The Delinearization of *C* programs

Chirantan Mukherjee, Marc Moreno Maza, Linxiao Wang

Ontario Research Centre for Computer Algebra

April 9, 2024

Content

One-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
```

One-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
```

1. Can we parallelize the two `for-loops`?

One-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
```

1. Can we parallelize the two `for-loops`?
2. Is there data dependence between two different iterations of the nest?

One-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
```

1. Can we parallelize the two `for-loops`?
2. Is there data dependence between two different iterations of the nest?
3. Are there integer solutions to the following system of linear inequalities?

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 = j_2 - 1 \\ j_1 = j_2 - i_2 - 1 \end{cases}$$

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

$$\begin{cases} 0 \le i_1 < n \\ i_1 + 1 \le j_1 < n \\ 0 \le i_2 < n \\ i_2 + 1 \le j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

$$\begin{cases} 0 \le i_1 < n \\ i_1 + 1 \le j_1 < n \\ 0 \le i_2 < n \\ i_2 + 1 \le j_2 < n \\ i_1 = j_2 - 1 \\ j_1 = j_2 - i_2 - 1 \end{cases}$$

There is no integer solution, therefore, no dependence.

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Can we replace a one-dimension array A by another multi-dimensional array B so that every subscript of B is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_e, r_1, \ldots, r_d$: data parameters (known at execution time)

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_e, r_1, \ldots, r_d$: data parameters (known at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_e)$: polynomial. Coefficients are known at compile time.

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_e, r_1, \ldots, r_d$: data parameters (known at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_e)$: polynomial. Coefficients are known at compile time.

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_e, r_1, \ldots, r_d$: data parameters (known at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_e)$: polynomial. Coefficients are known at compile time.

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

- $f_1, \ldots, f_e$ are affine functions in $i_1, \ldots, i_d$. Coefficients $\in \mathbb{Z}$ TBD

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_e, r_1, \ldots, r_d$: data parameters (known at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_e)$: polynomial. Coefficients are known at compile time.

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

- $f_1, \ldots, f_e$ are affine functions in $i_1, \ldots, i_d$. Coefficients $\in \mathbb{Z}$ TBD
- B is an $M_1 \times \cdots \times M_e$-array

Can we replace a one-dimension array A by another multi-dimensional array B so that every subscript of B is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_e, r_1, \ldots, r_d$: data parameters (known at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_e)$: polynomial. Coefficients are known at compile time.

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

- $f_1, \ldots, f_e$ are affine functions in $i_1, \ldots, i_d$. Coefficients $\in \mathbb{Z}$ TBD
- B is an $M_1 \times \cdots \times M_e$-array
- $M_1, \ldots, M_e$ are affine functions in $m_1, \ldots, m_e$ Coefficients$\in \mathbb{Z}$ TBD

Can we replace a one-dimension array `A` by another multi-dimensional array `B` so that every subscript of `B` is an affine expression of the loop counters?

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

- $i_1, \ldots, i_d \in \mathbb{N}$, with,

$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix}$$

- L: lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_e, r_1, \ldots, r_d$: data parameters (known at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_e)$: polynomial. Coefficients are known at compile time.

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

- $f_1, \ldots, f_e$ are affine functions in $i_1, \ldots, i_d$. Coefficients $\in \mathbb{Z}$ TBD
- B is an $M_1 \times \cdots \times M_e$-array
- $M_1, \ldots, M_e$ are affine functions in $m_1, \ldots, m_e$ Coefficients$\in \mathbb{Z}$ TBD

such that:

$$\boxed{R = f_1 M_2 \cdots M_e \; + \; \cdots \; + \; f_{e-1} M_2 + f_e}$$

for each $(i_1, \ldots, i_d)$ in the iteration domain we have the validity conditions:

$$\boxed{0 \leq f_1 < M_1, \quad \ldots, 0 \leq f_e < M_e.}$$

The delinearization problem is divided into:

Polynomial System Solving Problem

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.
2. Express $f_1, \ldots, f_e$ and $M_1, \ldots, M_e$ offline as functions of the coefficients of $R = f_1 M_2 \cdots M_e + \cdots + f_{e-1} M_2 + f_e$.

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.
2. Express $f_1, \ldots, f_e$ and $M_1, \ldots, M_e$ offline as functions of the coefficients of $R = f_1 M_2 \cdots M_e + \cdots + f_{e-1} M_2 + f_e$.
3. R is a polynomial in $\mathbb{Q}[i_1, \ldots i_d, m_1, \ldots, m_e]$.

Quantifier Elimination Problem

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.
2. Express $f_1, \ldots, f_e$ and $M_1, \ldots, M_e$ offline as functions of the coefficients of $R = f_1 M_2 \cdots M_e + \cdots + f_{e-1} M_2 + f_e$.
3. R is a polynomial in $\mathbb{Q}[i_1, \ldots i_d, m_1, \ldots, m_e]$.

Quantifier Elimination Problem

1. For each $(i_1, \ldots, i_d)$, we have, $\begin{cases} 0 \leq f_1 < M_1 \\ \vdots \quad \vdots \quad \vdots \\ 0 \leq f_e < M_e \end{cases}.$

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.
2. Express $f_1, \ldots, f_e$ and $M_1, \ldots, M_e$ offline as functions of the coefficients of $R = f_1 M_2 \cdots M_e + \cdots + f_{e-1} M_2 + f_e$.
3. R is a polynomial in $\mathbb{Q}[i_1, \ldots i_d, m_1, \ldots, m_e]$.

Quantifier Elimination Problem

1. For each $(i_1, \ldots, i_d)$, we have, $\begin{cases} 0 \leq f_1 < M_1 \\ \vdots \quad \vdots \quad \vdots \\ 0 \leq f_e < M_e \end{cases}$.
2. Want to do QE over $\mathbb{Z}$.

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.
2. Express $f_1, \ldots, f_e$ and $M_1, \ldots, M_e$ offline as functions of the coefficients of $R = f_1 M_2 \cdots M_e + \cdots + f_{e-1} M_2 + f_e$.
3. R is a polynomial in $\mathbb{Q}[i_1, \ldots i_d, m_1, \ldots, m_e]$.

Quantifier Elimination Problem

1. For each $(i_1, \ldots, i_d)$, we have, $\begin{cases} 0 \leq f_1 < M_1 \\ \vdots \quad \vdots \quad \vdots \\ 0 \leq f_e < M_e \end{cases}$.
2. Want to do QE over $\mathbb{Z}$.
3. Is a decidable problem using `Presburger arithmetic`.

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.
2. Express $f_1, \ldots, f_e$ and $M_1, \ldots, M_e$ offline as functions of the coefficients of $R = f_1 M_2 \cdots M_e + \cdots + f_{e-1} M_2 + f_e$.
3. R is a polynomial in $\mathbb{Q}[i_1, \ldots i_d, m_1, \ldots, m_e]$.

Quantifier Elimination Problem

1. For each $(i_1, \ldots, i_d)$, we have, $\begin{cases} 0 \leq f_1 < M_1 \\ \vdots \quad \vdots \quad \vdots \\ 0 \leq f_e < M_e \end{cases}.$
2. Want to do QE over $\mathbb{Z}$.
3. Is a decidable problem using `Presburger arithmetic`.
4. Can be solved offline over $\mathbb{R}$.

The delinearization problem is divided into:

Polynomial System Solving Problem

1. When d and e are known.
2. Express $f_1, \ldots, f_e$ and $M_1, \ldots, M_e$ offline as functions of the coefficients of $R = f_1 M_2 \cdots M_e + \cdots + f_{e-1} M_2 + f_e$.
3. R is a polynomial in $\mathbb{Q}[i_1, \ldots i_d, m_1, \ldots, m_e]$.

Quantifier Elimination Problem

1. For each $(i_1, \ldots, i_d)$, we have, $\begin{cases} 0 \leq f_1 < M_1 \\ \vdots \quad \vdots \quad \vdots \\ 0 \leq f_e < M_e \end{cases}.$
2. Want to do QE over $\mathbb{Z}$.
3. Is a decidable problem using `Presburger arithmetic`.
4. Can be solved offline over $\mathbb{R}$.

2D-2D quantifier elimination (I/II)

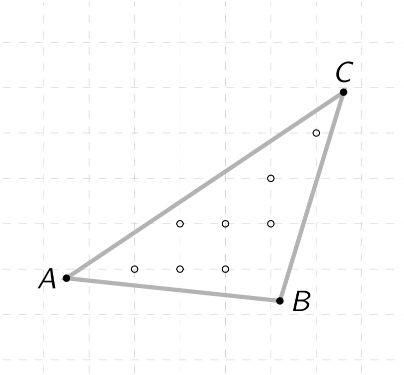
1. Loop counters can only be integers

2D-2D quantifier elimination (I/II)

1. Loop counters can only be integers
2. Finding the integer hull of a polyhedral set

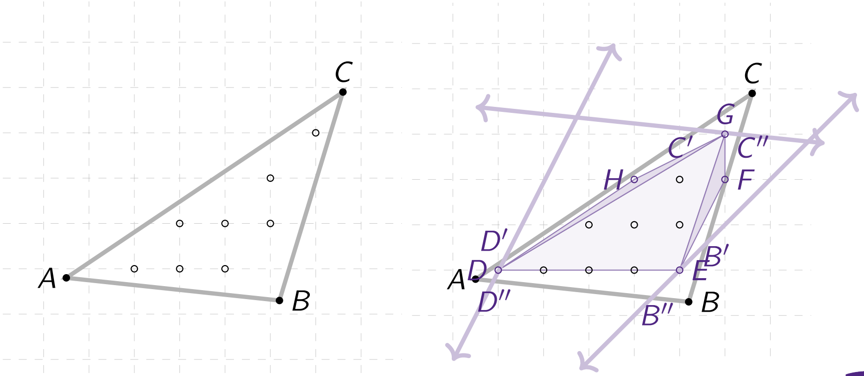
2D-2D quantifier elimination (I/II)

1. Loop counters can only be integers
2. Finding the integer hull of a polyhedral set



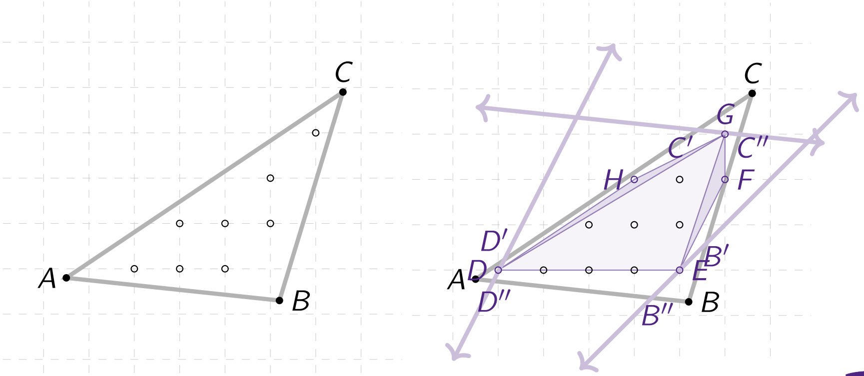
2D-2D quantifier elimination (I/II)

1. Loop counters can only be integers
2. Finding the integer hull of a polyhedral set



2D-2D quantifier elimination (I/II)

1. Loop counters can only be integers
2. Finding the integer hull of a polyhedral set



The integer hull $\{D, E, F, G, H\}$ is formed by $\{D, E, G\}$ and searching integer points $\{F, H\}$ in quadrilaterals $DD''B''E$, $EB'C''G$ and $D'DGC'$.

Parametric Integer Linear Programming

$$\begin{array}{ll} \max\limits_{(i_1,\ldots,i_d)} & f_k \\ \text{subject to} & (i_1, \ldots, i_d) \in \text{iteration domain} \\ & i_1, \ldots, i_d \in \mathbb{Z} \end{array}$$

For which, $0 \leq \max\limits_{(i_1,\ldots,i_d)} f_k < M_k$ for all $k \in [1, \ldots, e]$.

Parametric Integer Linear Programming

$$\begin{array}{ll} \max\limits_{(i_1,\ldots,i_d)} & f_k \\ \text{subject to} & (i_1, \ldots, i_d) \in \text{iteration domain} \\ & i_1, \ldots, i_d \in \mathbb{Z} \end{array}$$

For which, $0 \leq \max\limits_{(i_1,\ldots,i_d)} f_k < M_k$ for all $k \in [1, \ldots, e]$.

1. `PIP/PipLib` by Paul Feautrier
2. `isl` by Sven Verdoolaege
3. `barvinok` by Sven Verdoolaege

2D-2D delinearization problem

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

2D-2D delinearization problem

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

$d = 2 \rightsquigarrow i_1, i_2$ loop counters

$e = 2 \rightsquigarrow m_1, m_2$ program parameters

} known at compile time.

2D-2D delinearization problem

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

$d = 2 \rightsquigarrow i_1, i_2$ loop counters
$e = 2 \rightsquigarrow m_1, m_2$ program parameters
} known at compile time.

$$\left.\begin{aligned} M_1 &= a_1 m_1 + b_1 \\ M_2 &= a_2 m_2 + b_2 \end{aligned}\right\}, \text{ where } a_1, b_1, a_2, b_2 \in \mathbb{Z} \text{ TBD.}$$

2D-2D delinearization problem

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

$d = 2 \rightsquigarrow i_1, i_2$ loop counters

$e = 2 \rightsquigarrow m_1, m_2$ program parameters

} known at compile time.

$$\left.\begin{aligned} M_1 &= a_1 m_1 + b_1 \\ M_2 &= a_2 m_2 + b_2 \end{aligned}\right\}, \text{ where } a_1, b_1, a_2, b_2 \in \mathbb{Z} \text{ TBD.}$$

The reference $A[R]$ to A which encodes a reference $B[f1][f2]$ to B, where:

2D-2D delinearization problem

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

$$\left.\begin{array}{lll} d = 2 \rightsquigarrow i_1, i_2 & \text{loop counters} \\ e = 2 \rightsquigarrow m_1, m_2 & \text{program parameters} \end{array}\right\} \text{known at compile time.}$$

$$\left.\begin{array}{l} M_1 = a_1 m_1 + b_1 \\ M_2 = a_2 m_2 + b_2 \end{array}\right\}, \text{ where } a_1, b_1, a_2, b_2 \in \mathbb{Z} \text{ TBD.}$$

The reference $A[R]$ to A which encodes a reference $B[f1][f2]$ to B, where:

- $$\begin{cases} f1 = f_{11} i_1 + f_{12} i_2 + f_{10} \\ f2 = f_{21} i_1 + f_{22} i_2 + f_{20} \end{cases} \quad \text{and} \quad R = f1 M_2 + f2$$

2D-2D delinearization problem

Input

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   A[R(i_1, ..., i_d, m_1, ..., m_e)] <- ...
```

Output

```
for(i_1, ..., i_1 ++)
 ...
  for(i_d, ..., i_d ++)
   B[f_1], ..., B[f_e] <- ...
```

$$\left.\begin{array}{lll} d = 2 \rightsquigarrow i_1, i_2 & \text{loop counters} \\ e = 2 \rightsquigarrow m_1, m_2 & \text{program parameters} \end{array}\right\} \text{known at compile time.}$$

$$\left.\begin{array}{l} M_1 = a_1 m_1 + b_1 \\ M_2 = a_2 m_2 + b_2 \end{array}\right\}, \text{ where } a_1, b_1, a_2, b_2 \in \mathbb{Z} \text{ TBD.}$$

The reference $A[R]$ to A which encodes a reference $B[f1][f2]$ to B, where:

- $\begin{cases} f1 = f_{11} i_1 + f_{12} i_2 + f_{10} \\ f2 = f_{21} i_1 + f_{22} i_2 + f_{20} \end{cases}$ and $R = f1 M_2 + f2$
- for each (i_1, i_2), we have, $\begin{cases} 0 \leq f1 < M_1 \\ 0 \leq f2 < M_2 \end{cases}$, thus $\begin{cases} 0 \leq \max f1 < M_1 \\ 0 \leq \max f2 < M_2 \end{cases}$.

2D-2D polynomial system solving

Substituting $\begin{cases} f1 = f_{11} i_1 + f_{12} i_2 + f_{10} \\ f2 = f_{21} i_1 + f_{22} i_2 + f_{20} \end{cases}$ in $R = f1 M_2 + f2$, we obtain,

2D-2D polynomial system solving

Substituting $\begin{cases} f1 = f_{11} i_1 + f_{12} i_2 + f_{10} \\ f2 = f_{21} i_1 + f_{22} i_2 + f_{20} \end{cases}$ in $R = f1 M_2 + f2$, we obtain,

$$R = \underbrace{a_2 f_{11}}_{T_1} i_1 m_2 + \underbrace{a_2 f_{12}}_{T_2} i_2 m_2 + \underbrace{a_2 f_{10}}_{T_3} m_2 + (\underbrace{b_2 f_{11} + f_{21}}_{T_4}) i_1 + (\underbrace{b_2 f_{12} + f_{22}}_{T_5}) i_2 + (\underbrace{b_2 f_{10} + f_{20}}_{T_6}).$$

2D-2D polynomial system solving

Substituting $\begin{cases} f1 = f_{11} i_1 + f_{12} i_2 + f_{10} \\ f2 = f_{21} i_1 + f_{22} i_2 + f_{20} \end{cases}$ in $R = f1 M_2 + f2$, we obtain,

$$R = \underbrace{a_2 f_{11}}_{T_1} i_1 m_2 + \underbrace{a_2 f_{12}}_{T_2} i_2 m_2 + \underbrace{a_2 f_{10}}_{T_3} m_2 + (\underbrace{b_2 f_{11} + f_{21}}_{T_4}) i_1 + (\underbrace{b_2 f_{12} + f_{22}}_{T_5}) i_2 + (\underbrace{b_2 f_{10} + f_{20}}_{T_6}).$$

$$\begin{cases} T_1 = a_2 f_{11} \\ T_2 = a_2 f_{12} \\ T_3 = a_2 f_{10} \\ T_4 = b_2 f_{11} + f_{21} \\ T_5 = b_2 f_{12} + f_{22} \\ T_6 = b_2 f_{10} + f_{20} \end{cases} \implies \begin{cases} f_{11} = \dfrac{T_1}{a_2} \\ f_{12} = \dfrac{T_2}{a_2} \\ f_{10} = \dfrac{T_3}{a_2} \\ f_{21} = T_4 - b_2 f_{11} \\ f_{22} = T_5 - b_2 f_{12} \\ f_{20} = T_6 - b_2 f_{10} \end{cases}$$

2D-2D polynomial system solving

Substituting $\begin{cases} f1 = f_{11} i_1 + f_{12} i_2 + f_{10} \\ f2 = f_{21} i_1 + f_{22} i_2 + f_{20} \end{cases}$ in $R = f1 M_2 + f2$, we obtain,

$$R = \underbrace{a_2 f_{11}}_{T_1} i_1 m_2 + \underbrace{a_2 f_{12}}_{T_2} i_2 m_2 + \underbrace{a_2 f_{10}}_{T_3} m_2 + (\underbrace{b_2 f_{11} + f_{21}}_{T_4}) i_1 + (\underbrace{b_2 f_{12} + f_{22}}_{T_5}) i_2 + (\underbrace{b_2 f_{10} + f_{20}}_{T_6}).$$

$$\begin{cases} T_1 = a_2 f_{11} \\ T_2 = a_2 f_{12} \\ T_3 = a_2 f_{10} \\ T_4 = b_2 f_{11} + f_{21} \\ T_5 = b_2 f_{12} + f_{22} \\ T_6 = b_2 f_{10} + f_{20} \end{cases} \implies \begin{cases} f_{11} = \frac{T_1}{a_2} \\ f_{12} = \frac{T_2}{a_2} \\ f_{10} = \frac{T_3}{a_2} \\ f_{21} = T_4 - b_2 f_{11} \\ f_{22} = T_5 - b_2 f_{12} \\ f_{20} = T_6 - b_2 f_{10} \end{cases}$$

a_2, b_2 can NOT be uniquely determined, but $a_2 \mid gcd(T_1, T_2, T_3)$.

2D-2D quantifier elimination (I/II)

For each (i_1, i_2), we have, $0 \leq f2 < M_2$, thus $0 \leq \max f2 < M_2$,

$$\begin{cases} \max\limits_{(i_1,i_2)} & f_2 \\ \text{subject to} & (i_1, i_2) \in \text{iteration domain} \\ & i_1, i_2 \in \mathbb{Z} \end{cases}$$

2D-2D quantifier elimination (I/II)

For each (i_1, i_2), we have, $0 \leq f2 < M_2$, thus $0 \leq \max f2 < M_2$,

$$\begin{cases} \max\limits_{(i_1,i_2)} & f_2 \\ \text{subject to} & (i_1, i_2) \in \text{iteration domain} \\ & i_1, i_2 \in \mathbb{Z} \end{cases}$$

Depending on the shape of the iteration domain, we solve on a case to case basis.

2D-2D quantifier elimination (I/II) - Rectangular domain

For a `for loop` of the form,

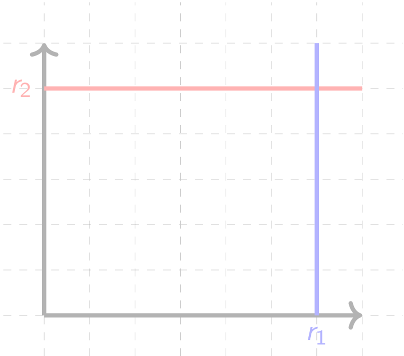
```
for (int i_1 = 0; i_1 < r_1; i_1 ++)
  for (int i_2 = 0; i_2 < r_2; i_2 ++)
```

2D-2D quantifier elimination (I/II) - Rectangular domain

For a `for loop` of the form,

```
for (int i_1 = 0; i_1 < r_1; i_1 ++)
  for (int i_2 = 0; i_2 < r_2; i_2 ++)
```

the iteration domain is of the shape,



2D-2D quantifier elimination (I/II) - Triangular domain

For a `for loop` of the form,

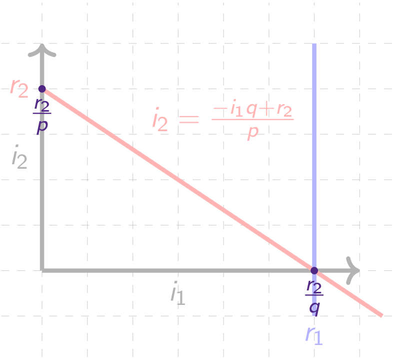
```
for (int i_1 = 0; i_1 < r_1; i_1 ++)
  for (int i_2 = 0; p * i_1 + q * i_2 < r_2; i_2 ++)
```

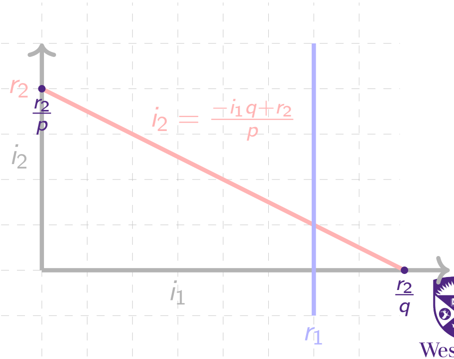
2D-2D quantifier elimination (I/II) - Triangular domain

For a `for loop` of the form,

```
for (int i_1 = 0; i_1 < r_1; i_1 ++)
  for (int i_2 = 0; p * i_1 + q * i_2 < r_2; i_2 ++)
```

the iteration domain is of the shape,





2D-2D quantifier elimination (I/II)

The parametric integer linear problem:

$$\begin{cases} \max\limits_{(i_1,i_2)} & f_2 = f_{21} i_1 + f_{22} i_2 + f_{20} \\ \text{subject to} & (i_1, i_2) \in \text{iteration domain} \\ & i_1, i_2 \in \mathbb{Z} \end{cases}$$

can be solved for,

2D-2D quantifier elimination (I/II)

The parametric integer linear problem:

$$\begin{cases} \max\limits_{(i_1,i_2)} & f_2 = f_{21} i_1 + f_{22} i_2 + f_{20} \\ \text{subject to} & (i_1, i_2) \in \text{iteration domain} \\ & i_1, i_2 \in \mathbb{Z} \end{cases}$$

can be solved for,

1. Rectangular domain by case inspection

2D-2D quantifier elimination (I/II)

The parametric integer linear problem:

$$\begin{cases} \max\limits_{(i_1,i_2)} & f_2 = f_{21} i_1 + f_{22} i_2 + f_{20} \\ \text{subject to} & (i_1, i_2) \in \text{iteration domain} \\ & i_1, i_2 \in \mathbb{Z} \end{cases}$$

can be solved for,

1. Rectangular domain by case inspection
2. Triangular domain by case inspection except when `f_21, f_22 > 0`, in which case the problem becomes,

2D-2D quantifier elimination (I/II)

The parametric integer linear problem:

$$\begin{cases} \max\limits_{(i_1,i_2)} & f_2 = f_{21} i_1 + f_{22} i_2 + f_{20} \\ \text{subject to} & (i_1, i_2) \in \text{iteration domain} \\ & i_1, i_2 \in \mathbb{Z} \end{cases}$$

can be solved for,

1. Rectangular domain by case inspection
2. Triangular domain by case inspection except when `f_21, f_22 > 0`, in which case the problem becomes,

$$\begin{array}{lll} \max\limits_{i_1} & f_{21} i_1 + f_{22} \lfloor \frac{-i_1 q + r_2}{p} \rfloor + f_{20} & \\ \text{subject to} & 0 \leq i_1 < r_1, & i_1 \in \mathbb{Z} \end{array}$$

2D-2D quantifier elimination (II/II)

QE over $\mathbb{R}$,

2D-2D quantifier elimination (II/II)

QE over $\mathbb{R}$, uses the `QuantifierElimination` function from the `RegularChains:-SemiAlgebraicSetTools` package.

2D-2D quantifier elimination (II/II)

QE over $\mathbb{R}$, uses the `QuantifierElimination` function from the `RegularChains:-SemiAlgebraicSetTools` package.

```
f := &A([i_1, i_2]), ((0 < i_1) &and (i_1 < r_1) &and
                      (0 < i_2) &and (i_2 < r_2) &and
                      (0 < r_1) &and (0 < r_2) &and
                      (0 < f_{21}) &and (0 < f_{22}) &and (0 < B))
                      \\ B = M_2 - f_{20}
            &implies (f_{21} * i_1 + f_{22} * i_2 < B);
```

2D-2D quantifier elimination (II/II)

QE over $\mathbb{R}$, uses the `QuantifierElimination` function from the `RegularChains:-SemiAlgebraicSetTools` package.

```
f := &A([i_1, i_2]), ((0 < i_1) &and (i_1 < r_1) &and
                      (0 < i_2) &and (i_2 < r_2) &and
                      (0 < r_1) &and (0 < r_2) &and
                      (0 < f_{21}) &and (0 < f_{22}) &and (0 < B))
                      \\ B = M_2 - f_{20}
             &implies (f_{21} * i_1 + f_{22} * i_2 < B);
```

After simplification, `r_1 * f_{21} + r_2 * f_{22} + f_{20} < M_2`.

2D-2D quantifier elimination (II/II)

QE over $\mathbb{R}$, uses the `QuantifierElimination` function from the `RegularChains:-SemiAlgebraicSetTools` package.

```
f := &A([i_1, i_2]), ((0 < i_1) &and (i_1 < r_1) &and
                      (0 < i_2) &and (i_2 < r_2) &and
                      (0 < r_1) &and (0 < r_2) &and
                      (0 < f_{21}) &and (0 < f_{22}) &and (0 < B))
                      \\ B = M_2 - f_{20}
           &implies (f_{21} * i_1 + f_{22} * i_2 < B);
```

After simplification, `r_1 * f_{21} + r_2 * f_{22} + f_{20} < M_2`. Substituting, $f_{11} = \frac{T_1}{a_2}, f_{12} = \frac{T_2}{a_2}, f_{10} = \frac{T_3}{a_2}, f_{21} = T_4 - b_2 f_{11}, f_{22} = T_5 - b_2 f_{12}, f_{20} = T_6 - b_2 f_{10}, M_2 = a_2 m_2 + b_2$, we obtain,

$$\boxed{r_1(T_4 - b_2\frac{T_1}{a_2}) + r_2(T_5 - b_2\frac{T_2}{a_2}) + T_6 - b_2\frac{T_3}{a_2} < a_2 m_2 + b_2.}$$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$
2. $T_1 = 2$, $T_2 = 0$, $T_3 = 1$, $T_4 = 0$, $T_5 = 3$, $T_6 = 2$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$
2. $T_1 = 2$, $T_2 = 0$, $T_3 = 1$, $T_4 = 0$, $T_5 = 3$, $T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$
2. $T_1 = 2,\ T_2 = 0,\ T_3 = 1,\ T_4 = 0,\ T_5 = 3,\ T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,
   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 5 < 10$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,

   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 5 < 10$

   delinearization valid when $r_1 = r_2 = 2, max\ f2 = 8 < 10$

Examples (I/II) - Rectangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = r_2$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,

   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 2, max\ f2 = 8 < 10$
   delinearization invalid when $r_1 = r_2 = 3, max\ f2 = 11 \not< 10$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2$, $T_2 = 0$, $T_3 = 1$, $T_4 = 0$, $T_5 = 3$, $T_6 = 2$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2$, $T_2 = 0$, $T_3 = 1$, $T_4 = 0$, $T_5 = 3$, $T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,
   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 2 < 10$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2$, $T_2 = 0$, $T_3 = 1$, $T_4 = 0$, $T_5 = 3$, $T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,
   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 2 < 10$
   delinearization valid when $r_1 = r_2 = 2, max\ f2 = 5 < 10$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,
   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 2 < 10$
   delinearization valid when $r_1 = r_2 = 2, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 3, max\ f2 = 5 < 10$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,
   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 2 < 10$
   delinearization valid when $r_1 = r_2 = 2, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 3, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 4, max\ f2 = 8 < 10$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 = ?, b_2 = ?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,

   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 2 < 10$
   delinearization valid when $r_1 = r_2 = 2, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 3, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 4, max\ f2 = 8 < 10$
   delinearization valid when $r_1 = r_2 = 5, max\ f2 = 8 < 10$

Examples (II/II) - Triangular domain

```
for (int i_1 = 0; i_1 <= r_1; i_1 ++)
  for (int i_2 = 0; i_1 + 2 * i_2 <= r_2; i_2 ++)
    A[2 * i_1 * m_2 + m_2 + 3 * i_2 + 2] = ...;
```

1. $max\ i_1 = r_1, max\ i_2 = \lfloor \frac{r_2}{2} \rfloor$
2. $T_1 = 2, T_2 = 0, T_3 = 1, T_4 = 0, T_5 = 3, T_6 = 2$
3. $a_2 =?, b_2 =?, f_{11} = 2, f_{12} = 0, f_{10} = 1, f_{21} = -2b_2, f_{22} = 3, f_{20} = 2 - b_2$
4. the validity condition $-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2 m_2 + b_2$
5. evaluating at $a_2 = 1, b_2 = 0$, we obtain, $f1 = 2i_1 + 1, f2 = 3i_2 + 2$
6. assuming $m_2 = 10$, i.e. $B[...][10]$, we get,

   delinearization valid when $r_1 = r_2 = 1, max\ f2 = 2 < 10$
   delinearization valid when $r_1 = r_2 = 2, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 3, max\ f2 = 5 < 10$
   delinearization valid when $r_1 = r_2 = 4, max\ f2 = 8 < 10$
   delinearization valid when $r_1 = r_2 = 5, max\ f2 = 8 < 10$
   delinearization invalid when $r_1 = r_2 = 6, max\ f2 = 11 \not< 10$

Concluding remarks

Summary and notes

1. In the area of optimizing compilers, delinearization of 1D array is a necessary step before applying the techniques of the polyhedral model.
2. Current compilers use heuristics fail to delinearize some for-loop nests.
3. Solving the delinearization problem is an algebraic problem with two sub-problems: a *polynomial system solving* one and a *QE* one.
4. It is desirable to solve them *at compile time* as much as possible, although some parameters are only known at *execution time*.
5. We have shown that this is indeed possible for some classes of delinearization problems (2D-2D and specific iterations domains)

Work in progress

1. We are currently extending our results to higher dimension
2. Solving the QE problem at compile time requires to improve existing techniques for PILP (parametric integer linear programming) and/or Presburger arithmetic.

References I

Johannes Doerfert, Tobias Grosser and Sebastian Hack.
Optimistic loop optimization.
IEEE Press, 2017.

Tobias Grosser, J. Ramanujam, Louis-Noël Pouchet, P. Sadayappan and Sebastian Pop.
Optimistic Delinearization of Parametrically Sized Arrays.
Association for Computing Machinery, 2015.

Mohamed-Walid Benabderrahmane, Louis-Noël Pouchet, Albert Cohen and Cédric Bastoul.
The Polyhedral Model Is More Widely Applicable Than You Think.
Springer, 2010.

References II

Paul Feautrier.
Automatic parallelization in the polytope model.
Springer, 2005.

Michael J. Fischer and Michael O. Rabin.
Super-Exponential Complexity of Presburger Arithmetic.
Springer, 1998.

Marc Moreno Maza and Linxiao Wang.
Computing the Integer Hull of Convex Polyhedral Sets.
Springer, 2022.

Paul Feautrier.
PIP/PipLib, 1988.
http://www.piplib.org/

References III

Sven Verdoolage.
isl, 2009.
http://freshmeat.net/projects/isl/

Sven Verdoolage.
barvinok, 2003.
http://freshmeat.net/projects/barvinok/