Lecture 1: Polyhedra Model and Nonsingular Transformations

Chaofan Lin, TA of Advanced Compiler

Spring 2023

1 Polyhedra Model

1.1 Perfectly Nested Loops

For simplicity, we only consider a well-formed type of loops in this Chapter, which is called **perfectly nested loops**.

In general a n-dim perfectly nested loop has the form (**Definition 1**):

$$\begin{aligned}
&\text{do } i_1 = L_1, U_1 \\
&\quad \text{do } i_2 = L_2, U_2 \\
&\quad\quad \cdots \\
&\quad\quad\quad \text{do } i_n = L_n, U_n \\
&\quad\quad\quad\quad \mathcal{S}(i_1, i_2, \ldots, i_n)
\end{aligned}$$

For every loop statement, it has a loop index i_k, a lower bound L_k, a upper bound U_k and a step size (In this example the step size is 1 so it can be omitted). The semantic is similar with for-loop in popular programming languages.

A prominent feature is that the statement $\mathcal{S}$ is only in the **innermost loop**. So you can not have

$$\begin{aligned}
&\text{do } i_1 = L_1, U_1 \\
&\quad \mathcal{S}_1(i_1) \\
&\quad \text{do } i_2 = L_2, U_2 \\
&\quad\quad \mathcal{S}_2(i_1, i_2)
\end{aligned}$$

And there are some other constraints on the perfectly nested loops:

- For the k-th loop, $L_k(U_k)$ must be a maxima(minima) of affine expressions of the outer k - 1 loop indices $i_1, \ldots, i_{k-1}$ with rational constant coefficients. Problem size parameters (E.g. N, M in the OI problem) can be also viewed as a type of constants.
- Besides, ceiling / floor operators are also allowed in the L_k/U_k to make the bound a integer.

1.2 Iteration Space

An iteration is an instance of the innermost statement in our perfectly nested loops model. For example, if you have a 2-dim loop, each dim is just for 1 to N, then you get N^2 instances of the

innermost statement.

An iteration can be fully identified by all n loop indices. So we get an **iteration vector** of an iteration (**Definition 2**):

$$\vec{i} = (i_1, i_2, \ldots, i_n)$$

(**Definition 3**) the iteration space **I** is the set of all loop iteration vectors:

$$\mathbf{I} = \{(i_1, i_2, \ldots, i_n) \mid \forall 1 \leq k \leq n : L_k \leq i_k \leq U_k\}$$

If we put them in the $\mathbb{R}^n$ space, we can view every iteration as a **integer point** in the space. And the iteration space is just a set of points. Usually we will absue the language slightly by assuming implicitly that **I** is a subset of $\mathbb{R}^n$ (Even though it only contains integer points!).

Recall that we require max only presents in L_k and min only presents in U_k, which is the constraint of perfectly nested loop. This makes the iteration space "convex":

(Theorem 1) The iteration space **I** (as a subset of $\mathbb{R}^n$) is convex in $\mathbb{R}^n$, i.e. for any $x, y \in \mathbf{I}$ and $\lambda \in [0, 1]$, $\lambda x + (1 - \lambda)y \in \mathbf{I}$.

Proof. First let's not consider max/min presenting in bounds. Then it's just a n-dim rectangle which is certainly convex. And because the intersection of convex sets is also convex, **I** is convex. □

According its convexity, it can be written as the following **normalised form**:

$$\mathbf{I} = \{\vec{i} \mid \mathcal{B}\vec{i} \leq \vec{b}\}$$

where $\mathcal{B}$ is a $m \times n$ matrix. n is the dimension of the loop and m depends on the bounds.

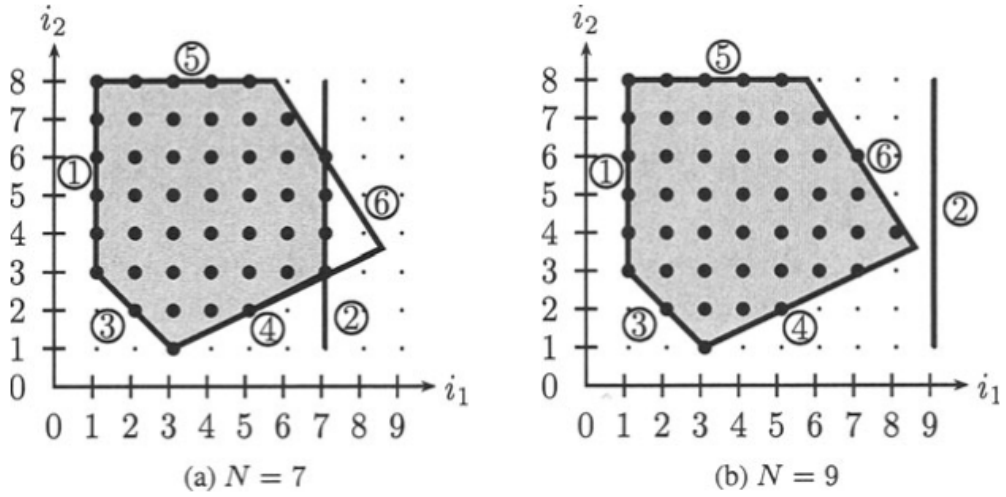
(**Example 1**) Let's look at an example of a double loop.

$$\begin{aligned}
&\text{do } i_1 = 1, N \\
&\quad \text{do } i_2 = \max(-i_1 + 4, \lceil \frac{i_1 - 1}{2} \rceil), \min(8, \lfloor \frac{-3i_1 + 33}{2} \rfloor) \\
&\quad\quad A(i_1, i_2) = A(i_1 - 1, i_2) + A(3i_1, i_2 + 1)
\end{aligned}$$

Its iteration space is an area constrained by six inequalities:

$$\mathbf{I} = \left\{ \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} \;\middle|\; \overbrace{1 \leqslant i_1}^{①}, \overbrace{i_1 \leqslant N}^{②}, \overbrace{-i_1 + 4 \leqslant i_2}^{③}, \overbrace{(i_1 - 1)/2 \leqslant i_2}^{④}, \overbrace{i_2 \leqslant 8}^{⑤}, \overbrace{i_2 \leqslant (-3i_1 + 33)/2}^{⑥} \right\}$$

And in $\mathbb{R}^2$ it looks like as follows. Note that as the problem size parameter N changes, the shape of iteration space changes probably.



(a) $N = 7$ (b) $N = 9$

We can write its normalised form:

$$\mathbf{I} = \left\{ \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} \middle| \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ -1 & -1 \\ 1 & -2 \\ 0 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} \leq \begin{pmatrix} -1 \\ N \\ -4 \\ 1 \\ 8 \\ 33 \end{pmatrix} \right\}$$

2 Nonsingular Transformations

2.1 Introduction

To learn loop transformations, we start from a simple transformation: **nonsingular transformations**. It's a type of **iteration reordering transformation**. For every iteration reordering transformation, we have a reordering mapping $\varphi : \mathbf{I} \mapsto \mathbf{I}'$.

(Definition 4) $T \in \mathbb{Z}^{n \times n}$ is nonsingular (i.e. full-rank). Then the nonsingular transformation defined by this matrix is

$$\varphi : \mathbf{I} \mapsto \mathbf{I}', \varphi(\vec{i}) = T\vec{i}$$

It's obvious that it is a linear transformation and the reordering mapping is injective. It will reorder the iterations because for an iteration space, in each loop indices we always go from lower bound to upper bound.

2.2 Some Math Preparation

Given that we are studying integer matrices, there are some important concepts and lemmas needed to be introduced. First let's consider the "full-rank matrix" in integer matrices, unimodular matrices.

(**Definition 5**) A unimodular matrix is an square integer matrix A with $\det(A) = \pm 1$.

An important property is that it is closed in terms of inverse.

(**Lemma 1**) For an integer matrix A,

$$A \text{ is unimodular} \iff A^{-1} \text{ is an integer matrix.}$$

Proof. $\implies$: First $\det(A) \neq 0$ so A is inversible. And $A^{-1} = \frac{A^*}{\det(A)}$ is surely an integer matrix.
$\impliedby$: $\det(A)\det(A^{-1}) = \det(I_n) = 1$. □

And naturally we have A is unimodular $\iff$ A^{-1} is unimodular. To show why unimodular matrices are similar with full-rank matrices, we need to introduce "spanning space" in the integer world, which is called **lattices**.

(**Definition 6**) The lattice of a matrix $A = [\vec{a}_1, \ldots, \vec{a}_n]$ is defined as

$$\mathcal{L}(A) = \{\lambda_1\vec{a}_1 + \cdots + \lambda_n\vec{a}_n \mid \lambda_1, \ldots, \lambda_n \in \mathbb{Z}\}$$

A interesting fact is that multiplying a unimodular matrix will not affect its lattice.

(**Lemma 2**) For any matrix A and any unimodular matrix U,

$$\mathcal{L}(A) = \mathcal{L}(AU)$$

Proof.

$$\mathcal{L}(A) \supseteq \mathcal{L}(AU) \supseteq \mathcal{L}(AUU^{-1})$$

□

Then we have a simple corollary

(**Corollary of Lemma 2**) For any unimodular matrix U,

$$\mathcal{L}(U) = \mathbb{Z}^n$$

Proof.

$$\mathcal{L}(U) = \mathcal{L}(UI_n) = \mathcal{L}(I_n) = \mathbb{Z}^n$$

□

And finally we have an important theorem which we will not give the proof here.

(**Theorem 2, Hermite Normal Form**) For every nonsingular integer matrix $A \in \mathbb{Z}^{n\times n}$, there is a unimodular matrix U such that

$$AU = H$$

where H is a nonnegative lower triangular matrix, in which each row has a unique maximum component, which is located on its main diagonal.

Here both U and H are unique. And if A is unimodular, then $H = I_n$.

2.3 Transformed Iteration Space

Suppose the original iteration space is

$$\mathbf{I} = \{\mathcal{B}\vec{i} \leq \vec{b}\}$$

Then one may naturally assume that the transformed iteration space $\mathbf{I}'$ is

$$\mathbf{I}' = \{\mathcal{B}T^{-1}\vec{i'} \leq \vec{b}\}$$

It is close but not all integer points in this region can find its corresponding original image, $\vec{i}$. Given that $\vec{i}$ is an integer vector, it requires that $\vec{i'} \in \mathcal{L}(T)$, so the answer is

$$\mathbf{I}' = \mathcal{L}(T) \cap \{\mathcal{B}T^{-1}\vec{i'} \leq \vec{b}\}$$

But if T is unimodular (We call it **unimodular transformation**), then $\mathcal{L}(T) = \mathbb{Z}^n$ so the first assumption is correct. In this case, the transformed iteration space is convex. But in general there are examples where $\mathbf{I}'$ is not convex.

2.4 Code Generation Algorithm

Code generation is the process converting transformed iteration space to the final loop code. A general method to generate nonsingular transformed loops is proposed by Xue, 1994. It contains

four steps. The generated loop has the form

$$\begin{aligned}
&\text{do } i_1 = L_1' + \delta_1', U_1', step_1' \\
&\quad \text{do } i_2 = L_2' + \delta_2', U_2', step_2' \\
&\quad \dots \\
&\quad\quad \text{do } i_n = L_n' + \delta_n', U_n', step_n' \\
&\quad\quad\quad \begin{pmatrix} i_1 \\ \vdots \\ i_n \end{pmatrix} = T^{-1} \begin{pmatrix} i_1' \\ \vdots \\ i_n' \end{pmatrix} \\
&\quad\quad\quad \mathcal{S}(i_1, i_2, \dots, i_n)
\end{aligned}$$

Here δ_k' is called **lower bound offset**. There are offsets and step-size changes after transforming because we need to intersect with $\mathcal{L}(T)$.

2.4.1 New Lower/Upper Bound

After transformation, we need to solve the inequalities system to find bounds for each loop index. There is a powerful algorithm called **Fourier-Motzkin elimination** to do this.

(**Algorithm 1, Fourier-Motzkin elimination**) Suppose we have an inequality system $P(x_1, x_2, \dots, x_n) = \{\vec{x} \mid A\vec{x} \leq \vec{b}\}$. We want find the explicit representation of $P(x_1, x_2, \dots, x_{n-1})$ such that $P(x_1, x_2, \dots, x_{n-1}) = \{\vec{x}[1 \dots n-1] \mid A\vec{x} \leq \vec{b}\}$.
To eliminate x_n, we first divide all inequalities in the system into three categories:

- 1. Those that are independent of x_n,
- 2. Those that are lower bounds of x_n,
- 3. Those that are upper bounds of x_n.

We preserve all Type 1 inequalities. And for every pair of Type 2 and 3,

$$x_n \geq l(x_1, \dots, x_{n-1}),\ x_n \leq u(x_1, \dots, x_{n-1})$$

we construct a new inequality to the system

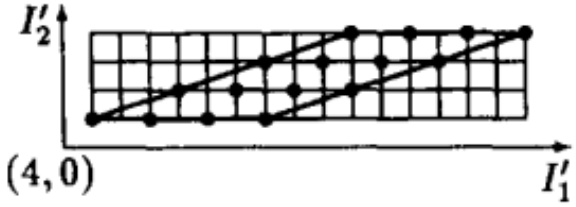
$$l(x_1, \dots, x_{n-1}) \leq u(x_1, \dots, x_{n-1})$$

Then the new system is free of x_n and satisfies our requirements.

We can first apply $n-1$ times FME and get the bounds of i_1', then go back to get the bounds of $i_2', \dots, i_n'$.

2.4.2 New Step Size

Given that the existence of $\mathcal{L}(T)$, there may be some "holes" in the transformed iteration space, which makes the new loop step size no longer be 1. As the following shows, it should use step size 2 in the loop index I'_1.



Considering what step size we should take in k-th loop. We just need to consider what step size can generate the following one-dim lattice (Why?)

$$\mathcal{L}_k(T) := \{i'_k \mid (0, \ldots, 0, i'_k, \ldots, i'_n) \in \mathcal{L}(T)\}$$

Because $\mathcal{L}(T) = \mathcal{L}(H)$, it is more convenient for us to consider a lower triangle matrix. Since the first $k-1$ indices are zeros, the coefficients before these rows are enforced to be zeros two. Then we have $i'_k = \lambda H_{k,k}$ where $\lambda \in \mathbb{Z}$.

This means we have

$$\mathcal{L}_k(T) = \mathcal{L}_k([H_{k,k}])$$

So the step size is exactly $H_{k,k}$ (The k-th value in the main diagonal of H).

2.4.3 New Lower Bound Offset

Because the lower bound L_k can't always fall in the lattice $\mathcal{L}(T)$, there is an offset between lower bound and the true start position. The core idea is to add a minimal number such that the points can fall in the lattice $\mathcal{L}(T)$. Given that it is a bit complicated, I will not introduce more here.

3 References

- Loop Tiling For Parallelism, Jingling Xue, 2000.
- Automating non-unimodular loop transformations for massive parallelism, Jingling Xue, Parallel Computing 20 (1994) 711-728.