Применение полиэдральной модели для оптимизации гнезд циклов

Молдованова Ольга Владимировна

ovm@sibguti.ru, ovm@isp.nsc.ru

Кафедра вычислительных систем, Сибирский государственный университет телекоммуникаций и информатики

Лаборатория вычислительных систем, Институт физики полупроводников им. А.В. Ржанова СО РАН

Семинар «Вычислительные системы»

Институт физики полупроводников им. А.В. Ржанова СО РАН

г. Новосибирск, 16 марта 2018 г.

Задача оптимизации гнезд циклов

- Значительная часть программ содержит вычисления в гнездах циклов
- Такие вычисления занимают большую часть времени выполнения программ
- Для уменьшения времени, затрачиваемого на циклы, необходимо:
  - ✓ выполнять вычисления в циклах параллельно (OpenMP, векторизация и т.п.)
  - ✓ увеличить локальность данных (размещение данных в кэш-памяти, уменьшение числа промахов при обращении к кэш-памяти и т.д.)
- Выполнение оптимизационных преобразований гнезд циклов при компиляции с использованием обычных промежуточных представлений (AST, SSA, CFG и т.п.) трудоемко и неэффективно
- В 1980-х гг. была разработана модель, основанная на алгебраическом представлении программ и их преобразований, — **полиэдральная модель (polyhedral model)**

Определения

Мир — четырехмерное аффинное пространство [...]
В.И. Арнольд
«Математические методы классической механики»

- **Аффинное пространство**

  Аффинное пространство − это множество $\mathcal{A}$ точек, для которого заданы:
  - некоторое линейное пространство V, над числовым полем $\mathbb{K}$, называемое ассоциированным с $\mathcal{A}$;
  - отображение $f : \mathcal{A} \times \mathcal{A} \to V$, которое ставит в соответствие каждой упорядоченной паре точек $A, B \in \mathcal{A}$ некоторый вектор из V, обозначаемый $\vec{a}$ (точка A − начало, точка B − конец вектора $\vec{a}$)
- **Аффинная гиперплоскость**

  Гиперплоскость в m-мерном аффинном пространстве является плоскостью размерности $m - 1$
- **Аффинная функция**

  Функция $f : \mathbb{K}^m \to \mathbb{K}^n$ является аффинной, если существуют вектор $\vec{b} \in \mathbb{K}^m$ и матрица $A \in \mathbb{K}^{m \times n}$ такие, что:
  $\forall \vec{x} \in \mathbb{K}^m, f(\vec{x}) = A\vec{x} + \vec{b}$
- **Полиэдр**

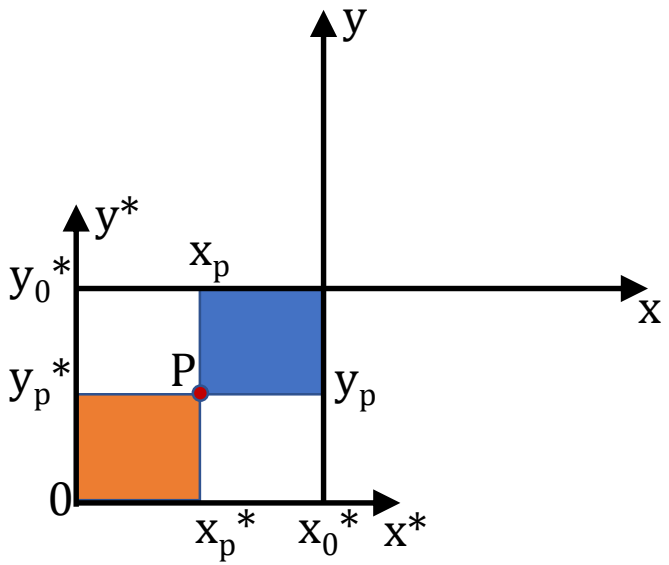
  Множество $\mathcal{P} \in \mathbb{K}^m$ является полиэдром, если существует конечная система неравенств $A\vec{x} \leq \vec{b}$ такая, что:
  $\mathcal{P} = \{\vec{x} \in \mathbb{K}^m \,|A\vec{x} \leq \vec{b}\}$
- **Параметрический полиэдр**

  Пусть $\vec{n}$ − вектор символических параметров, $\mathcal{P}$ является параметрическим полиэдром, если он определен как: $\mathcal{P} = \{\vec{x} \in \mathbb{K}^m \,|A\vec{x} \leq B\vec{n} + \vec{b}\}$
- **Аффинные преобразования**

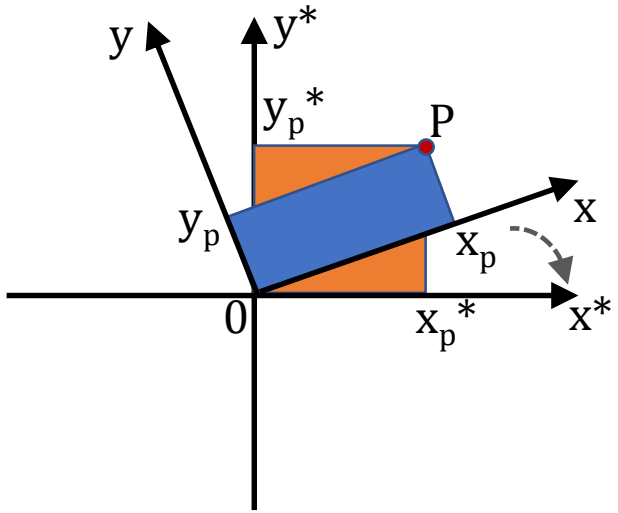
  Отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся в пересекающиеся, скрещивающиеся в скрещивающиеся (например, движение, сжатие/растяжение, преобразование подобия и др.)

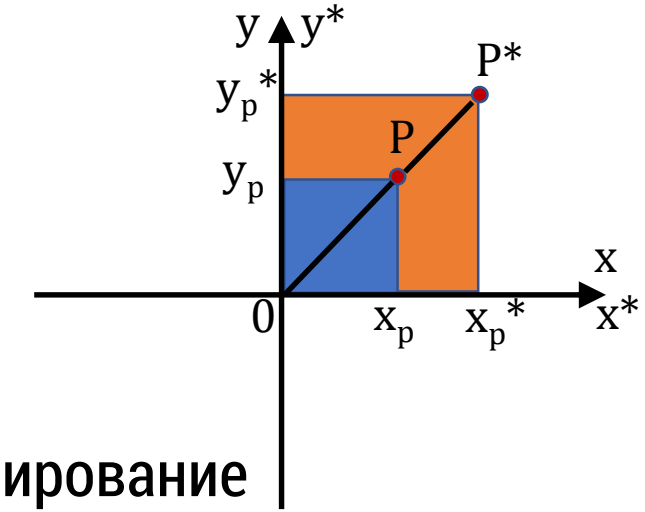
Примеры аффинных преобразований

Сдвиг

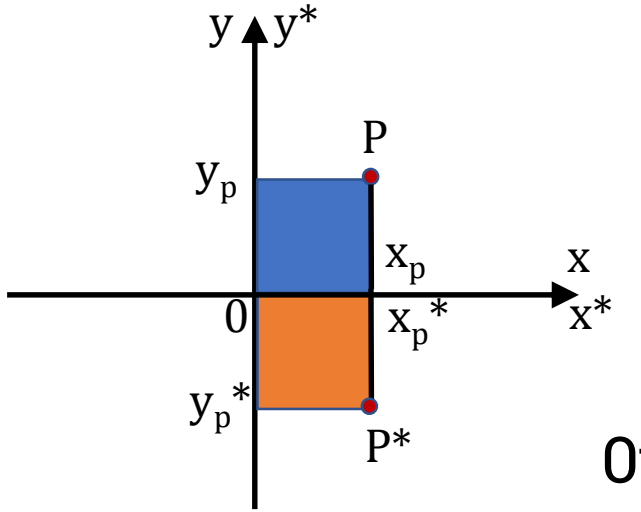


Поворот





Масштабирование



Отражение

Косников Ю.Н. *Геометрические преобразования в компьютерной графике. Конспект лекций.*
URL: http://window.edu.ru/resource/898/72898/files/stup559.pdf

Полиэдральная модель

Гнездо циклов (Loop Nest)

```
for i1 = L1,U1,T1
   ...
   for i2 = L2,U2,T2
      ...
      for ik = Lk,Uk,Tk
         ...
         for in = Ln,Un,Tn
            Body
            ...
      ...
   ...
...
```

```
S1
S2
...
Sj
...
Sm
```

i_k – счетчик циклов (loop iterator, loop counter)
T_k – шаг цикла (loop step)
L_k, U_k – границы цикла (loop bounds)

$\vec{i} = \begin{pmatrix} i_1 \\ i_2 \\ \dots \\ i_k \\ \dots \\ i_n \end{pmatrix}$ – вектор итераций (iteration vector)

$d = n$ – глубина вложенности циклов (loop depth)
$\mathbb{Z}^n$ – пространство итераций (iteration space)
S_j – выражение (statement)
$S(\vec{i})$ – экземпляр выражения (statement instance)

Полиэдр ($\mathbb{Z}$-полиэдр) – выпуклое множество точек, т.е. множество точек в векторном пространстве $\mathbb{Z}$, ограниченное аффинными неравенствами:

$I_s = \{\vec{i} \mid \vec{i} \in \mathbb{Z}^n, A\vec{i} + b \geq \vec{0}\}$ – область итераций (iteration domain)

$I_s \subseteq \mathbb{Z}^n$

Статическая контрольная часть

Максимальное множество последовательных выражений с полиэдральными областями итераций называется **статической контрольной частью** (Static Control Part, SCoP)

```
for i = 1,n
    S1
    for j = 1,i*i
        S2
        for k = 0,j
            if(j >= 2) then
                S3
            S4
    for p = 0,6
        S5
        S6
```

Статическая контрольная часть

Максимальное множество последовательных выражений с полиэдральными областями итераций называется **статической контрольной частью** (Static Control Part, SCoP)

```
for i = 1,n
  S1                                  SCoP1 содержит одно выражение S1 (non-rich)
  for j = 1,i*i
    S2
    for k = 0,j
      if(j >= 2) then
        S3
      S4
  for p = 0,6
    S5
    S6
```

Статическая контрольная часть

Максимальное множество последовательных выражений с полиэдральными областями итераций называется **статической контрольной частью** (Static Control Part, SCoP)

```
for i = 1,n
    S1
    for j = 1,i*i
        S2
        for k = 0,j
            if(j >= 2) then
                S3
            S4
    for p = 0,6
        S5
        S6
```

$SCoP_1$ содержит одно выражение S_1 (non-rich)

$SCoP_2$ содержит три выражения S_2,S_3,S_4 (rich)
параметры: i, j
счетчик цикла: k

Статическая контрольная часть

Максимальное множество последовательных выражений с полиэдральными областями итераций называется **статической контрольной частью** (Static Control Part, SCoP)

```
for i = 1,n
    S1
    for j = 1,i*i
        S2
        for k = 0,j
            if(j >= 2) then
                S3
            S4
    for p = 0,6
        S5
        S6
```

$SCoP_1$ содержит одно выражение S_1 (non-rich)

$SCoP_2$ содержит три выражения S_2,S_3,S_4 (rich)
параметры: i, j
счетчик цикла: k

$SCoP_3$ содержит два выражения S_5,S_6 (non-rich)

Статическая контрольная часть

Максимальное множество последовательных выражений с полиэдральными областями итераций называется **статической контрольной частью** (Static Control Part, SCoP)

```
for i = 1,n
    S1
    for j = 1,i*i
        S2
        for k = 0,j
            if(j >= 2) then
                S3
            S4
    for p = 0,6
        S5
        S6
```

n – глобальный параметр для $SCoP_1$, $SCoP_2$, $SCoP_3$

Предварительная обработка циклов

До обработки

```
n = 10;
for(i = 1; i <= m; i++)
   for(j = n*i; j <= n*m; j++)
      S1;
```

После обработки

```
for(i = 1; i <= m; i++)
   for(j = 10*i; j <= 10*m; j++)
      S1;
```

Распространение констант (Constant Propagation)

```
i = 1;
while(i <= m){
   S1;
   i = i + 1;
}
```

```
for(i = 1; i <= m; i++)
   S1;
```

Преобразование цикла `while` в цикл `for` (`While`-loop to `For`-loop Conversion)

```
for(i = 1; i <= m; i++)
   for(j = i; j <= n; j+=2)
      S1;
```

```
for(i = 1; i <= m; i++)
   for(jj = 1; jj <= (n-i+2)/2; jj++)
      S1(2*jj - 2 + i);
```

Нормализация цикла (Loop Normalization)

Bastoul C. *Improving Data Locality in Static Control Programs*.
URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bastoul_thesis.pdf

Предварительная обработка циклов

До обработки

```
for(i = 1; i <= m; i++)
   function(i,m);
```

После обработки

```
for(i = 1; i <= m; i++)
   for(j = 1; j <= n; j++)
      S1;
```

Встраивание функций (Inlining)

```
ind = 0;
for(i = 1; i <= 100; i++){
   for(j = 1; j <= 100; j++){
      ind = ind + 2;
      a[ind] = a[ind] + b[j];
   }
   c[i] = a[ind];
}
```

```
for(i = 1; i <= 100; i++){
   for(j = 1; j <= 100; j++)
      a[200*i+2*j-200] =
            a[200*i+2*j-200] + b[j];
   c[i] = a[200*i];
}
```

Подстановка индуктивных переменных (Induction Variable Substitution)

Bastoul C. *Improving Data Locality in Static Control Programs*.
URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bastoul_thesis.pdf

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

SCoP содержит одно выражение S

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

SCoP содержит одно выражение S

$d = 2$ – глубина вложенности

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

SCoP содержит одно выражение S

$d = 2$ – глубина вложенности

$\vec{i} = \begin{pmatrix} i \\ j \end{pmatrix}$ – вектор итераций

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

SCoP содержит одно выражение S

$d = 2$ – глубина вложенности

$\vec{i} = \begin{pmatrix} i \\ j \end{pmatrix}$ – вектор итераций

$\vec{n} = (n)$ – вектор глобальных параметров

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

SCoP содержит одно выражение S

$d = 2$ – глубина вложенности

$\vec{i} = \begin{pmatrix} i \\ j \end{pmatrix}$ – вектор итераций

$\vec{n} = (n)$ – вектор глобальных параметров

$\mathbb{Z}^2$ – пространство итераций

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 0, j = 0
```

Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
```

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 1, j = 0
```

Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0)
```

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 1, j = 1
```

Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
```

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 2, j = 0
```

Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0)
```

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 2, j = 1
```

Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1)
```

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 2, j = 2
```

Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
```

Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 3, j = 0...3
```

Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
```

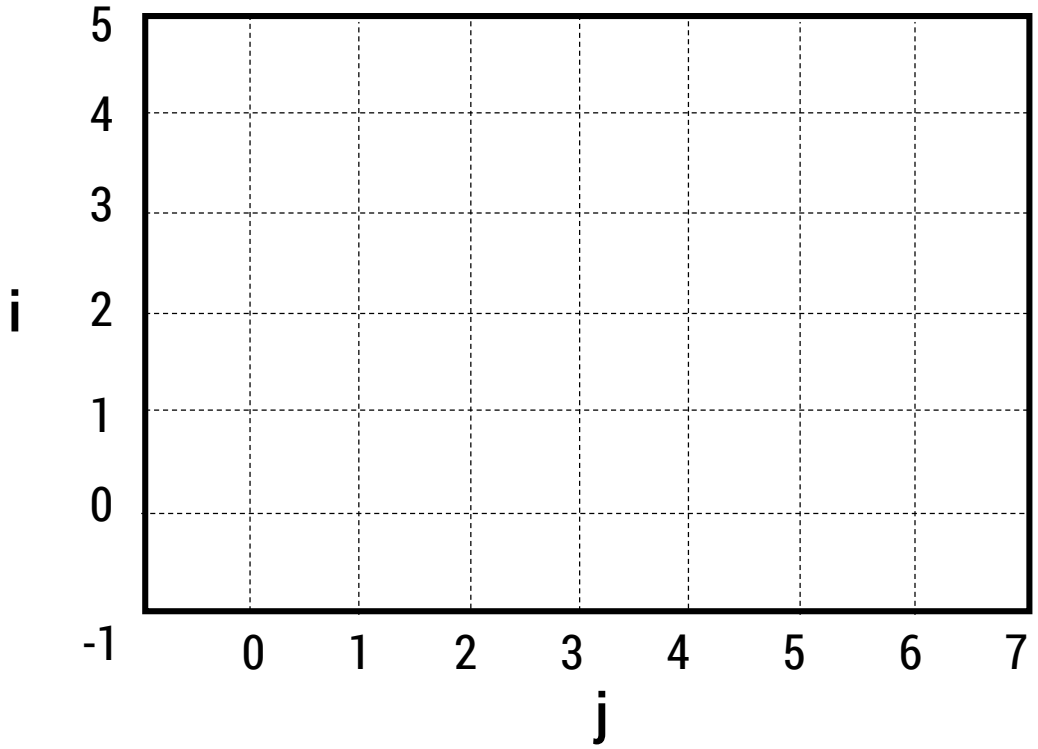
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

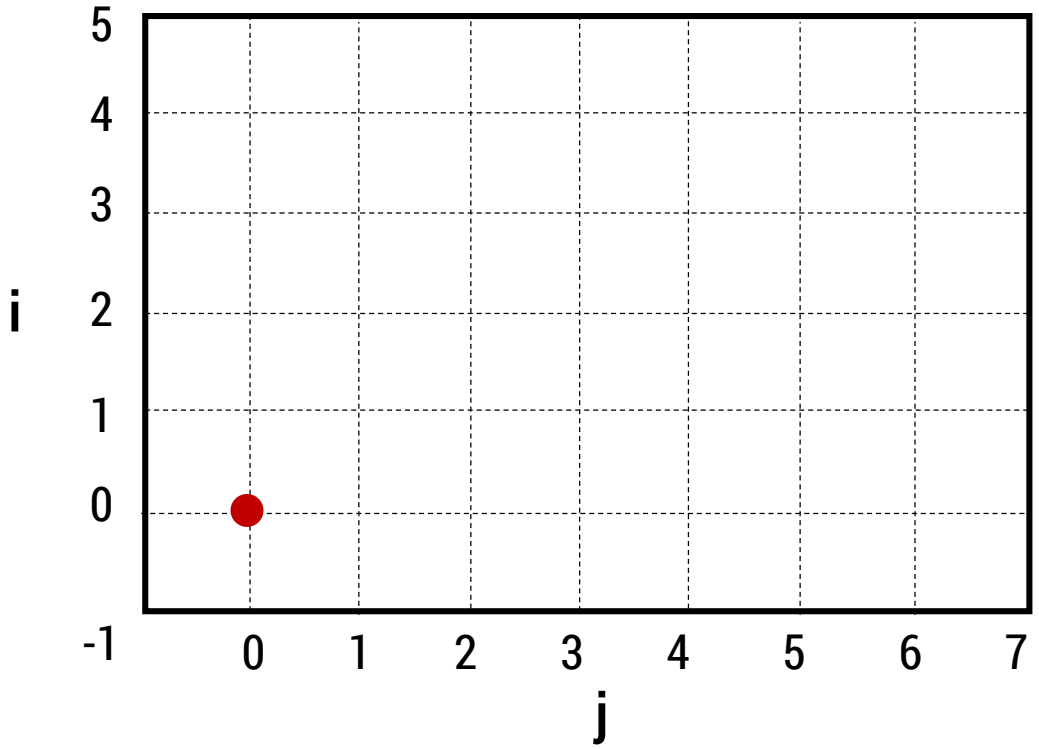
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

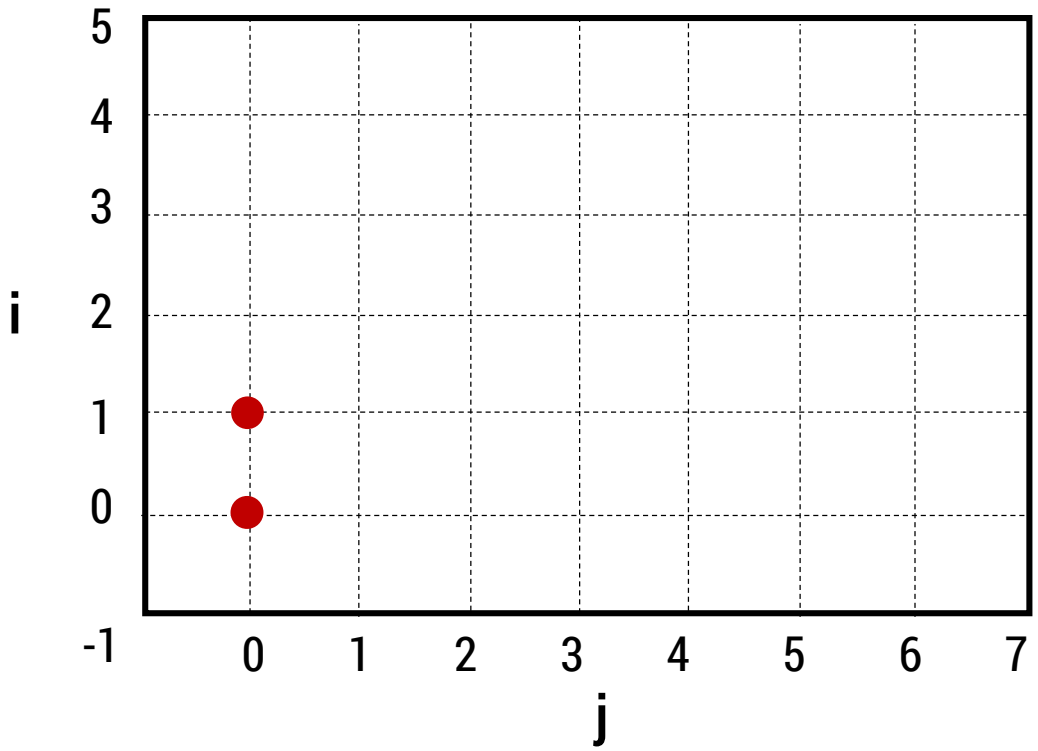
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

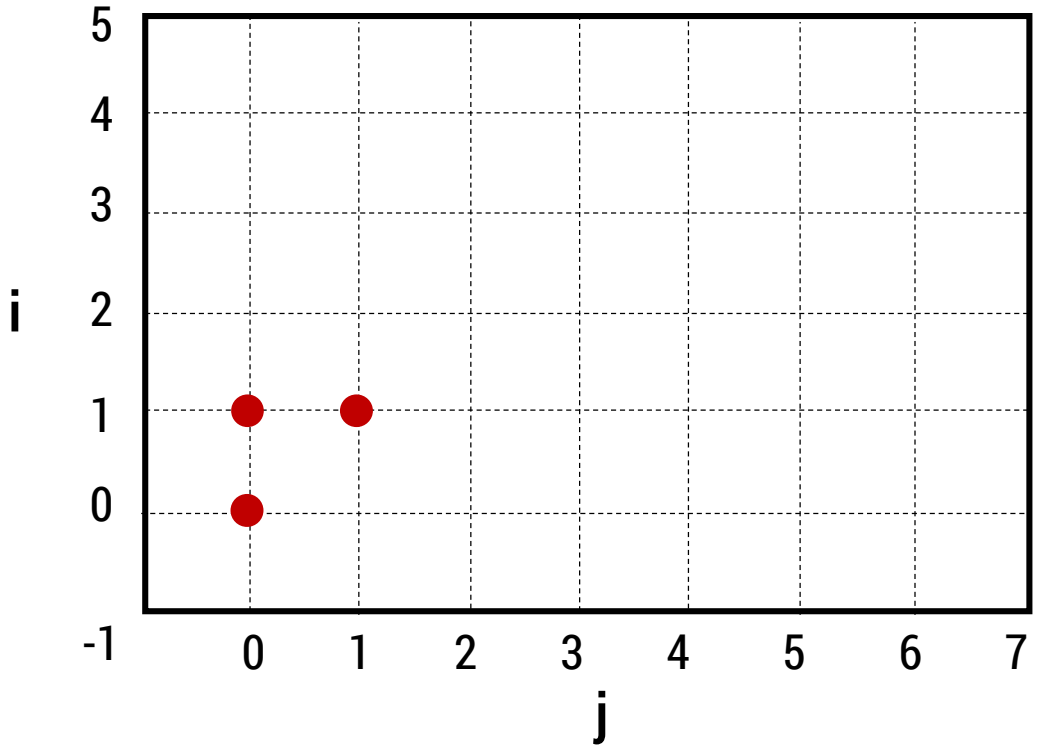
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

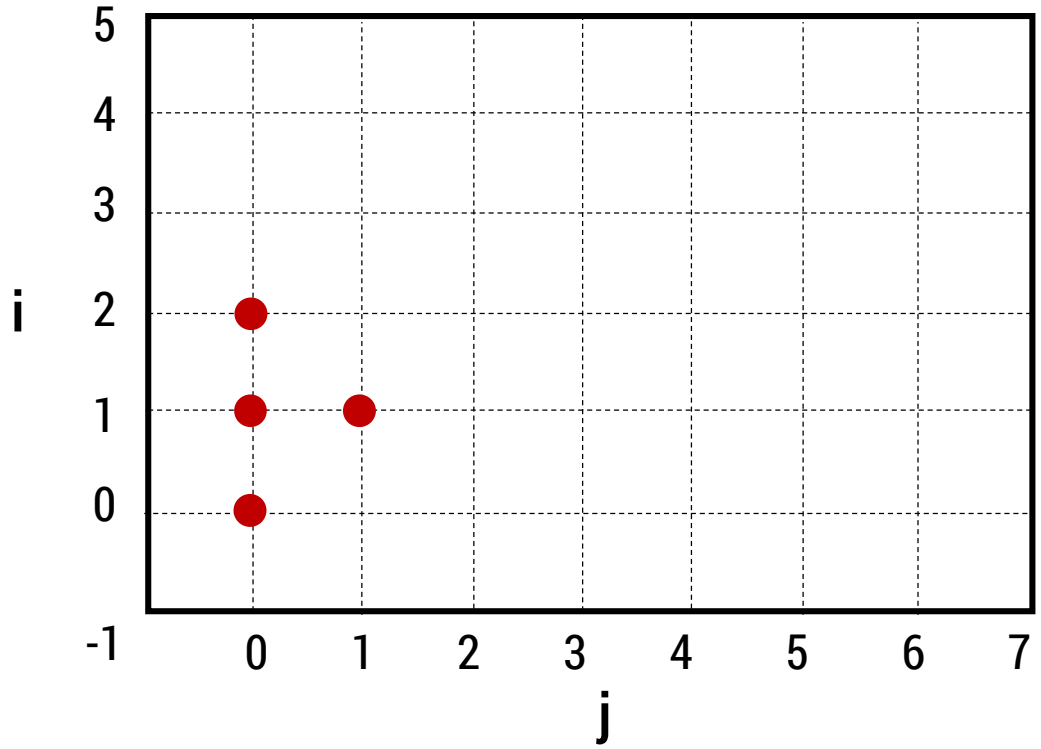
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

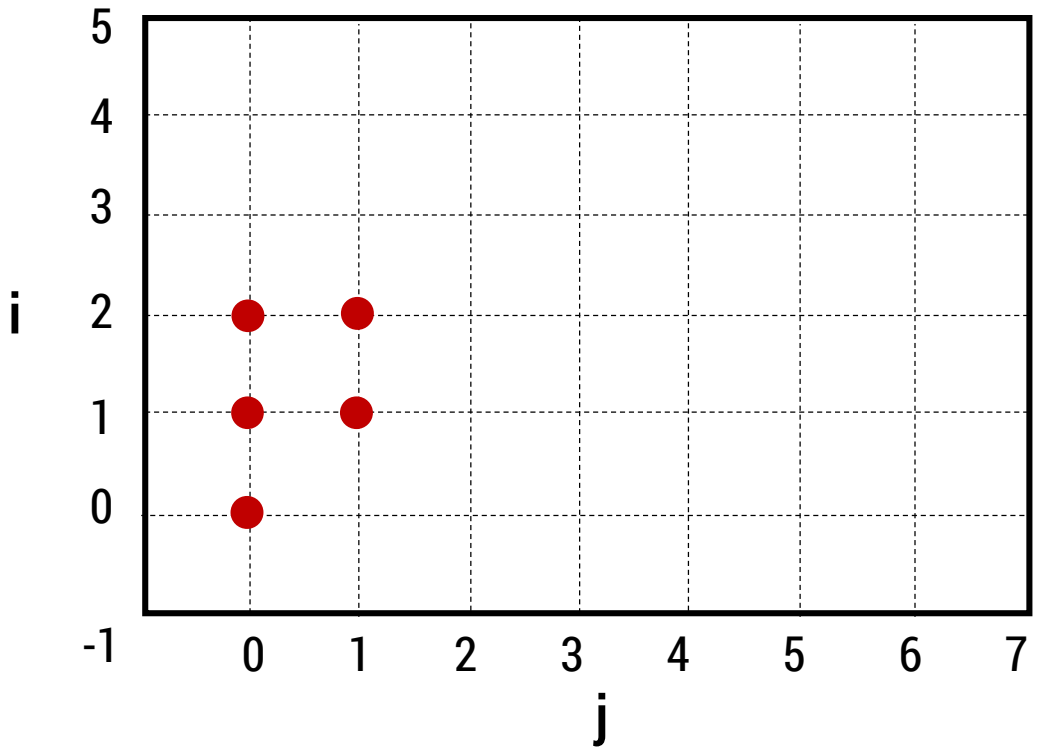
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

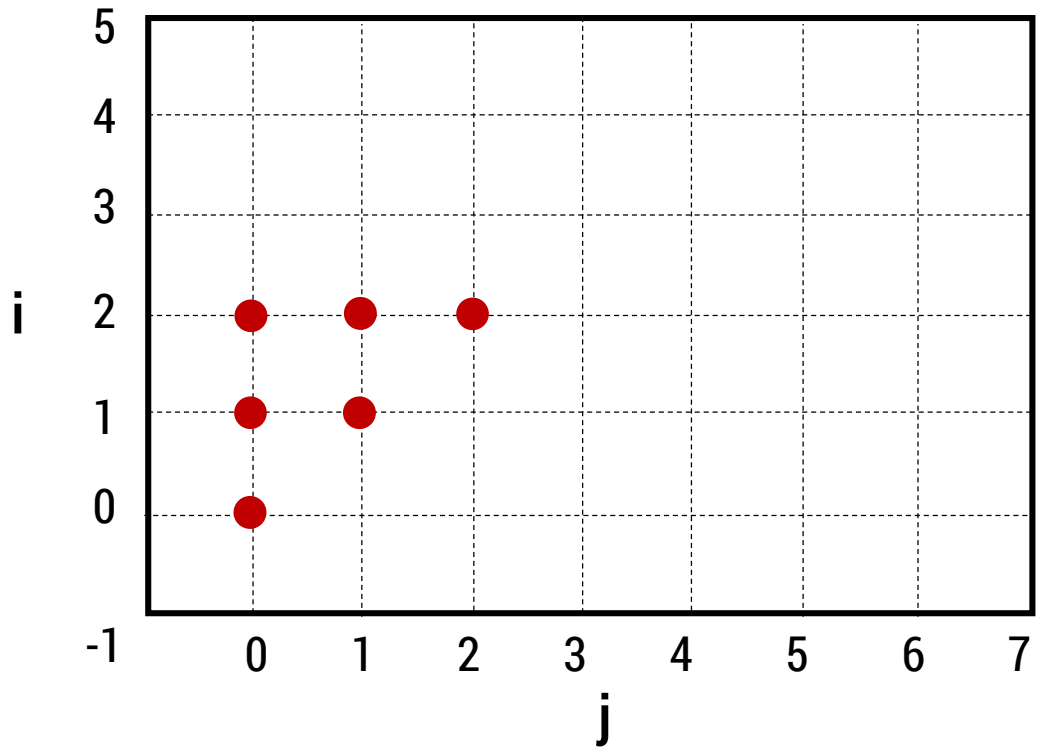
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

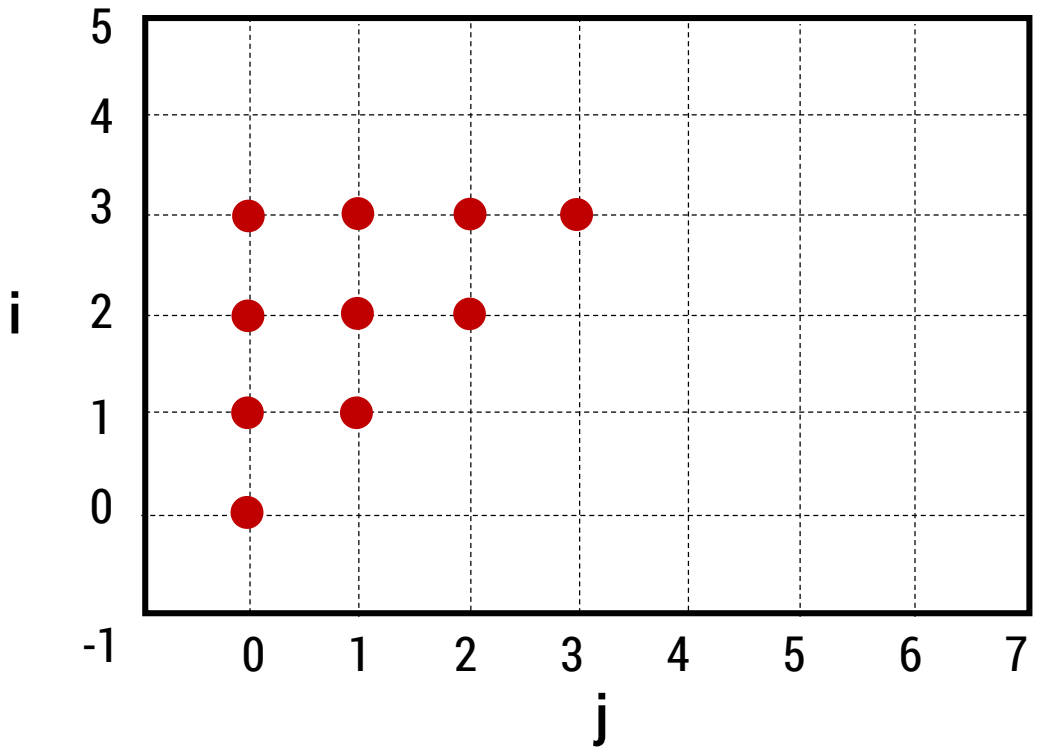
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

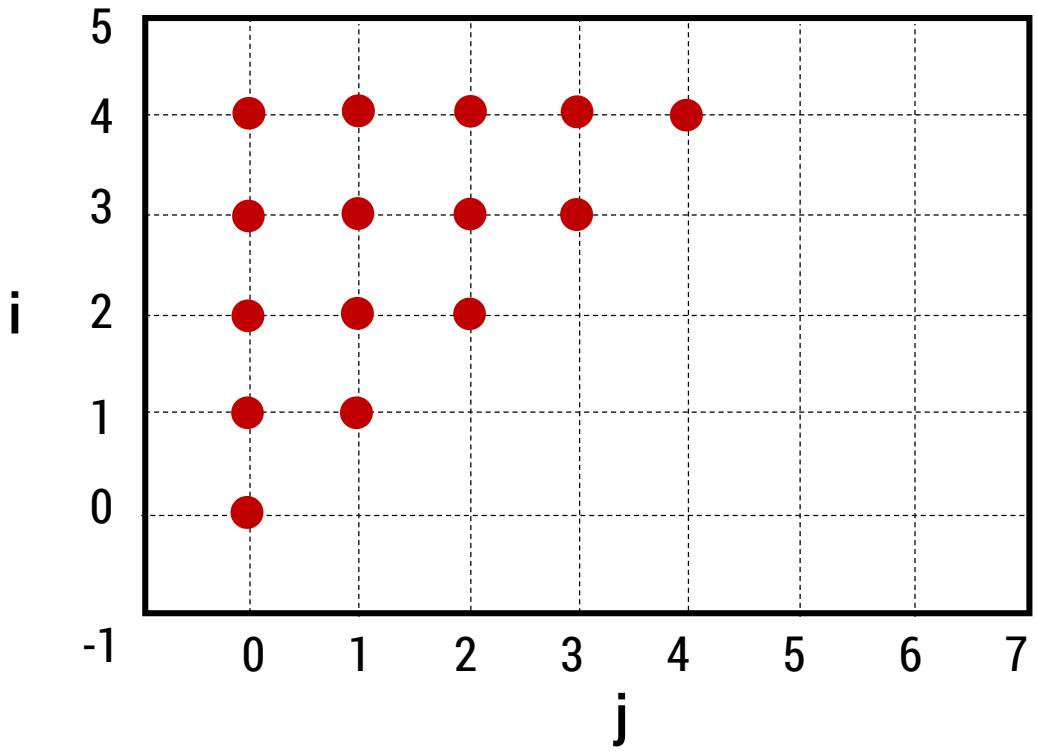
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

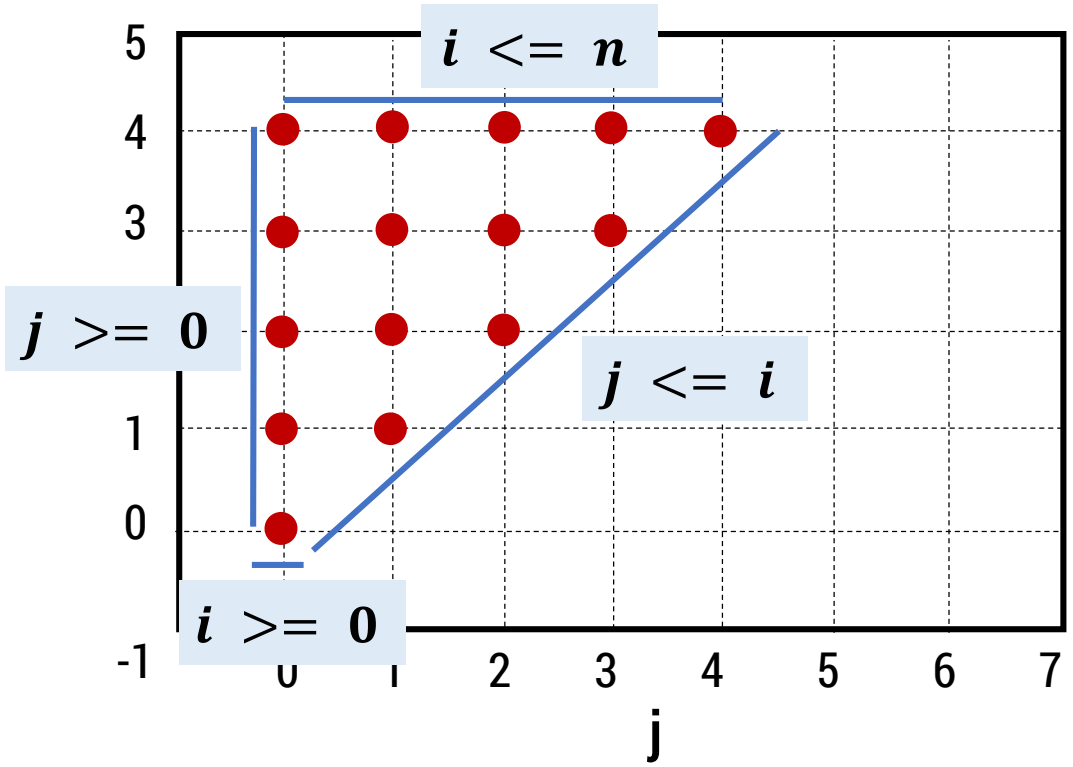
Полиэдральная модель

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

Полиэдральная модель

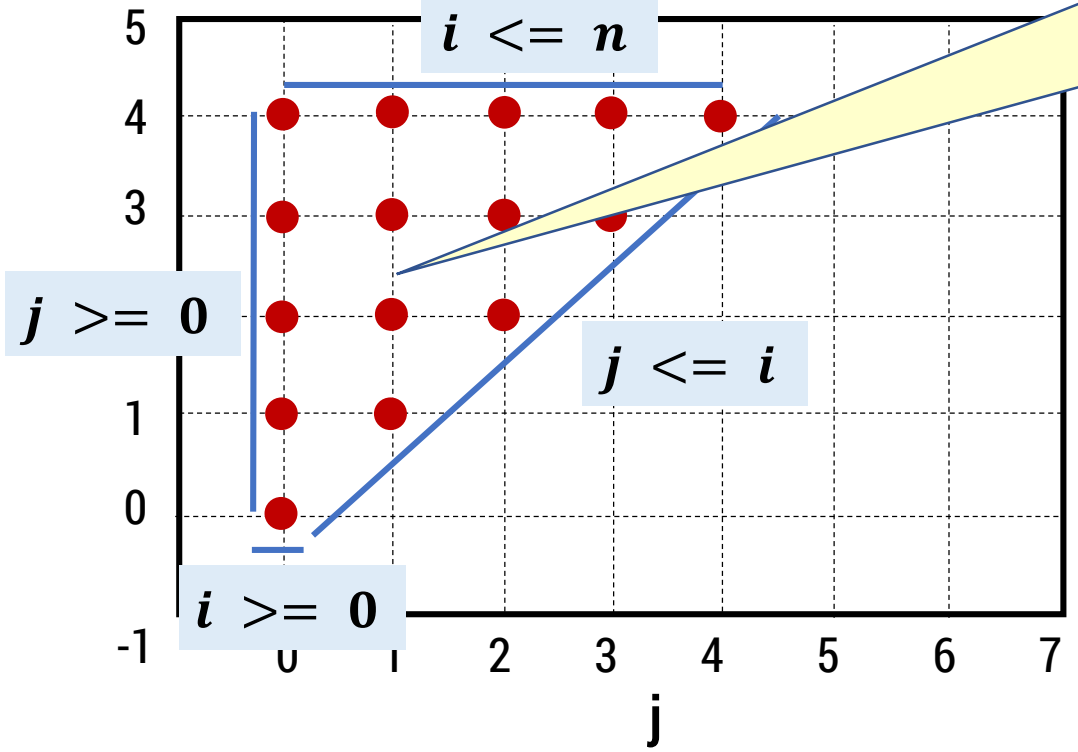
```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)

$$I_s = \{(i,j) \in \mathbb{Z}^2 \mid 0 \le i \le n \wedge 0 \le j \le i\}$$



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

Полиэдральная модель

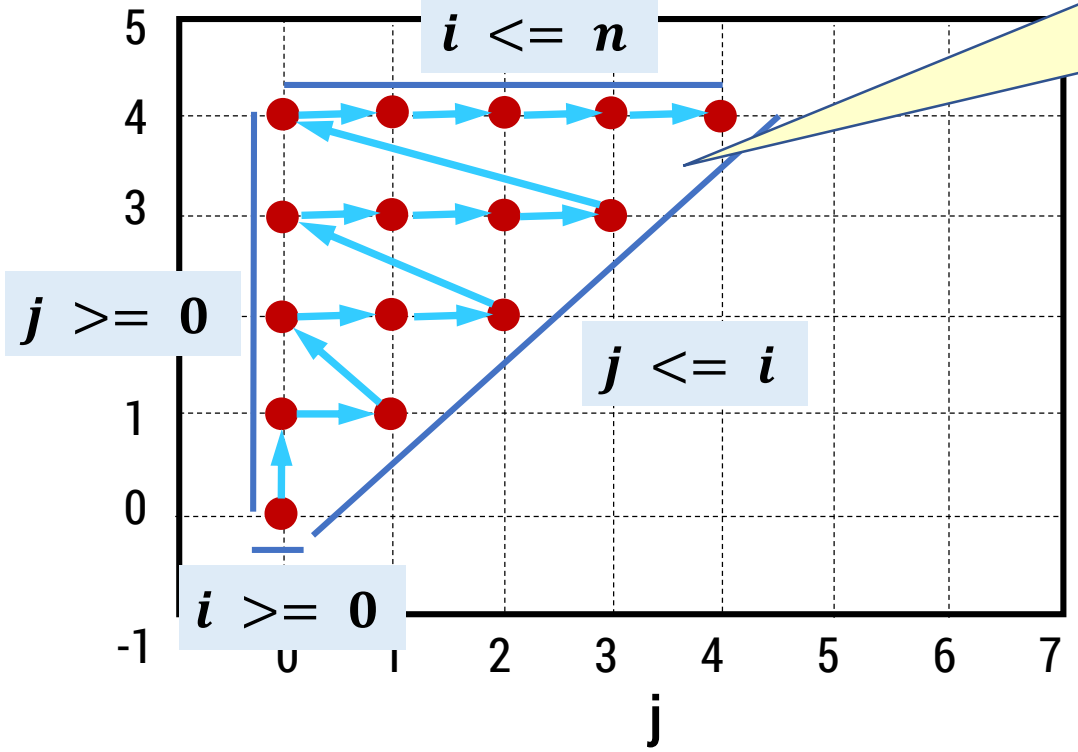
```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)

$$I_s = \{(i,j) \in \mathbb{Z}^2 \mid 0 \le i \le n \wedge 0 \le j \le i\}$$



Исполняемые экземпляры выражения
(Statement Instances Executed)

```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
S(3,0), S(3,1), S(3,2), S(3,3)
S(4,0), S(4,1), S(4,2), S(4,3), S(4,4)
```

$$\Theta_s = \{S(i,j) \rightarrow (i,j)\}$$

Порядок выполнения экземпляров выражения (Schedule)

Полиэдральная модель

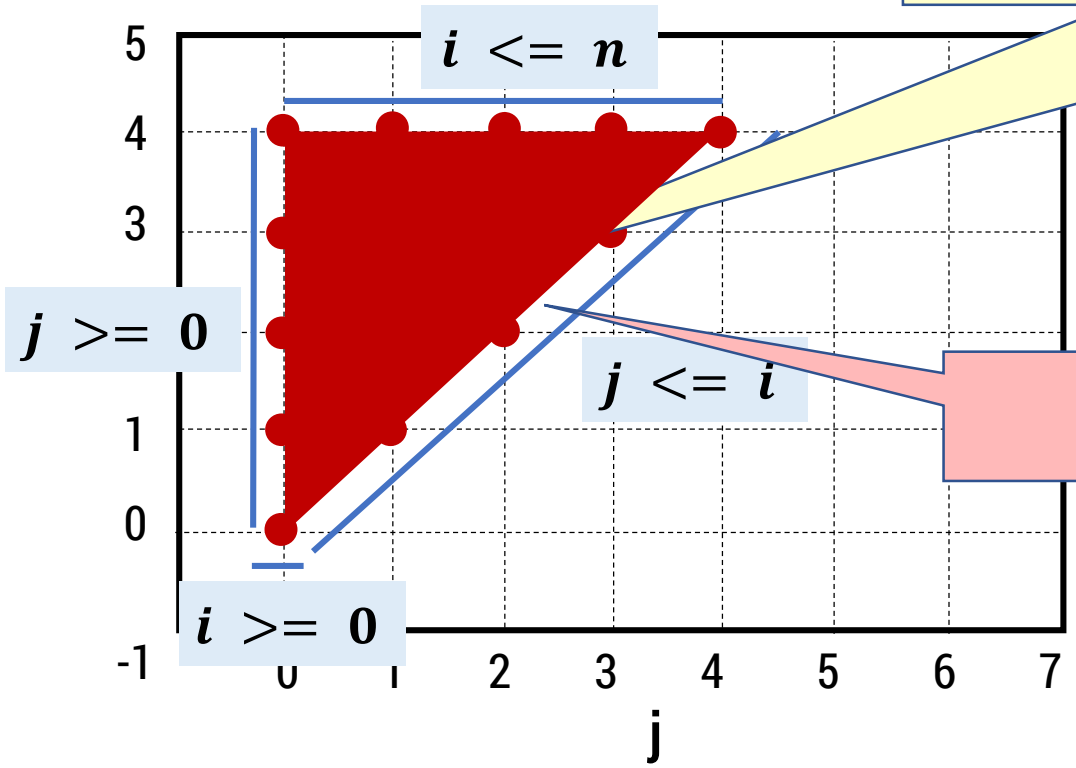
```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

```
n = 4

i = 4, j = 0...4
```

Область итераций
(Iteration Domain)

$$I_s = \{(i,j) \in \mathbb{Z}^2 \mid 0 \le i \le n \wedge 0 \le j \le i\}$$



Полиэдр

Исполняемые экземпляры выражения
(Statement Instances Executed)

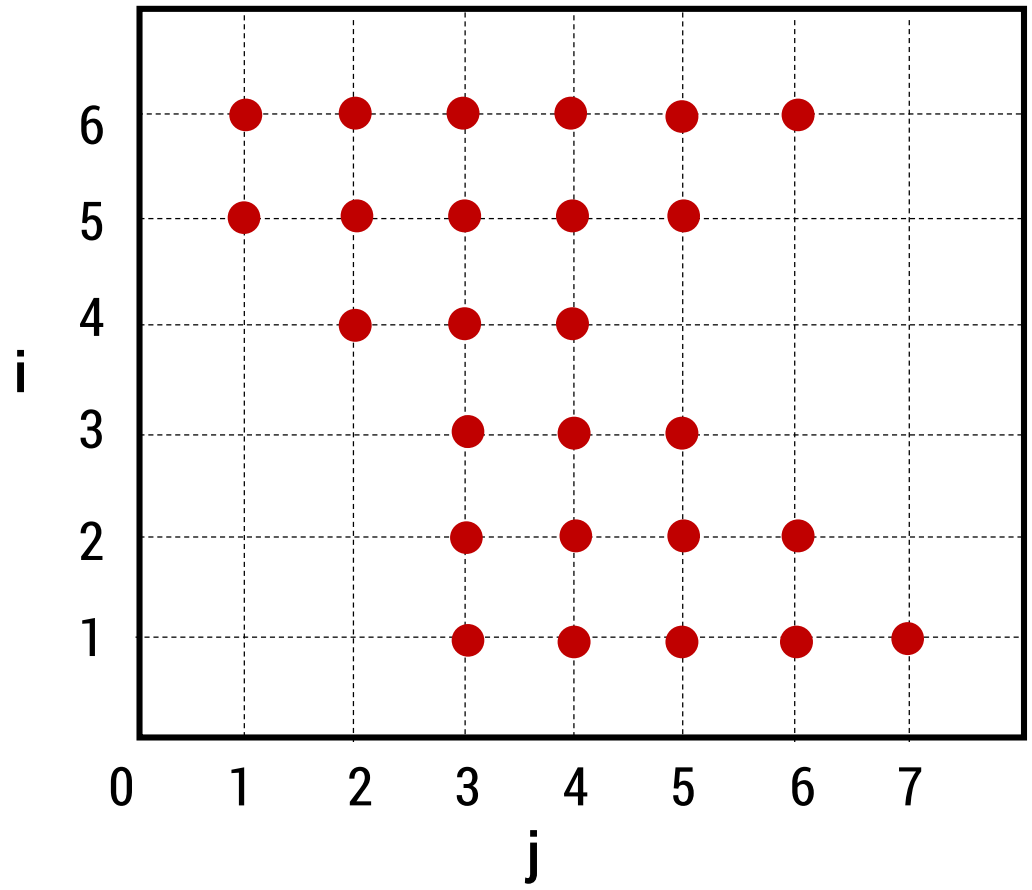
```
S(0,0)
S(1,0), S(1,1)
S(2,0), S(2,1), S(2,2)
        S(3,1), S(3,2), S(3,3)
        S(4,1), S(4,2), S(4,3), S(4,4)
```

$$\Theta_s = \{S(i,j) \rightarrow (i,j)\}$$

Порядок выполнения экземпляров выражения (Schedule)

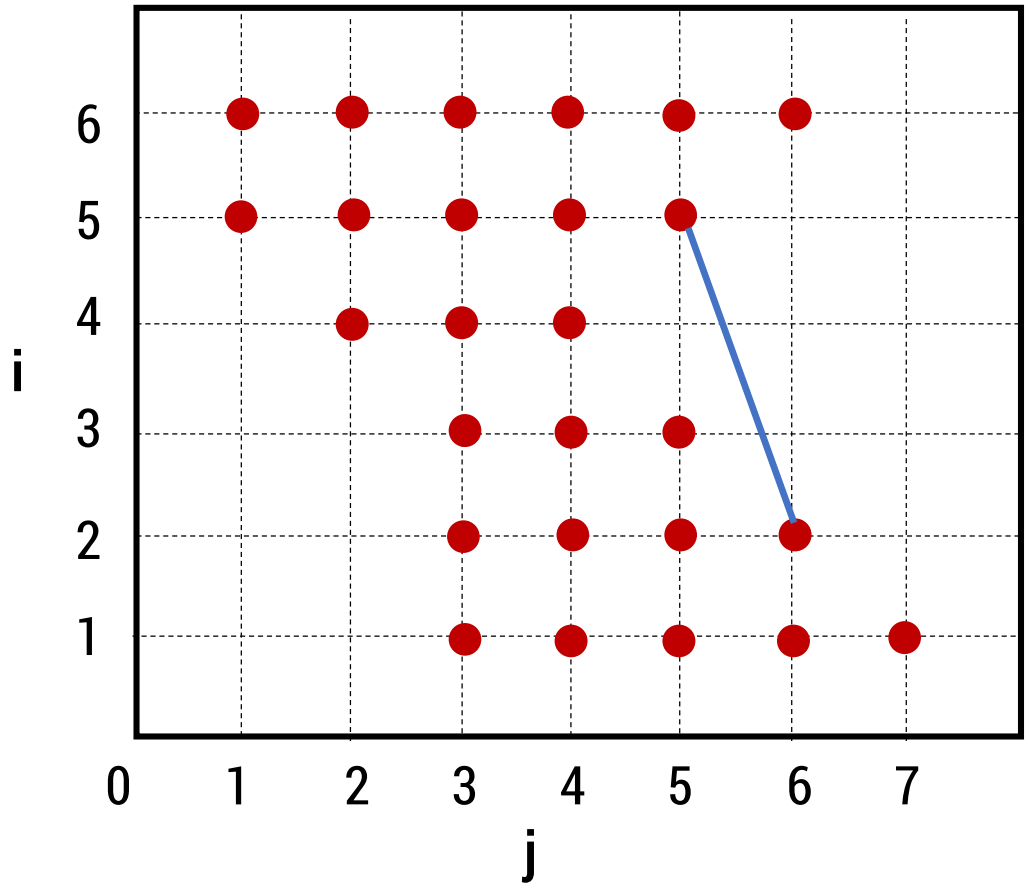
Полиэдральная модель

```
for(i = 1; i <= 6; i++)
   for(j = min(max(6-i,1),3); j <= max(8-i,2*i-5); j++)
      a[i][j] = a[i-1][j];
```



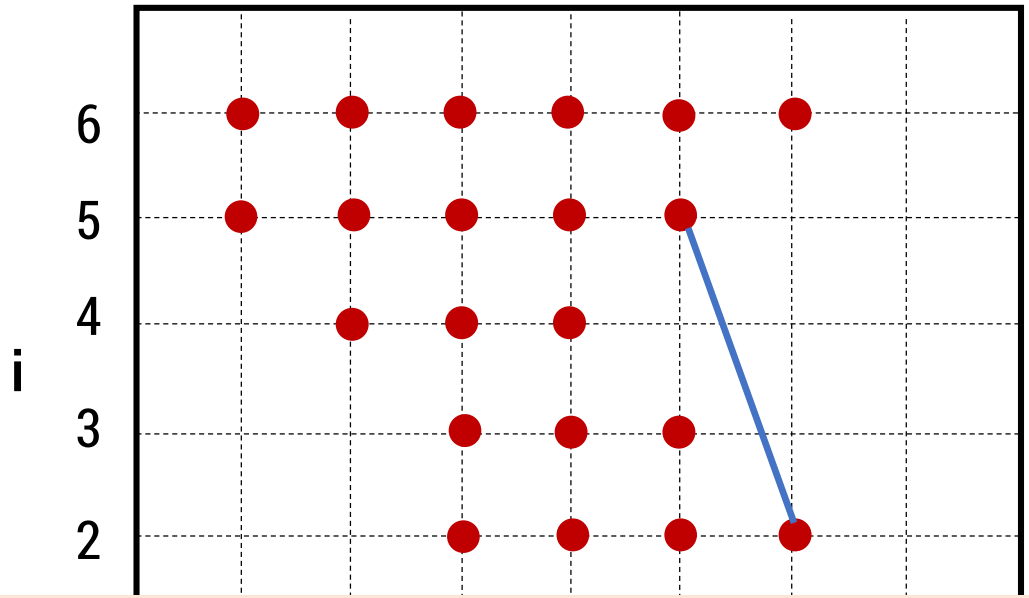
Полиэдральная модель

```
for(i = 1; i <= 6; i++)
   for(j = min(max(6-i,1),3); j <= max(8-i,2*i-5); j++)
      a[i][j] = a[i-1][j];
```



Полиэдральная модель

```
for(i = 1; i <= 6; i++)
   for(j = min(max(6-i,1),3); j <= max(8-i,2*i-5); j++)
      a[i][j] = a[i-1][j];
```



Полиэдр ($\mathbb{Z}$-полиэдр) – ВЫПУКЛОЕ множество точек, т.е. множество точек в векторном пространстве $\mathbb{Z}$, ограниченное аффинными неравенствами:

$I_s = \{\vec{\imath} \mid \vec{\imath} \in \mathbb{Z}^n, A\vec{\imath} + b \geq \vec{0}\}$ – область итераций (iteration domain)

Область итераций

Область итераций (Iteration Domain) – множество возможных значений векторов итераций для данного выражения

$$I_s = \{(i,j) \in \mathbb{Z}^2 \mid 0 \le i \le n \wedge 0 \le j \le i\}$$

$$\begin{cases} i \ge 0 \\ -i + n \ge 0 \\ j \ge 0 \\ i - j \ge 0 \end{cases} \Rightarrow \begin{cases} 1i + 0j + 0 \ge 0 \\ -1i + 0j + n \ge 0 \\ 0i + 1j + 0 \ge 0 \\ 1i - 1j + 0 \ge 0 \end{cases} \Rightarrow \underset{A}{\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix}} \underset{\vec{\imath}}{\begin{pmatrix} i \\ j \end{pmatrix}} + \underset{b}{\begin{pmatrix} 0 \\ n \\ 0 \\ 0 \end{pmatrix}} \ge \vec{0} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & n \\ 0 & 1 & 0 \\ 1 & -1 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \\ 1 \end{pmatrix} \ge \vec{0}$$

$$I_s = \left\{ (i,j) \in \mathbb{Z}^2, \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 0 \\ n \\ 0 \\ 0 \end{pmatrix} \ge \vec{0} \right\}$$

Полиэдр ($\mathbb{Z}$-полиэдр) – выпуклое множество точек, т.е. множество точек в векторном пространстве $\mathbb{Z}$, ограниченное аффинными неравенствами:

$I_s = \{\vec{\imath} \mid \vec{\imath} \in \mathbb{Z}^n, A\vec{\imath} + b \ge \vec{0}\}$ – область итераций (iteration domain)

Pouchet L.-N. *Iterative Optimization in the Polyhedral Model*.
URL: http://web.cs.ucla.edu/~pouchet/doc/pouchet-phdthesis.pdf

Порядок выполнения экземпляров выражений (Schedule)

Порядок выполнения выражений (Schedule) – функция, связывающая логическую временную метку с каждым выполнением данного выражения

$$\Theta_s = \{S(i,j) \to (i,j)\}$$

$$\Theta_s(\vec{\imath}) = T_s \begin{pmatrix} \vec{\imath} \\ \vec{n} \\ 1 \end{pmatrix}, T_s \in \mathbb{Z}^{p \times (\dim(\vec{\imath}) + \dim(\vec{n}) + 1)}$$

p – размерность расписания

$\vec{\imath}$ – вектор итераций

T_s – матрица преобразований

$\vec{n}$ – вектор глобальных параметров (переменные, значения которых не известны на этапе компиляции)

Порядок выполнения экземпляров выражений (Schedule)

Порядок выполнения выражений (Schedule) – функция, связывающая логическую временную метку с каждым выполнением данного выражения

Одномерные расписания описывают программу как один последовательный цикл, возможно включающий в себя один или несколько параллельных циклов

T_s - вектор

Многомерные расписания описывают программу как один или несколько вложенных последовательных циклов, возможно включающих в себя один или несколько параллельных циклов

T_s - матрица

p – размерность расписания

$\vec{\imath}$ – вектор итераций

T_s – матрица преобразований

$\vec{n}$ – вектор глобальных параметров (переменные, значения которых не известны на этапе компиляции)

Порядок выполнения экземпляров выражений (Schedule)

Порядок выполнения выражений (Schedule) – функция, связывающая логическую временную метку с каждым выполнением данного выражения

$$\Theta_s = \{S(i,j) \to (i,j)\}$$

$$\Theta_s \begin{pmatrix} i \\ j \end{pmatrix} = \overset{T_s}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}} \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix}$$

Порядок выполнения экземпляров выражений (Schedule)

Порядок выполнения выражений (Schedule) – функция, связывающая логическую временную метку с каждым выполнением данного выражения

$$\Theta_s = \{S(i,j) \to (i,j)\}$$

$$\Theta_s \begin{pmatrix} i \\ j \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ & \vec{\imath} & \vec{n} & \vec{c} \end{bmatrix} \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix}$$

Преобразования

$\vec{\imath}$	reversal	изменение порядка прохода по итерациям цикла на противоположный
	skewing	границы внутреннего цикла выражаются через счетчик внешнего цикла
	interchange	обмен циклов местами
$\vec{n}$	fusion	слияние двух циклов
	distribution	разделение одного гнезда циклов на несколько
$\vec{c}$	peeling	вынос итерации из цикла
	shifting	переупорядочивание циклов

Pouchet L.-N. *Polyhedral Compilation Foundations. Class #3*.
URL: http://www.cs.colostate.edu/~pouchet/lectures/doc/888.11.3.pdf

Порядок выполнения экземпляров выражений (schedule)

Порядок выполнения выражений (Schedule) – функция, связывающая логическую временную метку с каждым выполнением данного выражения

$$\Theta_s = \{S(i,j) \to (i,j)\}$$

$$\Theta_s \begin{pmatrix} i \\ j \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix}$$

$\vec{\imath}$ $\vec{n}$ $\vec{c}$

Преобразования

$\vec{\imath}$	reversal	изменение порядка прохода по итерациям цикла на противоположный
	skewing	границы внутреннего цикла выражаются через счетчик внешнего цикла
	interchange	обмен циклов местами
$\vec{n}$	fusion	слияние двух циклов
	distribution	разделение одного гнезда циклов на несколько
$\vec{c}$	peeling	вынос одной итерации из цикла
	shifting	переупорядочивание циклов

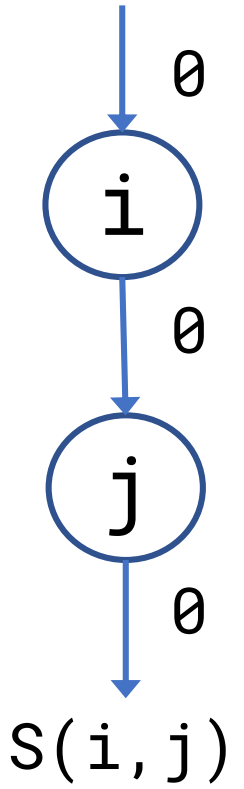
Преобразования, касающиеся только порядка выполнения выражений (пространство итераций и количество выражений не изменяются)

Порядок выполнения экземпляров выражений (Schedule)

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

$$\Theta_S(\vec{i}) = \{0, i, 0, j, 0\}^T$$

Абстрактное синтаксическое дерево
(abstract syntax tree, AST)



Bastoul C. *Code Generation in the Polyhedral Model Is Easier Than You Think*.
URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bas04-PACT.pdf

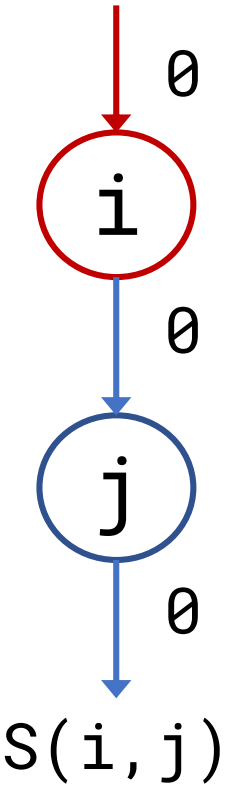
Порядок выполнения экземпляров выражений (Schedule)

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

$$\Theta_s(\vec{i}) = \{0, i, 0, j, 0\}^T$$

Цикл по `i` – самый внешний цикл

Абстрактное синтаксическое дерево
(abstract syntax tree, AST)



Bastoul C. *Code Generation in the Polyhedral Model Is Easier Than You Think*.
URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bas04-PACT.pdf

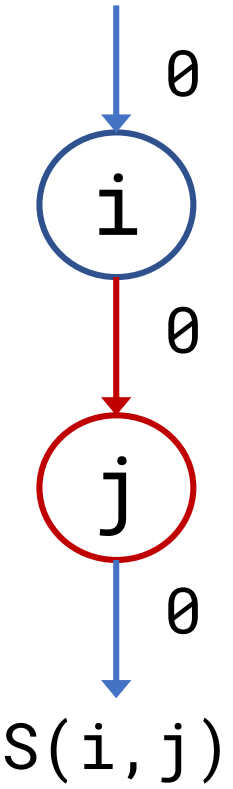
Порядок выполнения экземпляров выражений (Schedule)

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

$$\Theta_s(\vec{i}) = \{0, i, 0, j, 0\}^T$$

Цикл по `j` выполняется внутри цикла по `i` и имеет номер `0`

Абстрактное синтаксическое дерево (abstract syntax tree, AST)



Bastoul C. *Code Generation in the Polyhedral Model Is Easier Than You Think*.
URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bas04-PACT.pdf

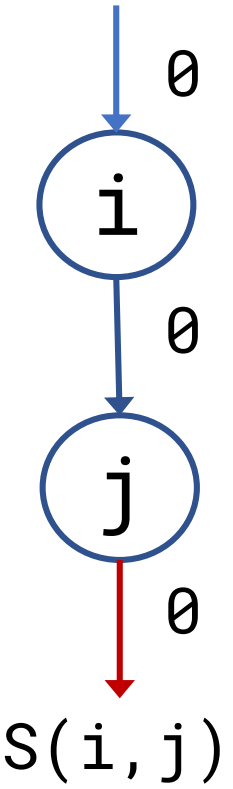
Порядок выполнения экземпляров выражений (Schedule)

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

$$\Theta_S(\vec{i}) = \{0, i, 0, j, 0\}^T$$

Выражение S имеет номер 0 внутри цикла по j

Абстрактное синтаксическое дерево
(abstract syntax tree, AST)



Bastoul C. *Code Generation in the Polyhedral Model Is Easier Than You Think*.
URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bas04-PACT.pdf

Порядок выполнения экземпляров выражений (Schedule)

```
for i=1,n
    S1
    for j=1,i-1
        S2
    S3
    for j=i+1,n
        S4
        for k=1,i-1
            S5
        S6
```

$$\Theta_{S1} = \{0, i, 0\}^T$$

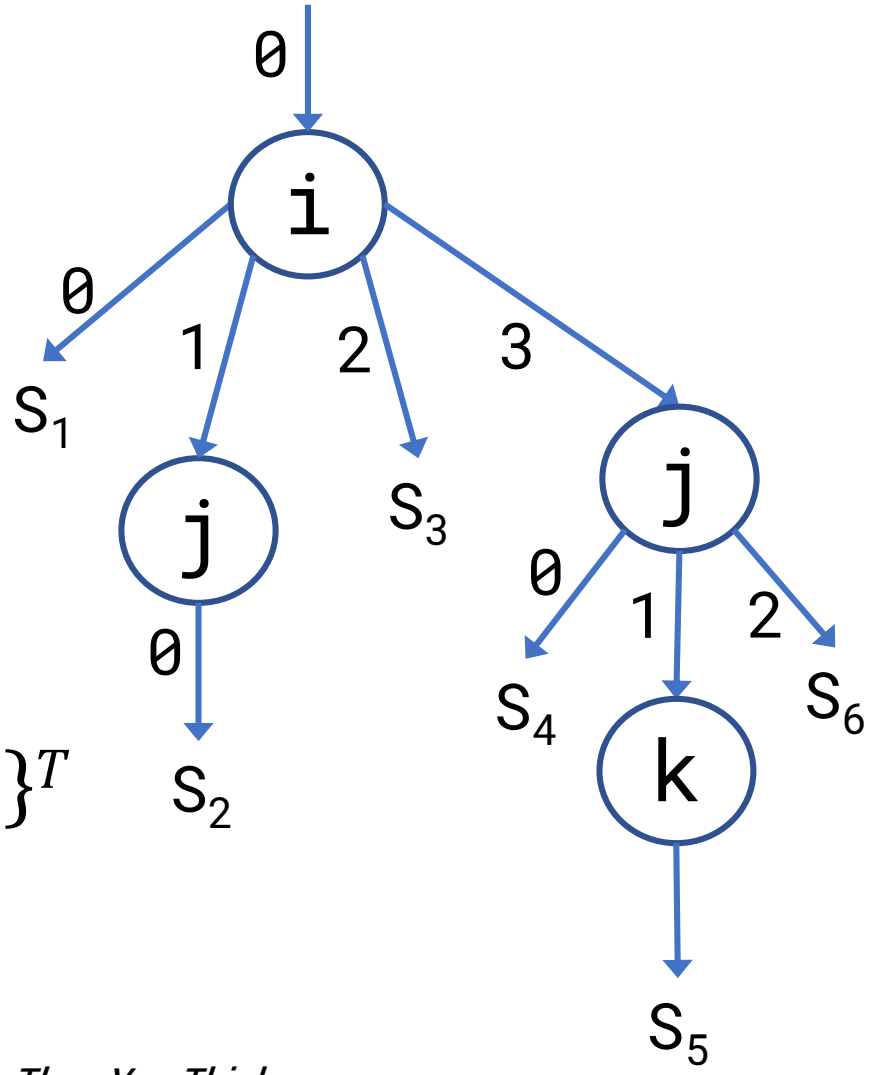
$$\Theta_{S2} = \{0, i, 1, j, 0\}^T$$

$$\Theta_{S3} = \{0, i, 2\}^T$$

$$\Theta_{S4} = \{0, i, 3, j, 0\}^T$$

$$\Theta_{S5} = \{0, i, 3, j, 1, k, 0\}^T$$

$$\Theta_{S6} = \{0, i, 3, j, 2\}^T$$



Bastoul C. *Code Generation in the Polyhedral Model Is Easier Than You Think.*
URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bas04-PACT.pdf

Функции обращения к данным (Access Functions)

- Каждое обращение к элементам массивов в выражениях представляется **отношением обращения к данным (access relation)**
- Это отношение отображает экземпляр выражения $\vec{i}$ на один или несколько элементов массива для выполнения операций чтения или записи
- Это отображение обычно выражается как множество аффинных выражений для итераторов циклов и глобальных параметров – **функций обращения к данным (access function)**

```
for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
        S(i,j);
```

$$\mathcal{A}_S(\vec{i}) = (S, i, j)$$

Shirako J., Hayashi A., Sarkar V. *Optimized two-level parallelization for GPU accelerators using the polyhedral model*. In Proceedings of the 26th International Conference on Compiler Construction (CC 2017). ACM, New York, NY, USA, 22-33.

Функции обращения к данным (Access Functions)

```
for(i = 0; i <= N; i++) {
   if(i <= N - 50)
S1:   A[5*i] = 1;
   else
S2:   A[3*i] = 2;
   for(j = 0; j <= N; j++)
S3:   B[i][2*j] = 3;
}
```

$$\mathcal{A}_{S1}(\vec{i}) = (A, 5i)$$
$$\mathcal{A}_{S2}(\vec{i}) = (A, 3i)$$
$$\mathcal{A}_{S3}(\vec{i}) = (B, i, 2j)$$

- Обращения к элементам массивов должны быть **аффинными функциями**

- Функция от одной или нескольких переменных $x_1, x_2, \ldots, x_n$ называется **аффинной**, если она может быть выражена как сумма константы и константных множителей, умноженных на переменные, т.е. как $c_0 + c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$, где $c_1, c_2, \ldots, c_n$ – константы

Функции обращения к данным (Access Functions)

```
for(i = 0; i <= N; i++) {
   if(i <= N - 50)
S1:   A[5*i] = 1;
   else
S2:   A[3*i] = 2;
   for(j = 0; j <= N; j++)
S3:   B[i][2*j] = 3;
}
```

$$\mathcal{A}_{S1}(\vec{i}) = (A, 5i)$$
$$\mathcal{A}_{S2}(\vec{i}) = (A, 3i)$$
$$\mathcal{A}_{S3}(\vec{i}) = (B, i, 2j)$$

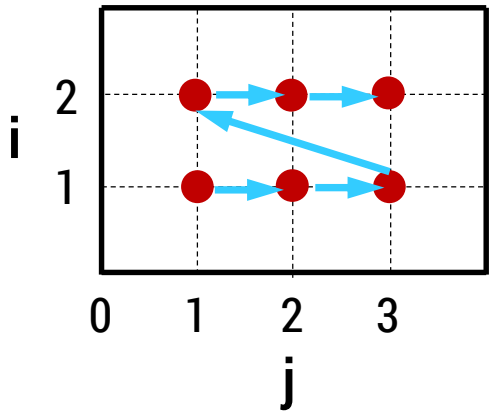
$$\mathcal{A}_{S3}(\vec{i}) = (B, i, 2j) = F(\vec{i}) + \vec{f} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- Обращения к элементам массивов должны быть **аффинными функциями**
- Функция от одной или нескольких переменных $x_1, x_2, \dots, x_n$ называется **аффинной**, если она может быть выражена как сумма константы и константных множителей, умноженных на переменные, т.е. как $c_0 + c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$, где $c_1, c_2, \dots, c_n$ – константы

Виды преобразований в полиэдральной модели

Преобразование в полиэдральной модели – это множество аффинных расписаний (schedules) для каждого выражения (statement) и(или) модификация полиэдрального представления

```
for (i = 1; i <= 2; i++)
    for (j = 1; j <= 3; j++)
```



$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

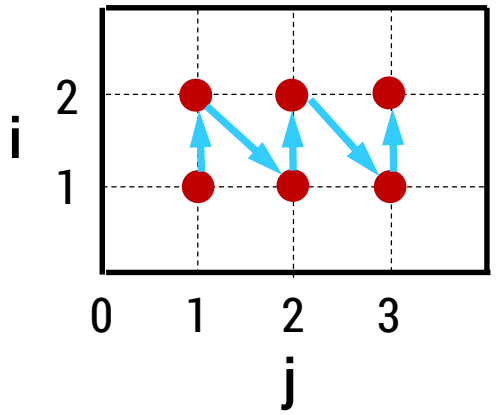
$$A\vec{\imath} + \vec{b} \geq \vec{0}$$

Обмен циклов (Interchange)

$$\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$$

$$\vec{\imath'} = T\vec{\imath}$$

```
for (j = 1; j <= 3; j++)
    for (i = 1; i <= 2; i++)
```



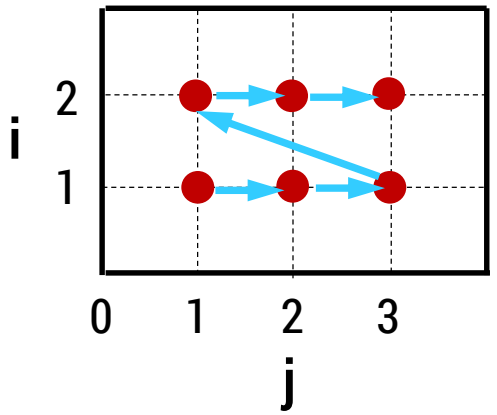
$$\begin{bmatrix} 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \end{bmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

$$(AT^{-1})\vec{\imath'} + \vec{b} \geq \vec{0}$$

Виды преобразований в полиэдральной модели

Преобразование в полиэдральной модели – это множество аффинных расписаний (schedules) для каждого выражения (statement) и(или) модификация полиэдрального представления

```
for (i = 1; i <= 2; i++)
    for (j = 1; j <= 3; j++)
```



$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

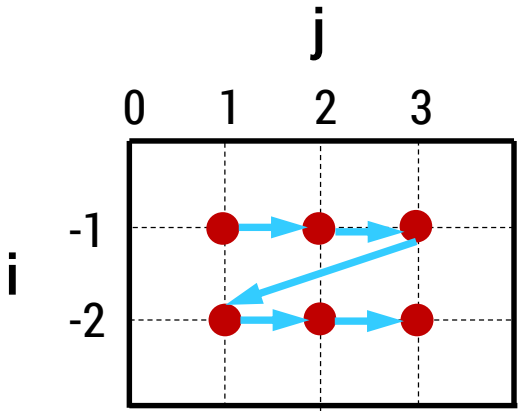
$$A\vec{\imath} + \vec{b} \geq \vec{0}$$

Изменение порядка прохода по итерациям цикла на противоположный (Reversal)

$$\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix}$$

$$\vec{\imath'} = T\vec{\imath}$$

```
for (i = -1; i >= -2; i--)
    for (j = 1; j <= 3; j++)
```



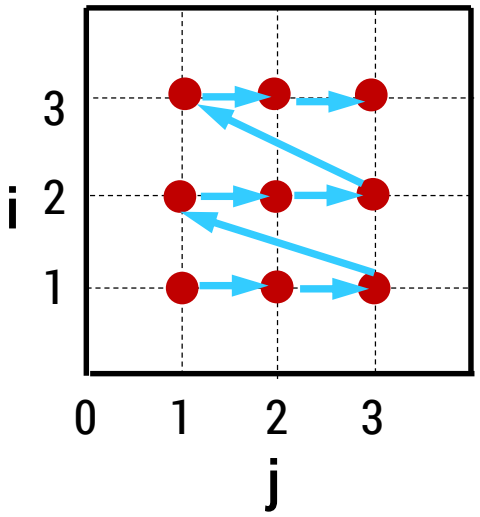
$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix} + \begin{pmatrix} 1 \\ -2 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

$$(AT^{-1})\vec{\imath'} + \vec{b} \geq \vec{0}$$

Виды преобразований в полиэдральной модели

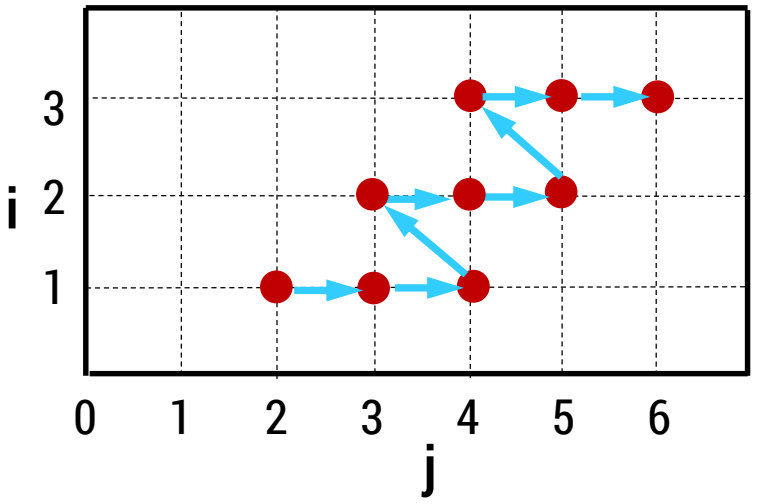
Преобразование в полиэдральной модели – это множество аффинных расписаний (schedules) для каждого выражения (statement) и(или) модификация полиэдрального представления

```
for (i = 1; i <= 3; i++)
    for (j = 1; j <= 3; j++)
```



Границы внутреннего цикла выражаются через счетчик внешнего цикла (Skewing)

```
for (i = 1; i <= 3; i++)
    for (j = i + 1; j <= i + 3; j++)
```



$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ 3 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0} \qquad \begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} \qquad \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix} + \begin{pmatrix} -1 \\ 3 \\ -1 \\ 3 \end{pmatrix} \geq \vec{0}$$

$$A\vec{i} + \vec{b} \geq \vec{0} \qquad \vec{i'} = T\vec{i} \qquad (AT^{-1})\vec{i'} + \vec{b} \geq \vec{0}$$

Зависимости по данным (dependencies)

Между двумя выражениями существует зависимость, если:

- они обращаются к одному и тому же элементу массива (к одной и той же ячейки памяти)
- одно из этих обращений – операция записи

Типы зависимостей:

- Потоковая (истинная) зависимость («чтение после записи», read-after-write, RAW)

```
    for(i = 0; i < N; i++){
S1:     A[i] = B[i] + C[i];
S2:     D[i] = A[i];
    }
```

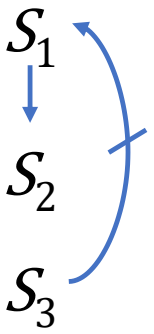
$S_1 \delta_= S_2$

- Антизависимость («запись после чтения», write-after-read, WAR)

```
    for(i = 0; i < N; i++){
S1:     A[i] = B[i] + C[i];
S2:     D[i] = A[i];
S3:     B[i] = D[i];
    }
```

$S_1 \delta_= S_2$

$S_3 \overline{\delta}_= S_1$

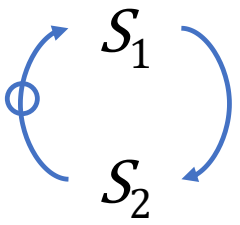


- Выходная зависимость («запись после записи», write-after-write, WAW)

```
    for(i = 0; i < N; i++){
S1:     A[i] = B[i] + C[i];
S2:     A[i+1] = A[i] + D[i];
    }
```

$S_1 \delta_= S_2$

$S_2 \delta^o_> S_1$



Зависимости по данным (dependencies)

Между двумя экземплярами выражений существует зависимость, если:

- они обращаются к одному и тому же элементу массива (к одной и той же ячейки памяти)
- одно из этих обращений – операция записи
- оба экземпляра действительно выполняются

Условия Бернстейна (Bernstein's Conditions):

- два экземпляра выражений a и b, обращающиеся к одной и той же ячейке памяти, могут выполняться в любом порядке, если ни одно из этих обращений не является обращением записи:

$$\begin{cases} W_a \cap W_b = \emptyset \\ R_a \cap W_b = \emptyset \\ W_a \cap R_b = \emptyset \end{cases}$$

R_x (W_x) – множества ячеек памяти, читаемых (записываемых) экземплярами выражений x

Преобразования порядка выполнения являются «законными», если они не меняют порядок выполнения зависимых пар экземпляров выражений

Bernstein A.J. *Program Analysis for Parallel Processing* // IEEE Trans. on Electronic Computers, EC-15, Oct. 1966, 757-762.

Зависимости по данным в полиэдральной модели

Выражение R зависит от выражения S ($S\delta R$), если существуют экземпляры выражений $S(\vec{\imath}_S)$ и $R(\vec{\imath}_R)$ и ячейка памяти m такая, что:

- $S(\vec{\imath}_S)$ и $R(\vec{\imath}_R)$ обращаются к одной и той же ячейке памяти m, и хотя бы один из них выполняет запись в эту ячейку

$$F_S(\vec{\imath}_S) + \vec{f}_S = F_R(\vec{\imath}_R) + \vec{f}_R$$

- $\vec{\imath}_S$ и $\vec{\imath}_R$ принадлежат пространству итераций R и S

$$A_S\vec{\imath}_S + \vec{b}_S \geq \vec{0}, A_R\vec{\imath}_R + \vec{b}_R \geq \vec{0}$$

- в исходном последовательном порядке выполнения $S(\vec{\imath}_S)$ выполняется перед $R(\vec{\imath}_R)$

$$P_S\vec{\imath}_S - P_R\vec{\imath}_R + \vec{p} \geq \vec{0}$$

Зависимости по данным в полиэдральной модели

- **Полиэдр зависимостей (dependence polyhedron)** для $R\delta S$ на данном уровне зависимости l и для данной пары обращений к памяти в выражениях R и S:

$$D\begin{pmatrix}\vec{i}_S\\ \vec{i}_R\end{pmatrix}+\vec{d}=\left[\begin{array}{cc}F_S & -F_R\\ \hline A_S & 0\\ 0 & A_R\\ P_S & -P_R\end{array}\right]\begin{pmatrix}\vec{i}_S\\ \vec{i}_R\end{pmatrix}+\begin{pmatrix}\vec{f}_R-\vec{f}_S\\ \vec{b}_S\\ \vec{b}_R\\ \vec{p}\end{pmatrix}\geqq\vec{0}$$

Зависимости по данным в полиэдральной модели

```
for (i = 1; i <= N; i++)
    for (j = 1; j <= N; j++)
S1:       A[i][j] = A[i+1][j+1];
```

- Область итераций

$$I_{S1}(\vec{i}) = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} -1 \\ N \\ -1 \\ N \end{pmatrix} \geq \vec{0}$$

- Функции обращения к данным

$$F_{S1}(\vec{i}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \geq \vec{0}$$

$$F_{S1}(\vec{i'}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \geq \vec{0}$$

Зависимости по данным в полиэдральной модели

```
for (i = 1; i <= N; i++)
    for (j = 1; j <= N; j++)
S1:       A[i][j] = A[i+1][j+1];
```

- Порядок предшествования $(i - i' = 1\,, j - j' = 1)$

$$P_{S1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix} + \begin{pmatrix} -1 \\ -1 \end{pmatrix} \geq \vec{0}$$

- Полиэдр зависимостей

$$\left[\begin{array}{cccc} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ \hline 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{array}\right] \begin{pmatrix} i \\ j \\ i' \\ j' \end{pmatrix} + \left(\begin{array}{c} 1 \\ 1 \\ \hline -1 \\ N \\ -1 \\ N \\ -1 \\ N \\ -1 \\ N \\ -1 \\ -1 \end{array}\right) \begin{matrix} = \\ \hline \geq \end{matrix} \vec{0}$$

Программные реализации

- CLooG: https://www.cloog.org/
- Polly: http://polly.llvm.org/index.html
- Graphite: https://gcc.gnu.org/wiki/Graphite
- LooPo: https://www.infosun.fim.uni-passau.de/cl/loopo/
- PLUTO: http://pluto-compiler.sourceforge.net/
- PPCG: http://ppcg.gforge.inria.fr/
- ISL: http://isl.gforge.inria.fr/

Литература

1. Ершов А.В. Линейные и аффинные пространства и отображения
   URL: https://mipt.ru/education/chair/mathematics/study/uchebniki/AffPreobr.pdf

2. Аффинные преобразования
   URL: https://ru.wikibooks.org/wiki/Аффинные_преобразования

3. Bastoul C., Cohen A., Girbal S., Sharma S., Temam O. Putting Polyhedral Loop Transformations to Work.
   URL: http://icps.u-strasbg.fr/~bastoul/research/papers/BCGST03-LCPC.pdf

4. Pouchet L.-N. Polyhedral Compilation Foundations. Lectures.
   URL: http://www.cs.colostate.edu/~pouchet/#lectures

5. Bastoul C. Improving Data Locality in Static Control Programs.
   URL: http://icps.u-strasbg.fr/~bastoul/research/papers/Bastoul_thesis.pdf

6. Pouchet L.-N. Iterative Optimization in the Polyhedral Model.
   URL: http://web.cs.ucla.edu/~pouchet/doc/pouchet-phdthesis.pdf

7. Grosser T., Doerfert J., Benaissa Z. Analyzing and Optimizing your Loops with Polly. URL:
   https://llvm.org/devmtg/2016-03/Tutorials/applied-polyhedral-compilation.pdf

Спасибо за внимание!

Молдованова Ольга Владимировна

ovm@sibguti.ru, ovm@isp.nsc.ru

Кафедра вычислительных систем, Сибирский государственный университет телекоммуникаций и информатики

Лаборатория вычислительных систем, Институт физики полупроводников им. А.В. Ржанова СО РАН

Семинар «Вычислительные системы»

Институт физики полупроводников им. А.В. Ржанова СО РАН

г. Новосибирск, 16 марта 2018 г.