POLYTOPE MODEL

Seminar "Sprachen für Parallelprogrammierung"

Tim Habermaas

POLYTOPE MODEL

- Ist ein mathematisches Modell zur automatischen Parallelisierung von Schleifen.

- Dabei wird das Quellprogramm als Polytop beschrieben und dieses Polytop so transformiert, dass das resultierende Zielprogramm parallelisierbar ist.

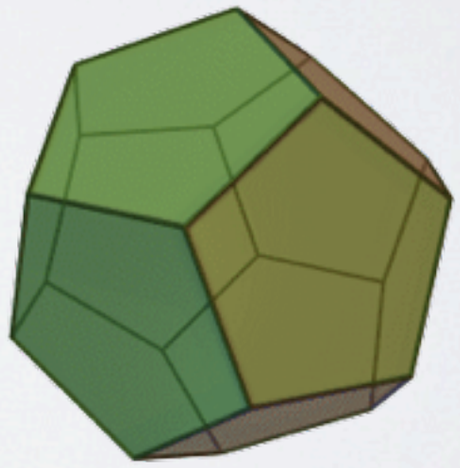
POLYTOP

- Verallgemeinertes Polygon in beliebiger Dimension

- Konvexe Polytope lassen sich als lineares Ungleichungssystem darstellen

$$A * x \geq b$$

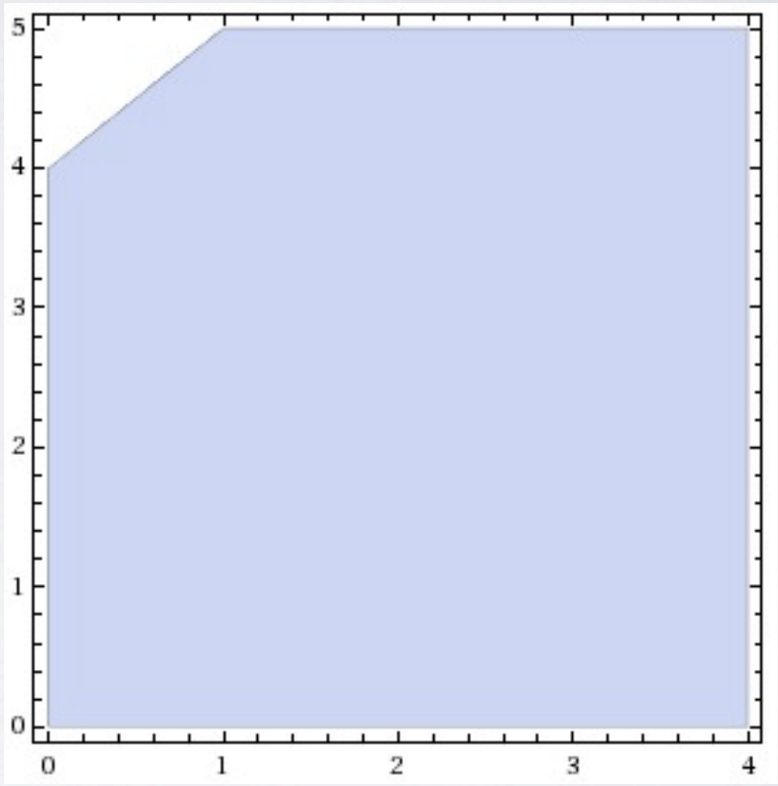
Polygon
n = 2



Polyeder
n = 3

POLYTOP-BEISPIEL

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & -1 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} \geq \begin{bmatrix} 0 \\ -4 \\ 0 \\ -5 \\ -4 \end{bmatrix}$$



QUELLPROGRAMM

```
for i1 = l1 to u1
  for i2 = l2 to u2
    ...
    a[i1, i2] = a[i1, 0] * 4 + b[i10 + 10 * i1]
  end
  S2
end
```

- Der einzige erlaubte Datentyp ist Array.
- Array-Indizes müssen affine Funktionen in den Schleifenvariablen sein.
- Die einzigen erlaubten Statements sind Schleifen (mit Schrittweite ±1) und Array-Zuweisungen.

INDEXRAUM

```
for i = 0 to n
  for j = 0 to m
    S1
  end
  S2
end
```

S_1: $i \geq 0$, $i \leq n$, $j \geq 0$, $j \leq m$

S_2: $i \geq 0$, $i \leq n$

- Jedes Statement **S** besitzt einen **Indexraum I_S**.
- Dieser Indexraum lässt sich als Ungleichungssystem darstellen, welches ein Polytop beschreibt.

INDEXVEKTOR

```
for i = 0 to 3
  for j = 0 to 4
    a[i, j] = 10
  end
end
```

$$\vec{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

- Ein Element eines Indexraumes heißt Indexvektor.

BEISPIEL

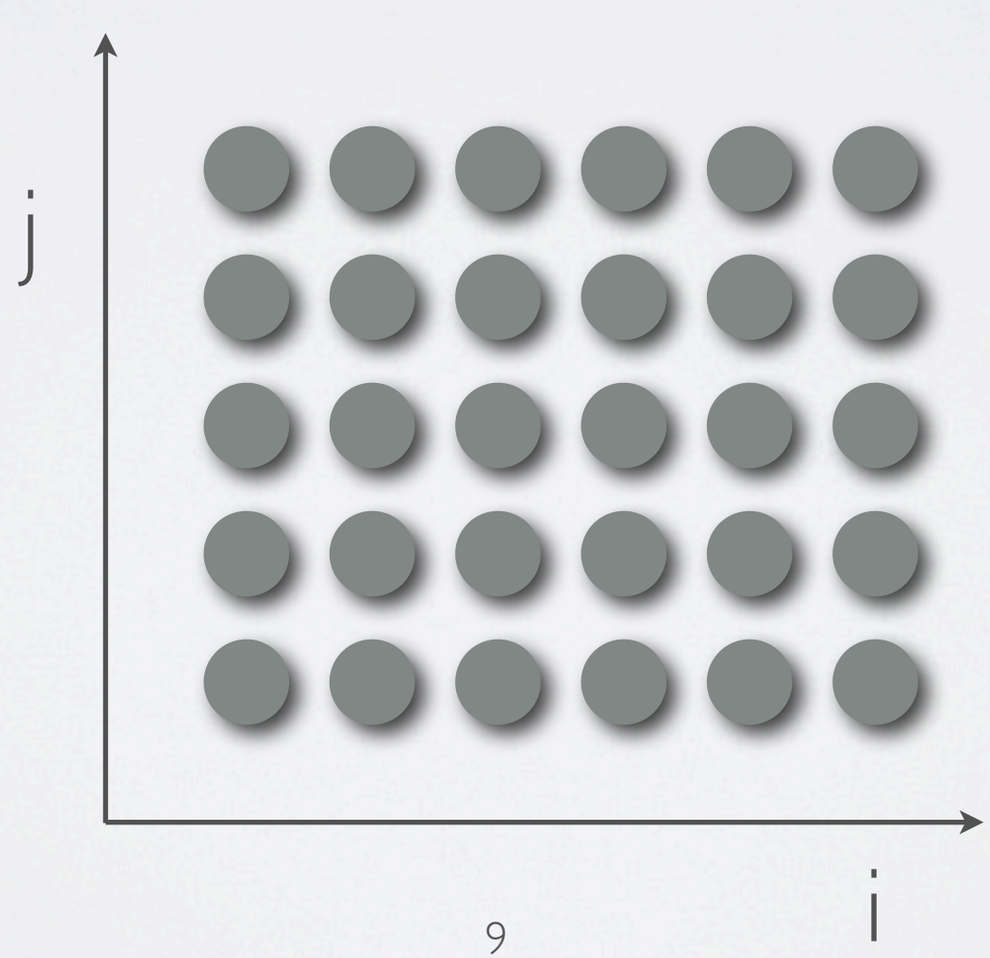
```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

Als Polytop:

$$A * \vec{x} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} i \\ j \end{bmatrix} \geq \begin{bmatrix} 0 \\ -5 \\ 0 \\ -4 \end{bmatrix}$$

BEISPIEL

```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

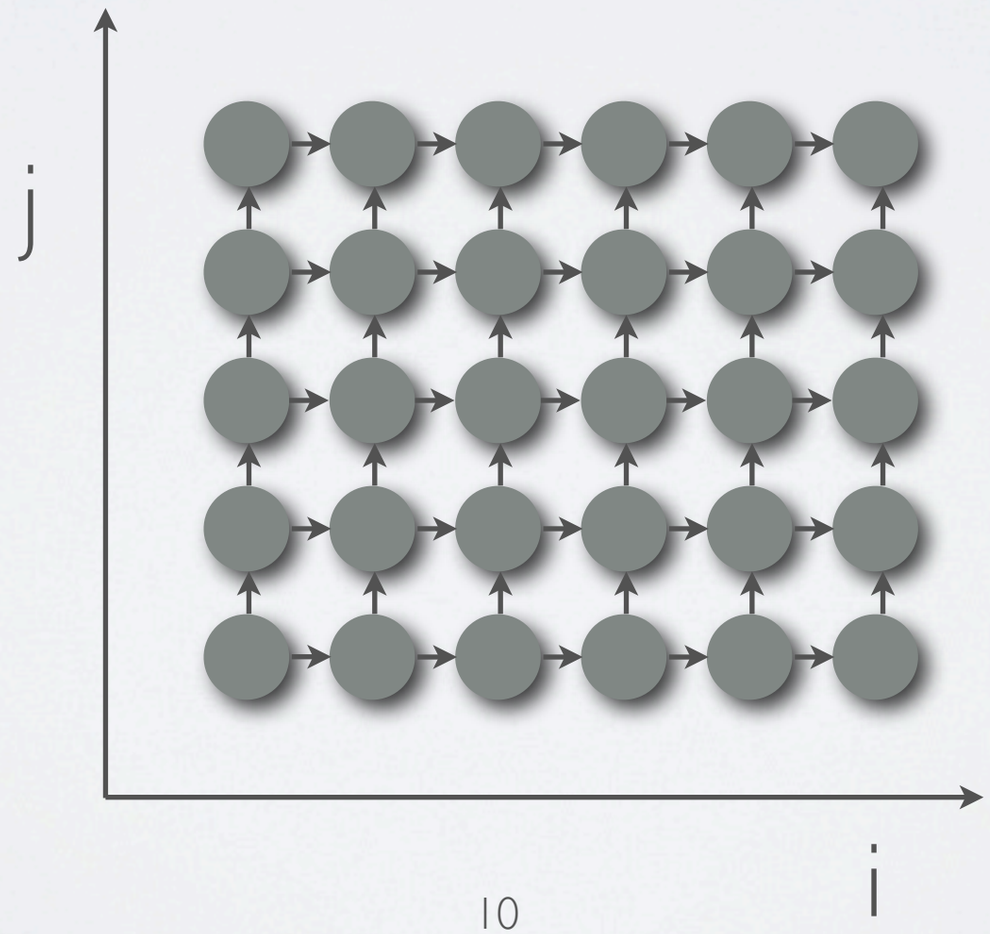


ABHÄNGIGKEITEN

```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

$$\vec{d_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\vec{d_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



SCHEDULE

$$t : Statement \times Indexvektor \to \mathbb{Z}$$

- Ist gültig, wenn er alle Abhängigkeiten erhält:

$$\forall x, x' : x, x' \in \Omega \wedge (x, x') \in E : t(x) <_{lex} t(x')$$

- Muss eine affine Funktion sein.

- Beschreibt welche Operationen zum Zeitpunkt **t** ausgeführt werden.

ALLOKATION

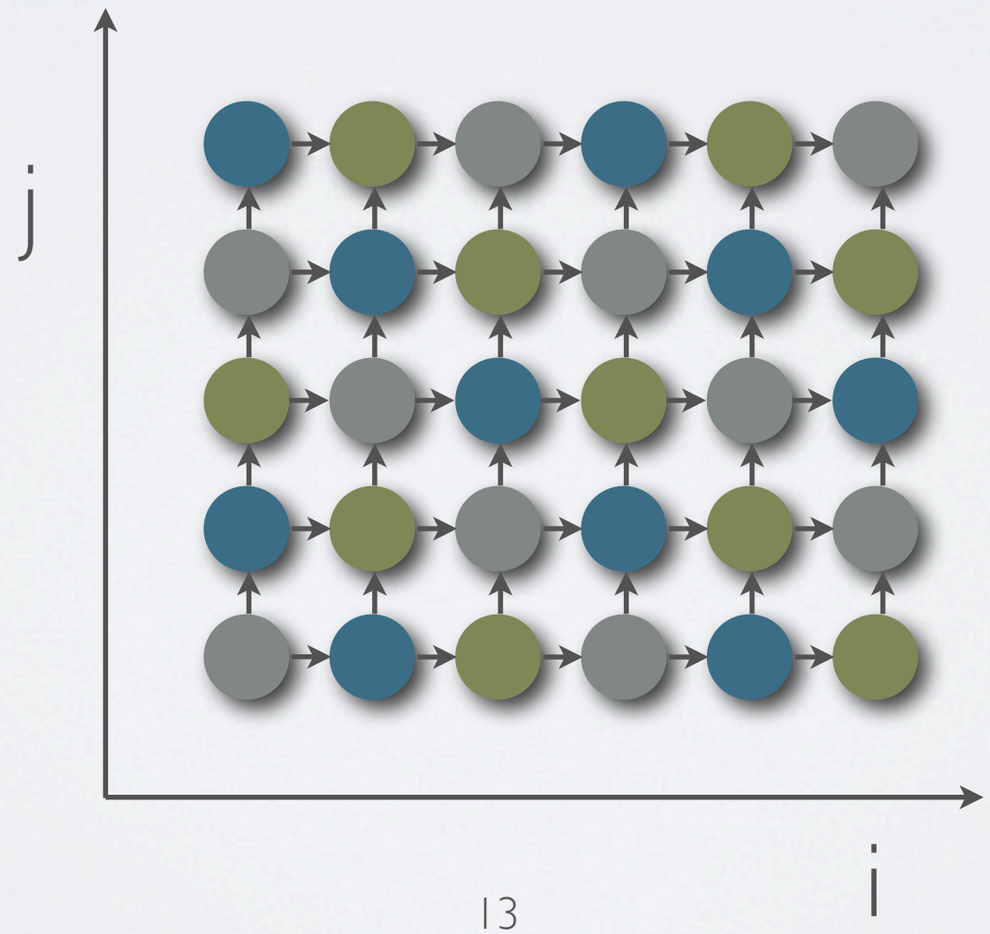
$$a : Statement \times Indexvektor \rightarrow \mathbb{Z}^r$$

- Weist einer Operation (Statement x Indexvektor) einen Prozessor zu.
- Muss ebenfalls eine affine Funktion sein.

SCHEDULE

```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

$$t(\begin{bmatrix} i \\ j \end{bmatrix}) = i + j$$

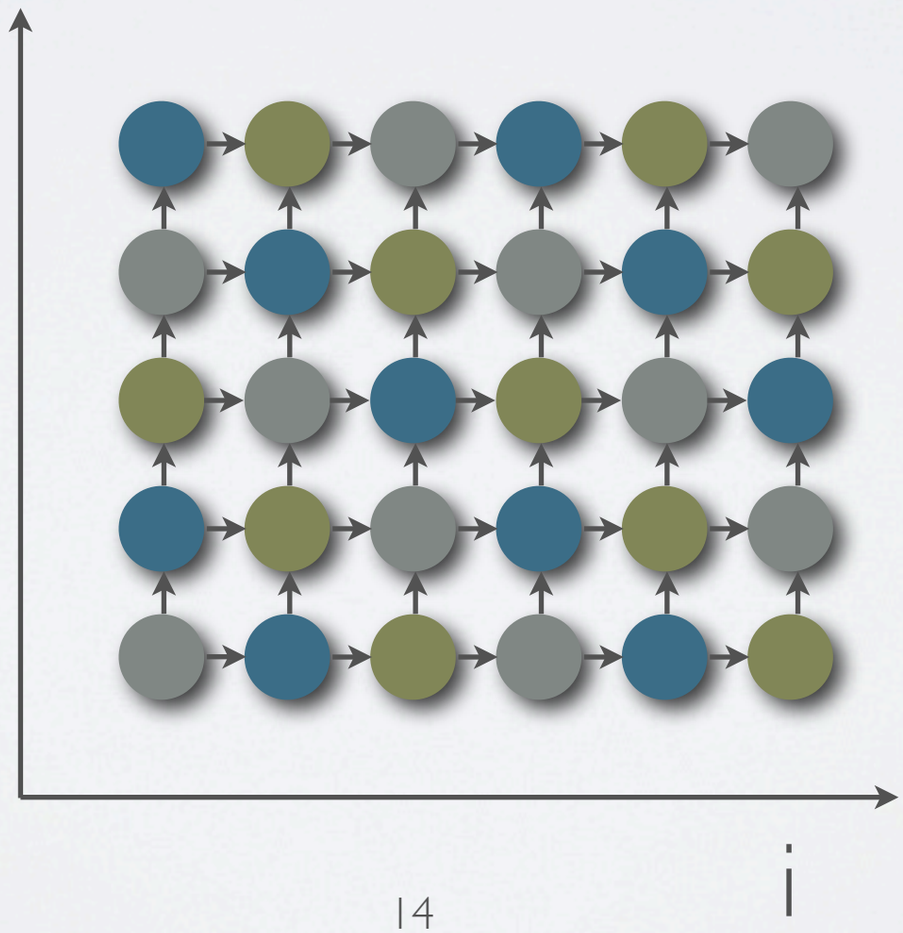


ALLOCATION

```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

$$t(\begin{bmatrix} i \\ j \end{bmatrix}) = i + j$$

$$a(\begin{bmatrix} i \\ j \end{bmatrix}) = i$$

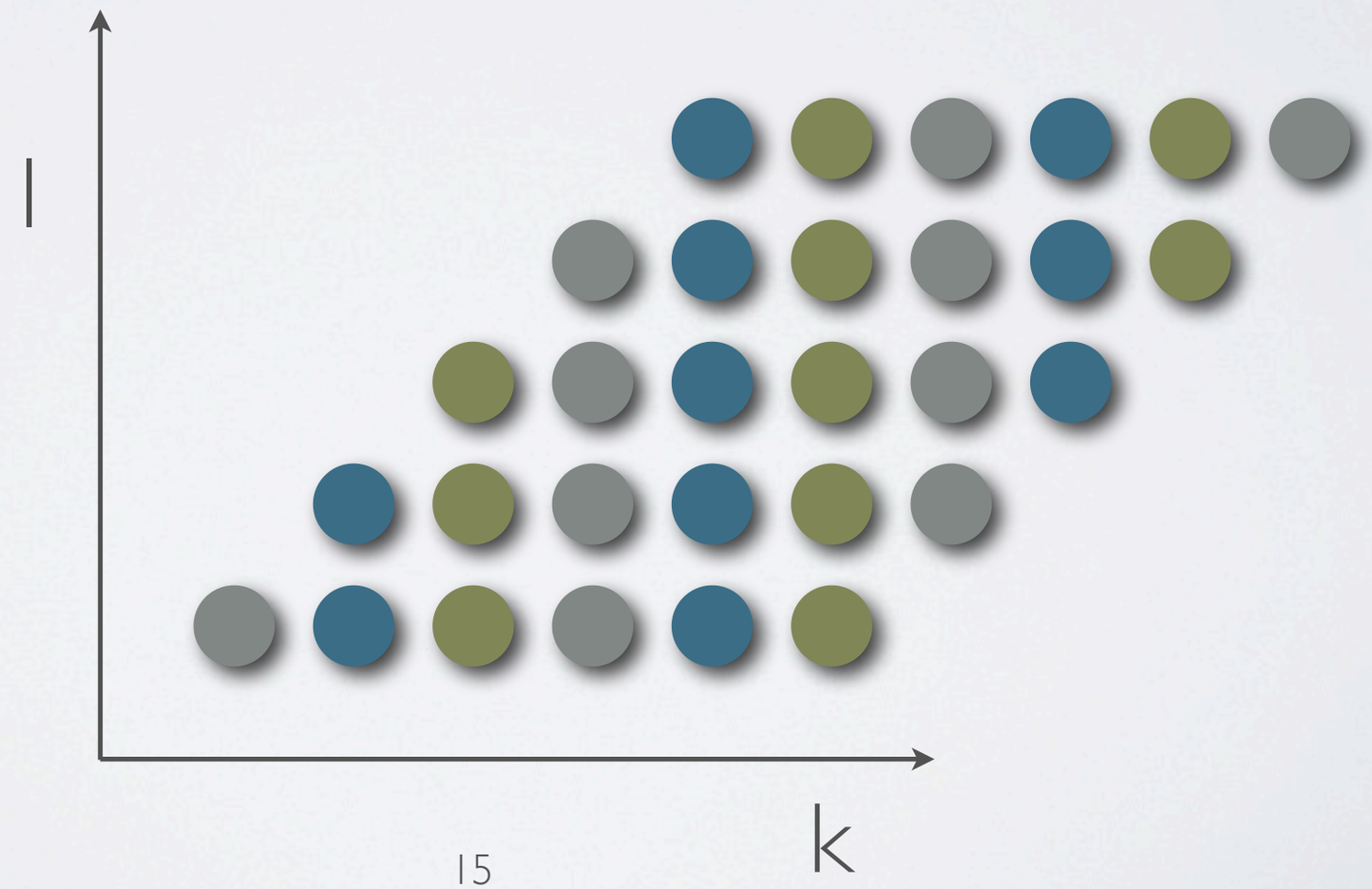


ZIEL

```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

$$t(\begin{bmatrix} i \\ j \end{bmatrix}) = i + j$$

$$a(\begin{bmatrix} i \\ j \end{bmatrix}) = i$$



TRANSFORMATIONSMATRIX

```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

$$t(\begin{bmatrix} i \\ j \end{bmatrix}) = i + j$$

$$a(\begin{bmatrix} i \\ j \end{bmatrix}) = i$$

$$T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

NEUE INDIZES

```
for i = 0 to 5
  for j = 0 to 4
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```

$$T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$T * \vec{x} = T * \begin{bmatrix} i \\ j \end{bmatrix} = \vec{y} = \begin{bmatrix} k \\ l \end{bmatrix} = \begin{bmatrix} i + j \\ i \end{bmatrix}$$

TRANSFORMIERTES POLYTOP

$$A * \vec{x} = A * (T^{-1} * \vec{y}) \geq \vec{b}$$

$$T^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} * (\begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} * \vec{y}) = \begin{bmatrix} l \\ -l \\ k-l \\ l-k \end{bmatrix} \geq \begin{bmatrix} 0 \\ -5 \\ 0 \\ -4 \end{bmatrix}$$

TRANSFORMIERTES POLYTOP

$$\begin{bmatrix} l \\ -l \\ k-l \\ l-k \end{bmatrix} \geq \begin{bmatrix} 0 \\ -5 \\ 0 \\ -4 \end{bmatrix}$$

Problem:

Das transformierte Polytope lässt sich nicht ohne weiteres in eine Schleife verwandeln, da **l** von **k** und **k** von **l** abhängt.

Lösung:

Fourier-Motzkin-Elimination

FOURIER-MOTZKIN-ELIMINATION

- Eliminiert Variablen aus einem Ungleichungssystem.

FOURIER-MOTZKIN-ELIMINATION

$$l \geq 0$$
$$l \geq k - 4$$
$$l \leq 5$$
$$l \leq k$$

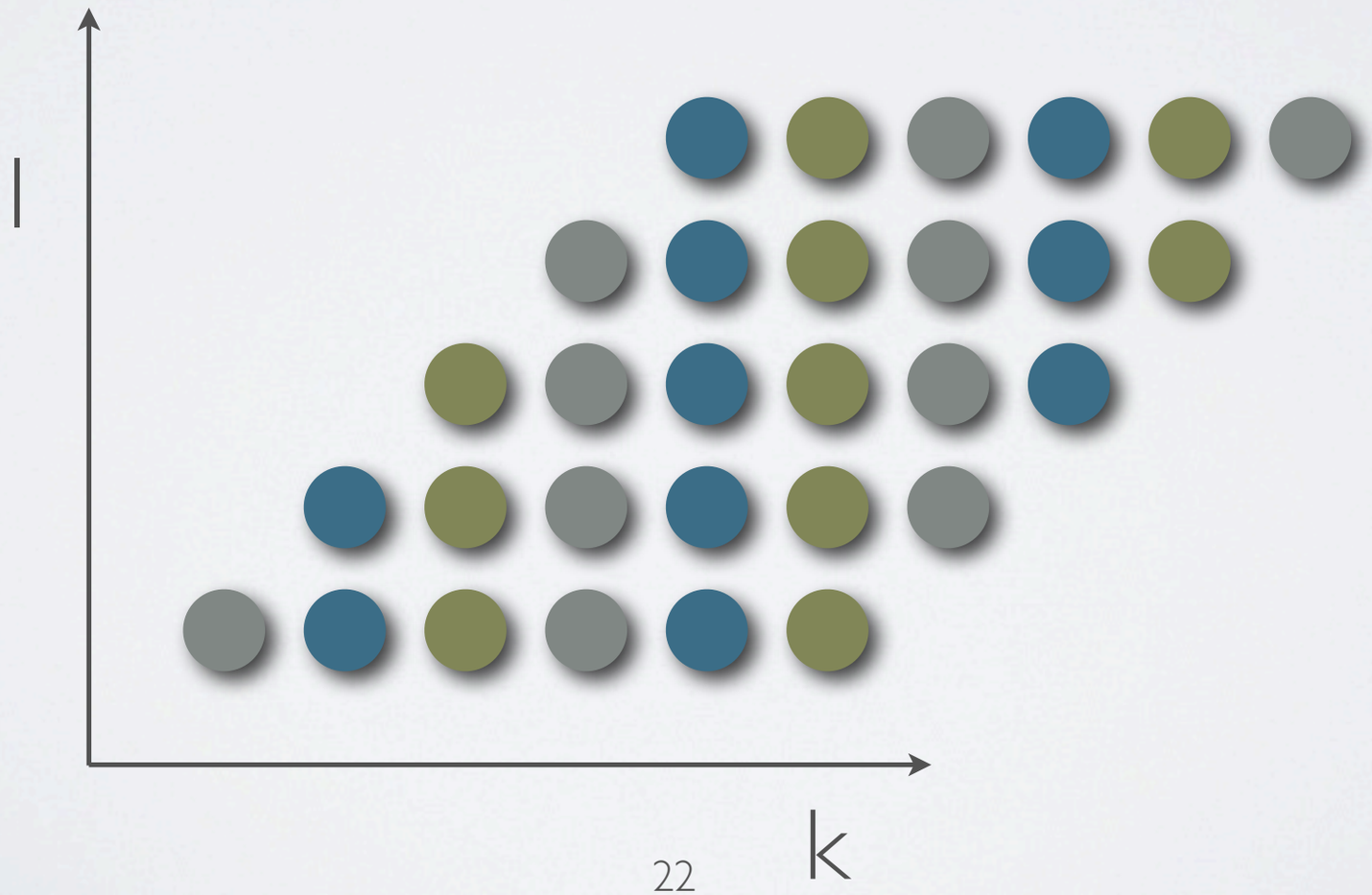
$$max(0, k-4) \leq l \leq min(k, 5) \Rightarrow$$

$$max(0, k-4) \leq min(k, 5)$$

$$\Rightarrow 0 \leq k$$
$$k \leq 9$$

ZIELPROGRAMM

```
for k = 0 to 9
  parfor l = max(0, k - 4) to min(5, k)
    i = l
    j = k - l
    a[i, j] = a[i - 1, j] + a[i, j - 1] * 10
  end
end
```



FINDEN EINES SCHEDULES

- Hyperebenen-Methode von Lampert
- Methode von Feautrier

HYPEREBENEN-METHODE

- benötigt perfekt verschachtelte Schleifen
- schleifenbasiert, dies heißt für jede Schleife wird eine Transformationsmatrix berechnet
- nicht immer optimal

```
for i1 = l1 to u1
  for i2 = l2 to u2
    S1
    S2
    ...
  end
end
```

METHODE VON FEAUTRIER

- statementbasiert, das heißt für jedes Statement gewinnt man eine eigene Transformationsmatrix
- optimal
- löst LIP (Linear Integer Program)
- hoher Rechenaufwand

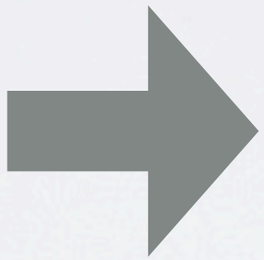
ZUSAMMENFÜGEN VON SCHLEIFEN

- Runtime
- Compile time

RUNTIME

```
for i = 0 to 8
  for j = 0 to m
    S1
  end
  S2
end
```

```
for i = 0 to 5
  S3
end
```

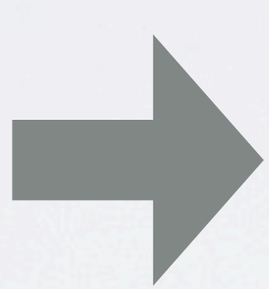


```
for i = 0 to 8
  for j = 0 to m
    S1
  end
  S2
  if i <= 5
    S3
  end
end
```

COMPILE TIME

```
for i = 0 to 8
  for j = 0 to m
    S1
  end
  S2
end
```

```
for i = 0 to n
  S3
end
```



```
for i = 0 to n
  for j = 0 to m
    S1
  end
  S2
  S3
end

for i = n + 1 to 8
  for j = 0 to m
    S1
  end
  S2
end
```

$0 <= n <= 8$

LITERATUR

- **Loop Parallelization in the Polytope Model** (Christian Lengauer)

- **Code Generation in the Polytope Model** (Martin Griebl, Christian Lengauer, Sabine Wetzel)

- **Automatische Methoden zur Parallelisierung im Polyedermodell** (Christian Wieninger)