-major array layout
- purely affine access function $F$
- purely affine per-statement schedule $T$
Consecutivity Criterion

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)
- intra-statement consecutivity (⇒ per statement)
- row-major array layout
- purely affine access function $F$
- purely affine per-statement schedule $T$

Transformed access function $F T^{-1}$ exhibits consecutivity if
- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

\[
S(x) \xrightarrow{F} \begin{bmatrix} T_1 & T_2 \end{bmatrix} = \begin{bmatrix} M_0 & N_1 \end{bmatrix}
\]

\[
L(i) \xrightarrow{T} A
\]
**Consecutivity Criterion**

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)
- intra-statement consecutivity ($\Rightarrow$ per statement)
- row-major array layout
- purely affine access function $F$
- purely affine per-statement schedule $T$

Transformed access function $FT^{-1}$ exhibits consecutivity if
- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$[\ldots + 0i_n] \ldots [\ldots + 0i_n][\ldots + 1i_n]$$

$$FT^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}$$
Consecutivity Criterion

Consecutive operations access consecutive memory elements

Assume (for the purpose of consecutivity)
- intra-statement consecutivity (⇒ per statement)
- row-major array layout
- purely affine access function $F = [G; H]$
- purely affine per-statement schedule $T = [T_1; T_2]$

Transformed access function $F T^{-1}$ exhibits consecutivity if
- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F T^{-1} = \begin{bmatrix} G & T_1^{-1} \\ H & T_2 \end{bmatrix} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}$$
Consecutivity Criterion Reformulation

Transformed access function $F T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F T^{-1} = \begin{bmatrix} G & T_1 \\ H & T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}$$
Consecutivity Criterion Reformulation

Transformed access function $F T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F T^{-1} = \begin{bmatrix} G & T_1 \end{bmatrix}^{-1} = \begin{bmatrix} M & 0 \\ H & T_2 \end{bmatrix} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}$$

- $G q = 0$

Note: $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & 0 \\ 0^t & 1 \end{bmatrix}$

$\Rightarrow q$ spans ker $T_1$

$\Rightarrow$ ker $T_1 \subseteq$ ker $G$ (Vasilache et al. 2012)

That is, rows of $G$ need to be linear combinations of rows of $T_1$

$G = A T_1$
Consecutivity Criterion and Spatial Locality

Consecutivity

\[ F T^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix} \]

(Kandemir, Ramanujam, and Choudhary 1999)
Consecutivity Criterion and Spatial Locality

Spatial Locality

\[ F T^{-1} = \begin{bmatrix} M & 0 \\ N & x \end{bmatrix} \]

- Consecutivity

\[ F T^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix} \]

(Kandemir, Ramanujam, and Choudhary 1999)
Consecutivity Criterion and Spatial Locality

Spatial Locality

$$FT^{-1} = \begin{bmatrix} M & 0 \\ N & x \end{bmatrix}$$

- Temporal Locality

$$FT^{-1} = \begin{bmatrix} M & 0 \\ N & 0 \end{bmatrix}$$

- Consecutivity

$$FT^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}$$

(Kandemir, Ramanujam, and Choudhary 1999)
Consecutivity Criterion and Spatial Locality

Spatial Locality

\[ F T^{-1} = \begin{bmatrix} M & 0 \\ N & x \end{bmatrix} \]

- Temporal Locality

\[ F T^{-1} = \begin{bmatrix} M & 0 \\ N & 0 \end{bmatrix} \]

- Consecutivity

\[ F T^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix} \]

in case of innermost temporal locality

⇒ consecutivity on next innermost loop iterator

\[ F T^{-1} = \begin{bmatrix} M & 0 & 0 \\ N & 1 & 0 \end{bmatrix} \]

(Kandemir, Ramanujam, and Choudhary 1999)
Related Work on Spatial Locality

Loop nest transformations (not per-statement)

- Wolf and Lam (1991)
  - define temporal ($\ker F$) and spatial ($\ker G$) reuse directions
  - partition original loop iterators

- Kandemir, Ramanujam, and Choudhary (1999)
  - aim: spatial locality
  - criterion more strict than required (ensures consecutivity)
  - incrementally fix elements of $T^{-1}$

- Kandemir, Ramanujam, Choudhary, and Banerjee (2001)
  - pick (second to) last column of $T^{-1}$ from $\ker G$
Related Work on Spatial Locality

Per-statement schedulers

- Bastoul and Feautrier (2004)
  - pick proto-schedule $T$ orthogonal to element from ker $G$ (or ker $F$)
  - construct valid schedule $C \cdot T$
  - imposing constraints on linear combinations
  - $\Rightarrow$ not directly applicable in isl

- Vasilache et al. (2012)
  - aim: spatial locality ($\ker T_1 \subseteq \ker G$)
  - one-shot scheduler called multiple times
  - soft constraints encoded in ILP

- Pluto (2012) post scheduling intra-tile interchange

- Kong et al. (2013)
  - aim: consecutivity (stride-1 or stride-0)
  - partition original loop iterators
  - soft constraints encoded in ILP

- Zinenko et al. (2018)
  - spatial locality through spatial proximity constraints
  - soft constraints encoded in ILP
Limitations

- partition original loop iterators
  - Kong et al. (2013)
    - loop iterators in *outer* index expressions *appear* in outer schedule rows
    - loop iterators in *innermost* index expression
      - do *not appear* in outer schedule rows

Other approaches, e.g., using $S[i, j] \rightarrow [j, -i]$: for $\textbf{i} = 0, N; \textbf{i} += 1$ for $\textbf{j} = -\textbf{i}, 0; \textbf{j} += 1$ $A[c_0][c_0 + c_1] = f(-c_1, c_0)$
Limitations

- partition original loop iterators
  Kong et al. (2013)
  - loop iterators in outer index expressions appear in outer schedule rows
  - loop iterators in innermost index expression do not appear in outer schedule rows

- consecutivity requires innermost index expression to be equal to innermost schedule row (+ linear combinations of outer schedule rows)
- how to handle iterators that appear in both?

```c
for (int i = 0; i < M; ++i)
    for (int j = 0; j < N; ++j)
        S: A[j][j - i] = f(i, j);
```
Limitations

- partition original loop iterators
  Kong et al. (2013)
  - loop iterators in outer index expressions appear in outer schedule rows
  - loop iterators in innermost index expression do not appear in outer schedule rows

- consecutivity requires innermost index expression to be equal to innermost schedule row (+ linear combinations of outer schedule rows)
- how to handle iterators that appear in both?

```c
for (int i = 0; i < M; ++i)
  for (int j = 0; j < N; ++j)
    S: A[j][j - i] = f(i, j);
```
Limitations

- partition original loop iterators
  Kong et al. (2013)
  - loop iterators in outer index expressions appear in outer schedule rows
  - loop iterators in innermost index expression do not appear in outer schedule rows
  - consecutivity requires innermost index expression to be equal to innermost schedule row (+ linear combinations of outer schedule rows)
  - how to handle iterators that appear in both?

```c
for (int i = 0; i < M; ++i)
  for (int j = 0; j < N; ++j)
    S: A[j][j - i] = f(i, j);
```

Other approaches, e.g., using $S[i,j] \rightarrow [j,-i]$:

```c
for (int c0 = 0; c0 < N; c0 += 1)
  for (int c1 = -c0; c1 <= 0; c1 += 1)
    A[c0][c0 + c1] = f(-c1, c0);
```
Limitations

- post-schedule interchange
  - does not perform reversal, skewing
  - does not differentiate between statements
  - does not affect shape of schedule (e.g., distribution)
Limitations

- post-schedule interchange
  - does not perform reversal, skewing
  - does not differentiate between statements
  - does not affect shape of schedule (e.g., distribution)

```c
void trps(int N, __pencil_consecutive float A[N][N],
           __pencil_consecutive float C[N][N])
{
    float tmp[N][N];
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++) {
            S: tmp[i][j] = A[i][j];
            T: C[j][i] = tmp[i][j];
        }
}
```

- without consecutivity:
  - temporal locality on `tmp` prevents loop distribution
- with consecutivity:
  - consecutivity requires different transformation per statement
  - loop distribution
Outline

1. Introduction
   - Consecutivity Concept
   - Pluto-Style Polyhedral Scheduling
   - Consecutivity Criterion
   - Related Work

2. Intra-Statement Consecutivity
   - Consecutivity Criterion
   - Specifying Schedule Constraints
   - Transformation to Constraints on Schedule Coefficients
   - Solving Constraints on Schedule Coefficients (isl)

3. Inter-Statement Consecutivity

4. Local Rescheduling

5. Conclusions and Future Work
Consecutivity Criterion Reformulation

Transformed access function $F T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F T^{-1} = \begin{bmatrix} G & T_1 \end{bmatrix}^{-1} \begin{bmatrix} M & 0 \\ H & T_2 \end{bmatrix} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}$$

$$G q = 0$$  
(with $q$ the final columns of $T^{-1}$)

Note: $$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & 0 \\ 0^t & 1 \end{bmatrix}$$

$\Rightarrow q$ spans $\ker T_1$
$\Rightarrow \ker T_1 \subseteq \ker G$  
(Vasilache et al. 2012)

That is, rows of $G$ need to be linear combinations of rows of $T_1$

$$G = A T_1$$
Consecutivity Criterion Reformulation

Transformed access function $F T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

\[
F T^{-1} = \begin{bmatrix} G & T_1 \end{bmatrix}^{-1} = \begin{bmatrix} M & 0 \\ H & T_2 \end{bmatrix}
\]

- $G q = 0$

Note: $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & 0 \\ 0^t & 1 \end{bmatrix}$

$\Rightarrow$ q spans ker $T_1$

$\Rightarrow$ ker $T_1 \subseteq$ ker $G$ 

(\textit{Vasilache et al. 2012})

That is, rows of $G$ need to be linear combinations of rows of $T_1$

$G = A T_1$

- $H q = 1$

$H = T_2 + B T_1$
Consecutivity Criterion Reformulation

Transformed access function $F T^{-1}$ exhibits consecutivity if

- outer index expressions independent of innermost loop iterator
- innermost index expression proportional to innermost loop iterator

$$F T^{-1} = \begin{bmatrix} G & T_1 \\ H & T_2 \end{bmatrix}^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}$$

- $G q = 0$ (with $q$ the final columns of $T^{-1}$)

Note: $\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} T^{-1} = \begin{bmatrix} I & 0 \\ 0^t & 1 \end{bmatrix}$

$\Rightarrow q$ spans ker $T_1$

$\Rightarrow$ ker $T_1 \subseteq$ ker $G$

(Vasilache et al. 2012)

That is, rows of $G$ need to be linear combinations of rows of $T_1$

$G = A T_1$

- $H q = 1$

$H = T_2 + B T_1$

$\Rightarrow H$ needs to be linearly independent of $G$
Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression
  \[ F = \begin{bmatrix} G \\ H \end{bmatrix} \]

  given
  - \( H \) linearly independent of \( G \)

Goal:
- \( G \) linear combination of outer schedule rows: \( G = A T_1 \)
- \( H \) equal to innermost schedule row: \( H = T_2 + B T_1 \)
Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression

\[ F = \begin{bmatrix} G \\ H \end{bmatrix} \]

given
  - \( H \) linearly independent of \( G \)

Goal:
  - \( G \) linear combination of outer schedule rows: \( G = A \ T_1 \)
  - \( H \) equal to innermost schedule row: \( H = T_2 + B \ T_1 \)

- multiple references per statement
  \[ \Rightarrow \text{potential conflicts} \]

Possible resolutions:
  - maximize number of satisfied consecutivity constraints
  - consider constraints in order specified by user
Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression

\[
F = \begin{bmatrix} G \\ H \end{bmatrix}
\]

given
  \( H \) linearly independent of \( G \)

Goal:
  \( G \) linear combination of outer schedule rows: \( G = A T_1 \)
  \( H \) equal to innermost schedule row: \( H = T_2 + B T_1 \)

- multiple references per statement
  \( \Rightarrow \) potential conflicts

Possible resolutions:
  \( \cdot \) maximize number of satisfied consecutivity constraints
  \( \cdot \) consider constraints in order specified by user
Multiple References

- single reference per statement
  Consecutivity constraint equal to index expression

\[ F = \begin{bmatrix} G \\ H \end{bmatrix} \]

given
- \( H \) linearly independent of \( G \)
- rows of \( H \) linearly independent

Goal:
- \( G \) linear combination of outer schedule rows: \( G = A \ T_1 \)
- \( H \) equal to innermost schedule rows: \( H = T_2 + B \ T_1 \)

- multiple references per statement
  \( \Rightarrow \) potential conflicts

Possible resolutions:
- maximize number of satisfied consecutivity constraints
- consider constraints in order specified by user
  \( \Rightarrow \) some constraints may be combined constraints with multi-row \( H \)
Multiple References Example: Matrix Multiplication

```
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```
Multiple References Example: Matrix Multiplication

```c
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

\[
F_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad F_B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad F_C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\]
Multiple References Example: Matrix Multiplication

```c
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

\[
F_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad F_B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad F_C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\]

\[
F_{BC} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\]
Multiple References Example: Matrix Multiplication

```c
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

\[ F_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad F_B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad F_C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \]

\[ F_{BC} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad F_{ABC} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \]
Multiple References Example: Matrix Multiplication

```c
for (int i = 0; i < N; ++i)
    for (int j = 0; j < M; ++j)
        for (int k = 0; k < K; ++k)
            C[i][j] += A[i][k] * B[k][j];
```

\[
F_A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0
\end{bmatrix} \quad F_B = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix} \quad F_C = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
F_{BC} = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{bmatrix} \quad F_{ABC} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

List: \( F_{ABC}, F_{AC}, F_{AB}, F_{BC}, F_A, F_B, F_C \)
Multiple Final Rows

- single final row

\[
FT^{-1} = \begin{bmatrix} M & 0 \\ N & 1 \end{bmatrix}
\]

or

\[
FT^{-1} = \begin{bmatrix} M & 0 & 0 \\ N & 1 & 0 \end{bmatrix}
\]
Multiple Final Rows

- single final row

\[
FT^{-1} = \begin{bmatrix}
M & 0 \\
N & 1
\end{bmatrix}
\]

or

\[
FT^{-1} = \begin{bmatrix}
M & 0 & 0 \\
N & 1 & 0
\end{bmatrix}
\]

- multiple final rows

\[
FT^{-1} = \begin{bmatrix}
0 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 0 & \ldots & 0 \\
\end{bmatrix}
\]

- multiple levels of consecutivity
- multiple levels of temporal locality (optional)
Constraints on Schedule Coefficients

Affine schedule row:

\[ f_S(x) = C_S x + d_S \]

Constraints:

- **validity:**
  \[ f_T(y) - f_S(x) \geq 0 \]
  Farkas \( \to \) constraints on \( C_S \) and \( d_S \)

- **proximity (temporal locality):**
  \[ f_T(y) - f_S(x) \text{ small} \]
  Farkas \( \to \) constraints on \( C_S \) and \( d_S \)

- **coincidence (parallelism):**
  \[ f_T(y) - f_S(x) = 0 \]
  Farkas \( \to \) constraints on \( C_S \) and \( d_S \)

- **linear independence of previous rows \( (T_{S,0}) \):**
  \( C_S \neq Y T_{S,0} \)
  \( \Rightarrow \) compute orthogonal complement of \( T_{S,0}: U_S T_{S,0}^t = 0 \)
  \( \Rightarrow \) impose \( U_S C_S^t \neq 0 \)
Constraints on Schedule Coefficients for Consecutivity

- $G$ linear combination of outer schedule rows: $G = A \ T_1$
- $H$ equal to innermost schedule rows: $H = T_2 + B \ T_1$
Constraints on Schedule Coefficients for Consecutivity

- \( G \) linear combination of outer schedule rows: \( G = A \ T_1 \)
- \( H \) equal to innermost schedule rows: \( H = T_2 + B \ T_1 \)

Three stages

1. \( G \) is not yet a linear combination of \( T_0 \)
   \( \Rightarrow \) take linear combination of \( G \) and \( T_0 \)
   (heuristic to make progress)
   \( \Rightarrow \) but linearly independent of \( H \) and \( T_0 \)

\[
C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \land C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix}
\]
Constraints on Schedule Coefficients for Consecutivity

- \( G \) linear combination of outer schedule rows: \( G = A \ T_1 \)
- \( H \) equal to innermost schedule rows: \( H = T_2 + B \ T_1 \)

Three stages

1. \( G \) is not yet a linear combination of \( T_0 \)
   - \( \Rightarrow \) take linear combination of \( G \) and \( T_0 \)
     (heuristic to make progress)
   - \( \Rightarrow \) but linearly independent of \( H \) and \( T_0 \)

2. \( G \) is linear combination of \( T_0 \)
   - \( \Rightarrow \) take \( C \) equal to next row of \( H \)

\[
C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \land C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix}
\]

\[
C = H_h + X \begin{bmatrix} T_1 \\ H_{<h} \end{bmatrix}
\]
Constraints on Schedule Coefficients for Consecutivity

- \( G \) linear combination of outer schedule rows: \( G = A \ T_1 \)
- \( H \) equal to innermost schedule rows: \( H = T_2 + B \ T_1 \)

Three stages

1. **G** is not yet a linear combination of \( T_0 \)
   - \( \Rightarrow \) take linear combination of \( G \) and \( T_0 \)
   - (heuristic to make progress)
   - \( \Rightarrow \) but linearly independent of \( H \) and \( T_0 \)

2. **G** is linear combination of \( T_0 \)
   - \( \Rightarrow \) take \( C \) equal to next row of \( H \)

3. All rows of \( H \) have been handled
   - \( \Rightarrow \) no further constraints (final zero columns in \( F \ T^{-1} \))

\[ C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \land C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix} \]

\[ C = H_h + X \begin{bmatrix} T_1 \\ H_{<h} \end{bmatrix} \]
Constraints on Schedule Coefficients for Consectivity

- $G$ linear combination of outer schedule rows: $G = A T_1$
- $H$ equal to innermost schedule rows: $H = T_2 + B T_1$

Three stages

1. $G$ is not yet a linear combination of $T_0$
   - $\Rightarrow$ take linear combination of $G$ and $T_0$
     - (heuristic to make progress)
   - $\Rightarrow$ but linearly independent of $H$ and $T_0$

2. $G$ is linear combination of $T_0$
   - $\Rightarrow$ take $C$ equal to next row of $H$

3. all rows of $H$ have been handled
   - $\Rightarrow$ no further constraints (final zero columns in $F T^{-1}$)

At any stage

- $C$ may also be linearly independent of $T_0$, $G$ and $H$
  (intermediate zero columns in $F T^{-1}$)
- $C$ of lower-dimensional statement may be linear combination of $T_0$

Solving Constraints on Schedule Coefficients (isl)

- validity, proximity, coincidence
  - ⇒ encoded in ILP
- linear independence
  \[ C \neq YT_0 \rightarrow UC^t \neq 0 \]
  - ⇒ not linear
  - ⇒ backtracking search (in isl): \( U_i C^t \geq 1 \) or \( U_i C^t \leq -1 \)
Solving Constraints on Schedule Coefficients (isl)

- validity, proximity, coincidence
  - encoded in ILP
- linear independence
  \[ C \neq Y T_0 \quad \Rightarrow \quad U C^t \neq 0 \]
  - not linear
  - backtracking search (in isl):
    \[ U_i C^t \geq 1 \quad \text{or} \quad U_i C^t \leq -1 \]
- consecutivity
  \[ C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \quad \Rightarrow \quad U' C^t = 0 \quad \text{linear} \]
  \[ C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix} \quad \Rightarrow \quad U'' C^t \neq 0 \quad \text{backtracking} \]

Note:
- extra rows \( H \Rightarrow \) fewer rows in \( U'' \) \( \Rightarrow \) fewer backtracking cases
- no extra ILP variables, but possibly more backtracking
Solving Constraints on Schedule Coefficients (isl)

- validity, proximity, coincidence
  ⇒ encoded in ILP

- linear independence
  \[ C \neq YT_0 \quad \Rightarrow \quad UC^t \neq 0 \]
  ⇒ not linear
  ⇒ backtracking search (in isl): \( U_iC^t \geq 1 \) or \( U_iC^t \leq -1 \)

- consecutivity

  \[ C = X \begin{bmatrix} T_0 \\ G \end{bmatrix} \quad \Rightarrow \quad U'C^t = 0 \quad \text{linear} \]

  \[ C \neq Y \begin{bmatrix} T_0 \\ H \end{bmatrix} \quad \Rightarrow \quad U''C^t \neq 0 \quad \text{backtracking} \]

Note:
- extra rows \( H \) ⇒ fewer rows in \( U'' \) ⇒ fewer backtracking cases
- no extra ILP variables, but possibly more backtracking

Differences with linear independence handling:
- optional
- fixed part that applies in each backtracking case
- disjunctive (independent or dependent rows)
- conditional (multiple consecutivity constraints)
Outline

1 Introduction
   - Consecutivity Concept
   - Pluto-Style Polyhedral Scheduling
   - Consecutivity Criterion
   - Related Work

2 Intra-Statement Consecutivity
   - Consecutivity Criterion
   - Specifying Schedule Constraints
   - Transformation to Constraints on Schedule Coefficients
   - Solving Constraints on Schedule Coefficients (isl)

3 Inter-Statement Consecutivity

4 Local Rescheduling

5 Conclusions and Future Work
Inter-Statement Consecutivity

Input:
\[
\text{for (int } i = 0; i < N; i += 2) \\
\quad \text{for (int } j = 0; j < M; j += 2) \\
\quad \quad B[j + 0][i + 0] = A[i + 0][j + 0]; \\
\quad \quad B[j + 1][i + 0] = A[i + 0][j + 1]; \\
\quad \quad B[j + 0][i + 1] = A[i + 1][j + 0]; \\
\quad \quad B[j + 1][i + 1] = A[i + 1][j + 1]; \\
\} \\
\]

Output (try and obtain distances 0 and 1):
\[
\text{for (int } c0 = 0; c0 < M - 1; c0 += 2) \\
\quad \text{for (int } c1 = 0; c1 < N - 1; c1 += 2) \\
\quad \quad B[c0][c1] = A[c1][c0]; \\
\quad \quad B[c0][c1 + 1] = A[c1 + 1][c0]; \\
\quad \}
\]
\[
\text{for (int } c1 = 0; c1 < N - 1; c1 += 2) \\
\quad B[c0 + 1][c1] = A[c1][c0 + 1]; \\
\quad B[c0 + 1][c1 + 1] = A[c1 + 1][c0 + 1]; \\
\]
\]
Outline

1. Introduction
   - Consecutivity Concept
   - Pluto-Style Polyhedral Scheduling
   - Consecutivity Criterion
   - Related Work

2. Intra-Statement Consecutivity
   - Consecutivity Criterion
   - Specifying Schedule Constraints
   - Transformation to Constraints on Schedule Coefficients
   - Solving Constraints on Schedule Coefficients (isl)

3. Inter-Statement Consecutivity

4. Local Rescheduling

5. Conclusions and Future Work
Local Rescheduling

Consecutivity usually only important inside tiles

1. compute schedule \textit{without} consecutivity (or lower priority)
2. tile
3. recompute schedule inside tile \textit{with} consecutivity
Local Rescheduling

Consecutivity usually only important inside tiles

1. compute schedule **without** consecutivity (or lower priority)
2. tile
3. recompute schedule inside tile **with** consecutivity

On trps:

```c
float tmp[N][N];
for (int c0 = 0; c0 < N; c0 += 32) {
    for (int c1 = 0; c1 < N; c1 += 32) {
        for (int c2 = c0; c2 <= min(N - 1, c0 + 31); c2 += 1)
            for (int c3 = c1; c3 <= min(N - 1, c1 + 31); c3 += 1)
                tmp[c2][c3] = A[c2][c3];
        for (int c2 = c1; c2 <= min(N - 1, c1 + 31); c2 += 1)
            for (int c3 = c0; c3 <= min(N - 1, c0 + 31); c3 += 1)
                C[c2][c3] = tmp[c3][c2];
    }
}
```
Outline

1. Introduction
   - Consecutivity Concept
   - Pluto-Style Polyhedral Scheduling
   - Consecutivity Criterion
   - Related Work

2. Intra-Statement Consecutivity
   - Consecutivity Criterion
   - Specifying Schedule Constraints
   - Transformation to Constraints on Schedule Coefficients
   - Solving Constraints on Schedule Coefficients (isl)

3. Inter-Statement Consecutivity

4. Local Rescheduling

5. Conclusions and Future Work
Conclusions and Future Work

Conclusions:

- slightly generalized criterion for consecutivity
- combining multiple references per statement
- approach for integration in Pluto-style scheduler
- implementation in isl/PPCG (branch consecutivity_CW_709)

Future work:

- experiment and fine-tune
References I


References II


References III


