On the delinearization of array references in for-loop nests

Marc Moreno Maza, Delaram Talaashrafi, Lin-Xiao Wang
University of Western Ontario, London, Ontario, Canada

LLVM Polyhedral Model Group
September 25, 2019

Plan

About our research group

The delinearization problem

The 2D-2D delinearization problem

The 3D-3D delinearization problem

Pattern matching via rank-preserving unimodular transformations

Concluding remarks

Plan

About our research group

The delinearization problem

The 2D-2D delinearization problem

The 3D-3D delinearization problem

Pattern matching via rank-preserving unimodular transformations

Concluding remarks

About us

The Ontario Research Center for Computer Algebra

1. Research directions: symbolic computation, but also high-performance computing
2. Industrial partners: IBM annd Maplesoft.

Sofware projects

1. in support of the `solve` command in MAPLE
2. `Polyehdralsets`, `ProgramAnalysis` (available in MAPLE) $\mathbb{Z}$-`Polyehdralsets` (to appear in MAPLE)
3. Basic Polynomial Algebra Subprograms
4. CUMODP Library
5. Metafork framework

Plan

Input:

for $(\mathbf{i}_1 \cdots;\ \cdots;\ \mathbf{i}_1{+}{+})$ **do**
  ...
  for $(\mathbf{i}_d \cdots; \cdots;\ \mathbf{i}_d{+}{+})$ **do**
    $A[R(\mathbf{i}_1, \ldots, \mathbf{i}_d, \mathbf{m}_1, \ldots, \mathbf{m}_\delta)] \leftarrow \cdots$
  end for
  ...
end for

- $\mathbf{i}_1, \ldots, \mathbf{i}_d$ take non-negative integer values such that

$$L \begin{pmatrix} \mathbf{i}_1 \\ \vdots \\ \mathbf{i}_d \end{pmatrix} \leq \begin{pmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_d \end{pmatrix},$$

- L is a lower-triangular full-rank matrix over $\mathbb{Z}_+$ (known at compile time) defining the iteration domain
- $\mathbf{m}_1, \ldots, \mathbf{m}_\delta, \mathbf{r}_1, \ldots, \mathbf{r}_\delta$: data parameters (known only at execution time)
- $R(\mathbf{i}_1, \ldots, \mathbf{i}_d, \mathbf{m}_1, \ldots, \mathbf{m}_\delta)$ is a polynomial, the coefficients of which are known at compile time.

Input:

for $(\mathbf{i}_1 \cdots;\ \cdots;\ \mathbf{i}_1{+}{+})$ **do**
  $\ldots$
  for $(\mathbf{i}_d \cdots; \cdots;\ \mathbf{i}_d{+}{+})$ **do**
    $A[R(\mathbf{i}_1, \ldots, \mathbf{i}_d, \mathbf{m}_1, \ldots, \mathbf{m}_\delta)] \leftarrow \cdots$
  end for
  $\ldots$
end for

- $\mathbf{i}_1, \ldots, \mathbf{i}_d$ take non-negative integer values such that

$$L \begin{pmatrix} \mathbf{i}_1 \\ \vdots \\ \mathbf{i}_d \end{pmatrix} \leq \begin{pmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_d \end{pmatrix},$$

- L is a lower-triangular full-rank matrix over $\mathbb{Z}_+$ (known at compile time) defining the iteration domain
- $\mathbf{m}_1, \ldots, \mathbf{m}_\delta, \mathbf{r}_1, \ldots, \mathbf{r}_\delta$: data parameters (known only at execution time)
- $R(\mathbf{i}_1, \ldots, \mathbf{i}_d, \mathbf{m}_1, \ldots, \mathbf{m}_\delta)$ is a polynomial, the coefficients of which are known at compile time.

Output:

for $(\mathbf{i}_1 \cdots;\ \cdots;\ \mathbf{i}_1{+}{+})$ **do**
  $\ldots$
  for $(\mathbf{i}_d \cdots; \cdots;\ \mathbf{i}_d{+}{+})$ **do**
    $\tilde{A}[f_1] \cdots [f_\delta] \leftarrow \cdots$
  end for
  $\ldots$
end for

- $f_1, \ldots, f_\delta$ are affine forms in $\mathbf{i}_1, \ldots, \mathbf{i}_d$ the coefficients of which are integers to-be-determined,
- $\tilde{A}$ is an $\mathbf{M}_1 \times \cdots \times \mathbf{M}_\delta$-array,
- $\mathbf{M}_1, \ldots, \mathbf{M}_\delta$ are affine forms in $\mathbf{m}_1, \ldots, \mathbf{m}_d$ the coefficients of which are integers TBD,

such that:

$$\boxed{R = f_1 \mathbf{M}_2 \cdots \mathbf{M}_\delta \ + \ \cdots \ + \ f_{\delta-1} \mathbf{M}_2 + f_\delta}$$

holds and for each $(\mathbf{i}_1, \ldots, \mathbf{i}_d)$ in the iteration domain we have:

$$\boxed{0 \leq f_1 < \mathbf{M}_1, \quad \ldots, 0 \leq f_\delta < \mathbf{M}_\delta.}$$

The sub-problems

Polynomial system solving

1. Expressing the coefficients of $f_1, \ldots, f_\delta$ and $\mathbf{M}_1, \ldots, \mathbf{M}_\delta$ as functions of the coefficients of R
2. This can be done off-line (that is, before compile-time) once d and δ are fixed.
3. Recall that the matrix L and the coefficients of the polynomial R are integer values known at compile-time.

Quantifier elimination

1. The constraint: for each $(\mathbf{i}_1, \ldots, \mathbf{i}_d)$ in the iteration domain we have:
$$0 \leq f_1 < \mathbf{M}_1, \quad \ldots, 0 \leq f_\delta < \mathbf{M}_\delta$$
implies constraints on the coefficients of $f_1, \ldots, f_\delta$. and $\mathbf{M}_1, \ldots, \mathbf{M}_\delta$
2. Off-line, this is a non-linear QE problem which can only be solved over the reals (not over the integers). The obtained constraint is then sufficient but not necessary.
3. At compile time, the coefficients of L and R are known and the QE problem can be reduced to Presburger arithmetic (that is, QE on affine forms over $\mathbb{Z}$) which can be solved by software like ISL.
4. This is the point of view of the paper *Optimistic Delinearization of Parametrically Sized Arrays* by T. Grosser, J. Ramanujam, L.-N. Pouchet, P. Sadayappan and S. Pop (ICS15).
5. At run-time, $\mathbf{m}_1, \ldots, \mathbf{m}_\delta, \mathbf{r}_1, \ldots, \mathbf{r}_\delta$ are known and the QE problem reduces to optimize peice-wise linear functions (actually *sawtooth functions*).

Plan

The 2D-2D case: set up

Loop counters and dimension sizes

1. $d = 2$, that is, 2 loop counters: $\mathbf{i}$ and $\mathbf{j}$.
2. $\delta = 2$, that is, the target array $\tilde{A}$ is 2D with dimension sizes $M_1 = a_1\mathbf{m} + b_1$ and $M_2 = a_2\mathbf{n} + b_2$, where $\mathbf{m}$, $\mathbf{n}$ are data parameters known at execution while a_1, b_1, a_2, b_2 are integers TBD.

Array references

1. Given the reference $A[R]$ to the array A with
$$R = T_1\,\mathbf{in} \;+\; T_2\,\mathbf{jn} \;+\; T_3\,\mathbf{n} \;+\; T_4\,\mathbf{i} \;+\; T_5\,\mathbf{j} \;+\; T_6,$$
2. we want f_1, f_2 such that
$$R = f_1 M_2 + f_2$$
with
$$f_1 = e_1\,\mathbf{i} + g_1\mathbf{j} + c_1 \text{ and } f_2 = e_2\mathbf{i} + g_2\mathbf{j} + c_2,$$
3. and for each iteration $(\mathbf{i},\mathbf{j})$ in the domain we have
$$0 \le f_2 < M_2.$$

The 2D-2D case: polynomial system solving

The system

From $R = f_1 M_2 + f_2$, we derive

$$\begin{cases} T_1 & = & a_2 e_1 \\ T_2 & = & a_2 g_1 \\ T_3 & = & a_2 c_1 \\ T_4 & = & b_2 e_1 + e_2 \\ T_5 & = & b_2 g_1 + g_2 \\ T_6 & = & b_2 c_1 + c_2 \end{cases} \tag{1}$$

Its solution

$$\begin{cases} e_1 & = & \frac{T_1}{a_2} \\ g_1 & = & \frac{T_2}{a_2} \\ c_1 & = & \frac{T_3}{a_2} \\ e_2 & = & T_4 - b_2 e_1 \\ g_2 & = & T_5 - b_2 g_1 \\ c_2 & = & T_6 - b_2 c_1 \end{cases} \tag{2}$$

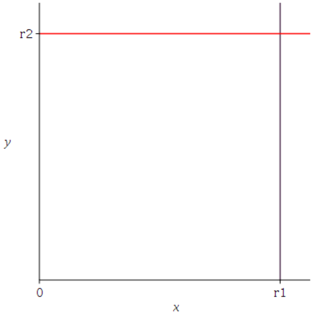
We observe that a_2 and b_2 cannot be uniquely determined. However, since a_2 is integer, it must divide T_1, T_2, T_3.

The 2D-2D case: the iteration domain

for-loops of the form

for $(\mathbf{i} = 0; \mathbf{i} < r_1; \mathbf{i}{+}{+})$ **do**
    for $(\mathbf{j} = 0; \mathbf{j} < r_2; \mathbf{j}{+}{+})$ **do**
        ...
    end for
end for

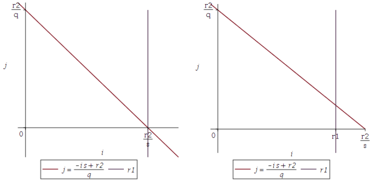
the iteration domain, say $\mathcal{D}$, looks like:



for-loops of the form

for $(\mathbf{i} = 0; \mathbf{i} < r_1; \mathbf{i}{+}{+})$ **do**
    for $(\mathbf{j} = 0; q\mathbf{i} + s\mathbf{j} < r_2; \mathbf{j}{+}{+})$ **do**
        ...
    end for
end for

the iteration domain looks like:



The 2D-2D case: QE solving

From QE to ILP

1. Recall the *validity condition*: for each $(\mathbf{i}, \mathbf{j})$ we have $0 \leq f_2 < M_2$.
2. That is: $0 \leq f_2$ and $F_2 < M_2$ both hold where F_2 is the maximum value of the following *integer linear programming* (ILP) problem

$$\begin{array}{ll} \underset{(\mathbf{i},\mathbf{j})}{\text{maximize}} & e_2\mathbf{i} + g_2\mathbf{j} + c_2 \\ \text{subject to} & (\mathbf{i}, \mathbf{j}) \in \mathcal{D} \end{array}$$

Solving the ILP problem

1. For the rectangular domain, the problem is solved by case inspection
2. For the truncated triangular domain, the problem is easy to solve by case inspection, except when $e_2 > 0$ and $g_2 > 0$ both hold.
3. For that special case, the problem becomes:

$$\begin{array}{ll} \underset{\mathbf{i}}{\text{maximize}} & e_2\,\mathbf{i} + g_2\lfloor \frac{r_2 - qi}{s} \rfloor + c_2 \\ \text{subject to} & 0 \leq \mathbf{i} < r_1 \end{array}$$

Formalizing the case $e_2 > 0$ and $g_2 > 0$

- Let a, b, c, d, e be integers such that $c > 0$, $d > 0$, $e > 0$ and $a < 0$ hold. We consider the function:

$$f : \begin{array}{ccc} \mathbb{Z} & \rightarrow & \mathbb{Q} \\ x & \longmapsto & e\lfloor \frac{ax+b}{d} \rfloor + cx \end{array}$$

  Given a positive integer X, we need to maximize $f(x)$ over $[0, X]$.
- Let x' be given by:

$$x' = \begin{cases} 0 & \text{if} \quad f(0) > f(X) \\ X & \text{otherwise.} \end{cases}$$

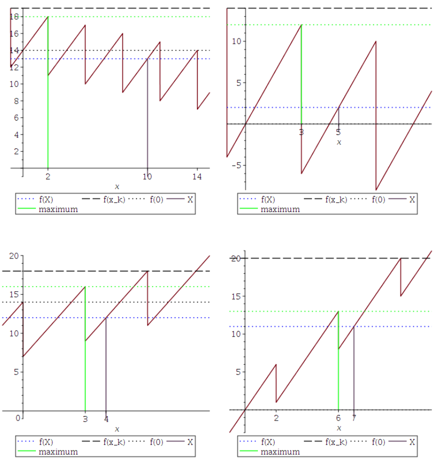
  Let $k \in \mathbb{Z}$ be such that

$$k = \begin{cases} \lfloor \frac{aX+b}{d} \rfloor & \text{if} \quad e + c\frac{d}{a} < 0 \\ \lfloor \frac{b}{d} \rfloor + 1 & \text{if} \quad e + c\frac{d}{a} > 0 \end{cases}$$

  Then, the number of evaluations of f over $[0, X]$ (in order to find its maximum) is no more than $\frac{c\,(dk-b-ax')}{ae+cd}$.
- The case $e + c\frac{d}{a} = 0$ is easy to handle.
- This rough estimate could be improved, making it feasible to evaluate the validity condition at execution time without solving any QE problem at compile-time.
- However, this requires guessing a_2, a_3, b_2, say trying $a_2 = a_3 = 1$ and $b_2 = 0$.

Problem visualization

There are mainly four different cases. Based on the shape of the plots, we call this type of functions "sawtooth" functions.



The 2D-2D case: QE solving again

Formal solution over the reals

Using the RegularChans library in MAPLE we can solve the following QE query (over the reals):

```
f := &A([i,j]), ((0 < i) &and (i < r1) &and
                 (0 < j) &and ( j < r2)  &and
                 (0 < r1) &and (0 < r2) &and
                 (0 < e2) &and (0 < g2) &and (0 < B))
       &implies (e2 * i + g2 * j < B);
sols := QuantifierElimination(f);
```

After simplification, with $B = M_2 - c_2$, we obtain:

$$r_1 e_2 + r_2 g_2 + c_2 < M_2$$

Consequence for b_2

Recall that we have:

$$e_1 = \frac{T_1}{a_2},\; g_1 = \frac{T_2}{a_2},\; c_1 = \frac{T_3}{a_2},\; e_2 = T_4 - b_2 e_1,\; g_2 = T_5 - b_2 g_1,\; \text{ and }\; c_2 = T_6 - b_2 c_1.$$

Hence we have: $\boxed{r_1(T_4 - b_2 \frac{T_1}{a_2}) + r_2(T_5 - b_2 \frac{T_2}{a_2}) + T_6 - b_2 \frac{T_3}{a_2} < a_2 \mathbf{n} + b_2.}$

Once the values of $T_1, \ldots, T_6, r_1, r_2, \mathbf{n}$ are known, we obtain a condition on b_2 (and a_2).

The 2D-2D case: rectangular domain

Loop Format:

for ($\mathbf{i} = 0; \mathbf{i} \le r_1; \mathbf{i}{+}{+}$) **do**
  for ($\mathbf{j} = 0; \mathbf{j} \le r_2; \mathbf{j}{+}{+}$) **do**
    $A[2 * \mathbf{i} * \mathbf{n} + \mathbf{n} + 3 * \mathbf{j} + 2] = \cdots$
  end for
end for

- We have $T = [2, 0, 1, 0, 3, 2]$.
- $[a_2 = a_2,\ b_2 = b_2,\ e_1 = 2,\ e_2 = -2b_2,\ g_1 = 0,\ g_2 = 3,\ c_1 = 1,\ c_2 = 2-b_2]$
- The validity condition (derived from QE over the reals) becomes:
  $$\boxed{-r_1 b_2 \frac{2}{a_2} + 3r_2 + 2 - b_2 \frac{1}{a_2} < a_2\mathbf{n} + b_2.}$$
- Evaluating at $b_2 = 0$ and $a_2 = 1$, we obtain $f_1 = 2\mathbf{i} + 1$ and $f_2 = 3\mathbf{j} + 2$
- $\max(i) = r_1,\ max(j) = r_2$
- Assume $\mathbf{n} = 10$, that is, $\tilde{A}[][10]$,
  - $r_1 = r_2 = 1,\ max(f_2) = 5 < 10,$ delinearization valid.
  - $r_1 = r_2 = 2,\ max(f_2) = 8 < 10,$ it is delinearization valid.
  - $r_1 = r_2 = 3,\ max(f_2) = 11 > 10,$ it is **invalid** delinearization unless a larger value of b_2 is chosen, since $1 < 6b_2$ must hold when we have $\mathbf{n} = 10, r_1 = r_2 = 3$.

The 2D-2D case: (truncated) triangular domain

Loop Format:

for $(\mathbf{i} = 0; \mathbf{i} \le r_1; \mathbf{i}{+}{+})$ **do**
  for $(\mathbf{j} = 0; \mathbf{i} + 2\mathbf{j} \le r_2; \mathbf{j}{+}{+})$ **do**
    $A[2 * \mathbf{i} * \mathbf{n} + \mathbf{n} + 3 * \mathbf{j} + 2] = \cdots$
  end for
end for

- $[a_2 = a_2,\ b_2 = b_2,\ e_1 = 2,\ e_2 = -2b_2,\ g_1 = 0,\ g_2 = 3,\ c_1 = 1,\ c_2 = 2{-}b_2]$
- Evaluating at $b_2 = 0$ and $a_2 = 1$, we obtain $f_1 = 2\mathbf{i} + 1$ and $f_2 = 3\mathbf{j} + 2$
- Assume $\mathbf{n} = 10$, that is, $\tilde{A}[\,][10]$,
  - $r_1 = r_2 = 1,\ max(f_2) = 3 < 10,$ delinearization valid.
  - $r_1 = r_2 = 2,\ max(f_2) = 5 < 10,$ delinearization valid.
  - $r_1 = r_2 = 3,\ max(f_2) = 5 < 10,$ delinearization valid.

Plan

The 3D-3D case: set up

Loop counters and dimension sizes

1. $d = 3$, that is, 3 loop counters: $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$.
2. $\delta = 3$, that is, the target array $\tilde{A}$ is 3D with dimension sizes $M_1 = a_1\mathbf{m} + b_1$, $M_2 = a_2\mathbf{n} + b_2$, and $M_3 = a_3\mathbf{p} + b_3$, where $\mathbf{m}$, $\mathbf{n}$, $\mathbf{p}$ are data parameters known at execution while $a_1, b_1, a_2, b_2, a_3, b_3$ are integers TBD.

Array references

1. Given the reference $A[R]$ to the array A with

$$\begin{aligned} R = & T_1\mathbf{inp} + T_5\mathbf{in} + T_9\mathbf{ip} + T_{13}\mathbf{i} + \\ & T_2\mathbf{jnp} + T_6\mathbf{jn} + T_{10}\mathbf{jp} + T_{14}\mathbf{j} + \\ & T_3\mathbf{knp} + T_7\mathbf{kn} + T_{11}\mathbf{kp} + T_{15}\mathbf{k} + \\ & T_4\mathbf{np} + T_8\mathbf{n} + T_{12}\mathbf{p} + T_{16} \end{aligned} \tag{3}$$

2. we want f_1, f_2, f_3 such that

$$R = f_1 M_2 M_3 + f_2 M_3 + f_3, \text{ where:}$$

$$f_1 = e_1\mathbf{i} + g_1\mathbf{j} + h_1\mathbf{k} + c_1,\ f_2 = e_2\mathbf{i} + g_2\mathbf{j} + h_2\mathbf{k} + c_2 \text{ and } f_3 = e_3\mathbf{i} + g_3\mathbf{j} + h_3\mathbf{k} + c_3$$

3. and for each iteration $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ in the domain we have

$$0 \le f_2 < M_2 \quad \text{and} \quad 0 \le f_3 < M_3.$$

The 3D-3D case: the polynomial system

The equation $R = f_1 M_2 M_3 + f_2 M_3 + f_3$ leads to the following system of polynomial equations:

$$\left\{\begin{array}{rcl} T_1 & = & a_2 a_3 e_1 \\ T_2 & = & a_2 a_3 g_1 \\ T_3 & = & a_2 a_3 h_1 \\ T_4 & = & a_2 a_3 c_1 \\ T_5 & = & a_2 b_3 e_1 \\ T_6 & = & a_2 b_3 g_1 \\ T_7 & = & a_2 b_3 h_1 \\ T_8 & = & a_2 b_3 c_1 \\ T_9 & = & a_3 b_2 e_1 + a_3 e_2 \\ T_{10} & = & a_3 b_2 g_1 + a_3 g_2 \\ T_{11} & = & a_3 b_2 h_1 + a_3 h_2 \\ T_{12} & = & a_3 b_2 c_1 + a_3 c_2 \\ T_{13} & = & b_2 b_3 e_1 + b_3 e_2 + e_3 \\ T_{14} & = & b_2 b_3 g_1 + b_3 g_2 + g_3 \\ T_{15} & = & b_2 b_3 h_1 + b_3 h_2 + h_3 \\ T_{16} & = & b_2 b_3 c_1 + b_3 c_2 + c_3 \end{array}\right. \quad (4)$$

- Recall that the coefficients $T_1, \ldots, T_{16}$ of R over **i**, **j**, **k**, **m**, **n**, **p** are known at compile time.
- Recall that the coefficients $a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3, e_1, e_2, e_3, g_1, g_2, g_3, h_1, h_2, h_3$ are integers TBD.

The 3D-3D case: solving the polynomial system

We obtain the following solution

$$\begin{cases} b_3 &= \frac{a_3 T_5}{T_1} \\ e_1 &= \frac{T_1}{a_2 a_3} \\ g_1 &= \frac{T_2}{a_2 a_3} \\ h_1 &= \frac{T_3}{a_2 a_3} \\ c_1 &= \frac{T_4}{a_2 a_3} \\ e_2 &= \frac{T_9 - a_3 b_2 e_1}{a_3} \\ g_2 &= \frac{T_{10} - a_3 b_2 g_1}{a_3} \\ h_2 &= \frac{T_{11} - a_3 b_2 h_1}{a_3} \\ c_2 &= \frac{T_{12} - a_3 b_2 c_1}{a_3} \\ e_3 &= T_{13} - (b_2 b_3 e_1 + b_3 e_2) \\ g_3 &= T_{14} - (b_2 b_3 g_1 + b_3 g_2) \\ h_3 &= T_{15} - (b_2 b_3 h_1 + b_3 h_2) \\ c_3 &= T_{16} - (b_2 b_3 c_1 + b_3 c_2) \end{cases} \tag{5}$$

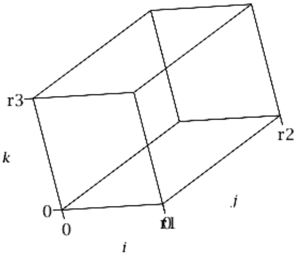
where

1. a_2, a_3 are free as long as $a_2 a_3$ divides $\gcd(T_1, T_2, T_3, T_4)$ and that $a_2 b_3$ divides $\gcd(T_5, T_6, T_7, T_8)$,
2. b_2 is free, and we have $\boxed{\frac{T_5}{T_1} = \frac{T_6}{T_2} = \frac{T_7}{T_3} = \frac{T_8}{T_4}.}$

The 3D-3D case: solving the QE problem

Principles: similar to the 2D-2D case

1. Inspect the different shapes of the $\mathbb{Z}$-polyhedra defining the iteration domain
2. for each shape, one con solve the problem off-line over the reals, keeping in mind that the obtained condition may be too strict.
3. for each shape, one can solve the problem at execution time by optimizing sawtooth functions.

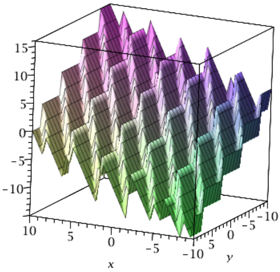


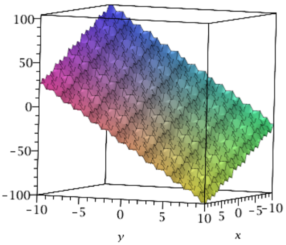
The 3D-3D case: using sawtooth functions

For the 3D-3D cases, the “sawtooth” function would be in the form of

$$f(x,y) = e\lfloor \frac{ax+b}{d} \rfloor + cx + h\lfloor \frac{kx+ly+g}{m} \rfloor$$

Here are two examples of the plot for the function.





The 3D-3D case: solving the QE problem over the reals

```
                         t_5 a_3    t_1     a_2 t_9 - b_2 t_1  t_1 t_13 - t_5 t_9
[a_1, a_2, a_3, b_1, b_2, -------, -------, -----------------, -----------------,
                            t_1    a_2 a_3       a_2 a_3              t_1

     t_2    a_2 t_10 - b_2 t_2  t_1 t_14 - t_10 t_5    t_1    a_2 t_11 - b_2 t_1
   -------, ------------------, -------------------, -------, ------------------,
   a_2 a_3       a_2 a_3                t_1          a_2 a_3       a_2 a_3

   t_1 t_15 - t_11 t_5    t_1    a_2 t_12 - b_2 t_1  t_1 t_16 - t_12 t_5
   -------------------, -------, ------------------, -------------------]
           t_1          a_2 a_3       a_2 a_3                t_1

>
> validity_condition_1 := `<`(r_1 * e_2 + r_2 *g_2 + r_3 * h_2 + c_2, a_2 * m + b_2);

      validity_condition_1 := e_2 r_1 + g_2 r_2 + h_2 r_3 + c_2 < a_2 m + b_2

> eval(validity_condition_1, [e_2=  (a_2 * t_9 - b_2 * t_1)   / (a_2 * a_3),
>                             g_2 = (a_2 * t_10 - b_2 * t_2) / (a_2 * a_3),
>                             h_2 =   ( a_2 * t_11 - b_2 * t_1)  / (a_2 * a_3)   ]);

r_1 (a_2 t_9 - b_2 t_1)   r_2 (a_2 t_10 - b_2 t_2)   r_3 (a_2 t_11 - b_2 t_1)
----------------------- + ------------------------ + ------------------------ + c_2
        a_2 a_3                   a_2 a_3                    a_2 a_3

     < a_2 m + b_2
```

The 3D-3D case: rectangular domain

Loop Format:

for $(i = 0; i \le r_1; i{+}{+})$ **do**
  for $(j = 0; j \le r_2; j{+}{+})$ **do**
    for $(k = 0; k \le r_3; k{+}{+})$ **do**
      $A[12\mathbf{np}i + 18\mathbf{np}j + 6\mathbf{np} + 6\mathbf{n}i + 9\mathbf{n}j + 16\mathbf{p}i + 22\mathbf{p}j + 3\mathbf{n} + 8\mathbf{p} + 8i + 11j + k + 4] = \cdots$
    end for
  end for
end for

- $T := [12, 18, 0, 6, 6, 9, 0, 3, 16, 22, 0, 8, 8, 11, 1, 4];$
- After solving the delinearization problem, we would get the validity conditions:

$$r_1(16 - 4b_2) + r_2(22 - 6b_2) - 4r_3b_2 + 8 - 4b_2 < b_2 + 3\mathbf{m}$$

  and

$$r_3 < \mathbf{p} + 1$$

- We can successfully retrieve $a_2 = 3, a_3 = 2$ and $b_3 = 1$.
- (The real array format for this problem should be $A[2 * \mathbf{m} + 1][3 * \mathbf{n} + 2][2 * \mathbf{p} + 1]$)

Plan

Another approach to the delinearization problem

Principles

1. Assume that the delinearization problem has been solved for a particular problem instance, say 2D-Jacobi.
2. Assume that we have another problem instance which looks very similar
3. We may want to check whether the solved problem instance is obtained from the unsolved problem instance via a *rank-preserving unimodular transformation* (between the two iteration domains)
4. If this the case, then the validity condition of the solved problem instance and the validity condition of the unsolved problem instance are equivalent.

Details

1. *rank-preserving* guarantees that the same array coefficients are read/written in the same order.
2. rank-preserving transformations are "classifiable" off-line, next slide.
3. *unimodularity* guarantees that we can map integers to integers back and forth.
4. This can be performed at compile time (at the simple cost of linear algebra) and leads to a case discussion which can be evaluated at execution time.

2D Pattern matching problem

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} ai + bj \\ ci + dj \end{bmatrix}$$

QE input:

$$\forall [i_1, j_1, i_2, j_2], (i_1 < i_2) \vee ((i_1 = i_2) \wedge (j_1 < j_2)) \implies$$
$$(a\, i_1 + b\, j_1 < a\, i_2 + b\, j_2) \vee ((a\, i_1 + b\, j_1 = a\, i_2 + b\, j_2) \wedge (c\, i_1 + d\, j_1 < c\, i_2 + d\, j_2))$$

QE output:

$$(b = 0) \wedge (0 < a) \wedge (0 < d)$$

We also assume that $ad - bc = 1$, which gives us the final matrix as below where $a > 0$

$$\begin{bmatrix} a & 0 \\ c & \frac{1}{a} \end{bmatrix}$$

3D Pattern matching problem

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & l \end{bmatrix} \times \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} ai + bj + ck \\ di + ej + fk \\ gi + hj + lf \end{bmatrix}$$

QE input:

$$\begin{aligned}
&\forall [i_1, j_1, k_1, i_2, j_2, k_2], \\
&(i_1 < i_2) \vee ((i_1 = i_2) \wedge (j_1 < j_2)) \vee ((i_1 = i_2) \wedge (j_1 = j_2) \wedge (k_1 < k_2)) \implies \\
&\quad (a\, i_1 + b\, j_1 + c\, k_1 < a\, i_2 + b\, j_2 + c\, k_2) \\
&\quad \vee((a\, i_1 + b\, j_1 + c\, k_1 = a\, i_2 + b\, j_2 + c\, k_2) \wedge (d\, i_1 + e\, j_1 + f\, k_1 < d\, i_2 + e\, j_2 + f\, k_2)) \\
&\quad \vee((a\, i_1 + b\, j_1 + c\, k_1 = a\, i_2 + b\, j_2 + c\, k_2) \wedge (d\, i_1 + e\, j_1 + f\, k_1 = d\, i_2 + e\, j_2 + f\, k_2) \\
&\qquad \wedge(g\, i_1 + h\, j_1 + l\, k_1 < g\, i_2 + h\, j_2 + l\, k_2))
\end{aligned}$$

QE output:

$$(f = 0) \wedge (0 < e) \wedge (c = 0) \wedge (b = 0) \wedge (0 < a) \wedge (0 < l)$$

which gives us the final matrix as below

$$\begin{bmatrix} a > 0 & 0 & 0 \\ c & e > 0 & 0 \\ g & h & l > 0 \end{bmatrix}$$

Plan

Concluding remarks: 4 or 5 approaches (1/2)

Using Presburger arithmetic at compile time

1. the computed validity conditions are sufficient and necessary
2. extra work at compile-time (the underlying algorithms are exponential in d and δ)
3. the implementation in the Polly framework seems not to handle some corner cases like $b_2 \neq 0$

Using QE over the reals off-line

1. the computed validity conditions are only sufficient, but still useful in practice as shown above
2. no extra work at compile-time
3. can determine a_2, a_3, b_2 at execution time and support automatic case discussion at execution time to ensure delinearization.

Concluding remarks: 4 or 5 approaches (1/2)

Using sawtooth functions

1. the computed validity conditions are sufficient and necessary
2. little extra work at execution time
3. combined with polynomial system solving (performed off-line) it can be used to determine a_2, a_3, b_2 at execution time, provided that we replace the optimization problem by a parametric one (work in progress).

Pattern matching via rank-preserving unimodular transformations

1. As for QE, most of the work can be done off-line
2. At compile-time, only linear system solving is needed, which can be regarded as cheap.
3. the computed validity conditions are sufficient and necessary
4. but this approach is heuristical
5. however, it is reasonable to think of building a good set of popular patterns.