Loop Transformations: Convexity, Pruning and Optimization

Louis-Noël Pouchet[1] Uday Bondhugula[2] Cédric Bastoul[3] Albert Cohen[3]
J. Ramanujam[4] P. Sadayappan[1] Nicolas Vasilache[5]

[1] **The Ohio State University**
[2] **IBM T.J. Watson Research Center**
[3] **ALCHEMY group, INRIA Saclay / University of Paris-Sud 11**
[4] **Louisiana State University**
[5] **Reservoir Labs, Inc.**

January 28, 2011
ACM 2011 Symposium on
Principles of Programming Languages
Austin, TX

THE OHIO STATE UNIVERSITY

Compiler Optimizations for Performance

- **High-level loop transformations are critical for performance...**
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)

Compiler Optimizations for Performance

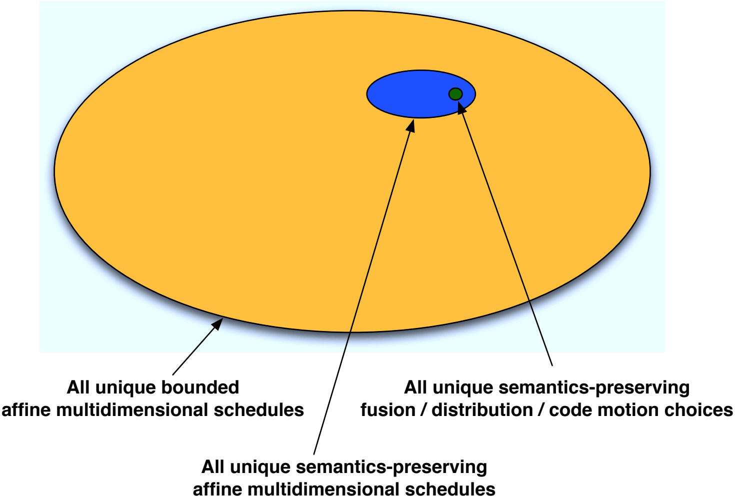
- **High-level loop transformations are critical for performance...**
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)
- **... But deciding the best sequence of transformations is hard!**
  - Conflicting objectives: more SIMD implies less locality, etc.
  - It is machine-dependent and of course program-dependent
  - Expressive search spaces are required, but challenge the search!

Compiler Optimizations for Performance

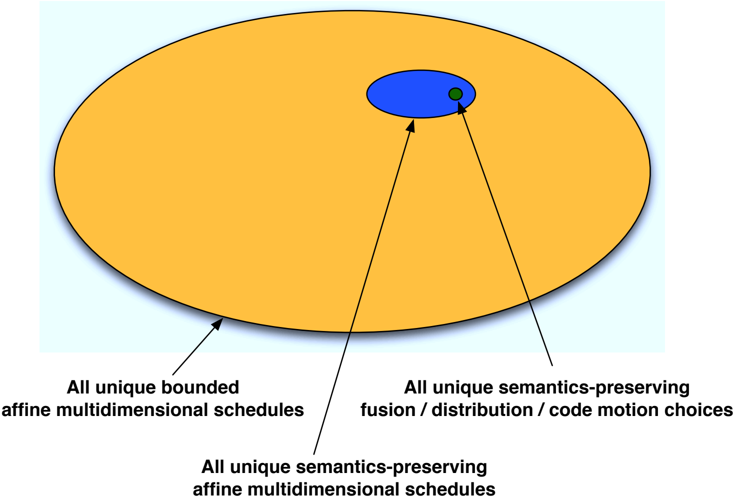
- **High-level loop transformations are critical for performance...**
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)
- **... But deciding the best sequence of transformations is hard!**
  - Conflicting objectives: more SIMD implies less locality, etc.
  - It is machine-dependent and of course program-dependent
  - Expressive search spaces are required, but challenge the search!

- **Our approach:**
  - **Convexity:** model optimization spaces as convex set (ILP, scan, project, etc.)
  - **Pruning:** make our spaces contain all and only semantically equivalent programs in our framework
  - **Optimization:** decompose in two more tractable sub-problems without any loss of expressiveness, empirical search + ILP models

Spaces of Affine Loop transformations

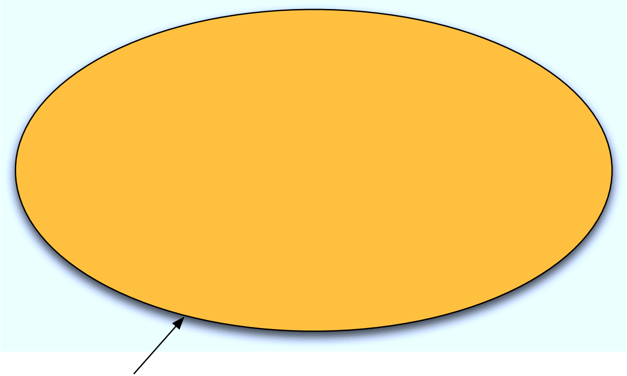


Spaces of Affine Loop transformations



Bounded: 10^{200} Legal: 10^{50} Empirical search: 10

Spaces of Affine Loop transformations



All unique bounded
affine multidimensional schedules

1 point $\leftrightarrow$ 1 unique transformed program

Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)

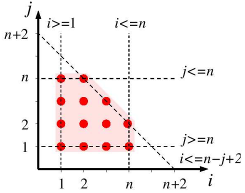
Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)
- Iteration domain: represented as integer polyhedra

```
for (i=1; i<=n; ++i)
. for (j=1; j<=n; ++j)
. . if (i<=n-j+2)
. . . s[i] = ...
```

$$\mathcal{D}_{S1} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ -1 & -1 & 1 & 2 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix} \geq \vec{0}$$



Iteration domain of S_1

Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)
- Iteration domain: represented as integer polyhedra
- Memory accesses: static references, represented as affine functions of $\vec{x}_S$ and $\vec{p}$

```
for (i=0; i<n; ++i) {
. s[i] = 0;
. for (j=0; j<n; ++j)
. . s[i] = s[i]+a[i][j]*x[j];
}
```

$$f_{\mathbf{s}}(\vec{x}_{S2}) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} . \begin{pmatrix} \vec{x}_{S2} \\ n \\ 1 \end{pmatrix}$$

$$f_{\mathbf{a}}(\vec{x}_{S2}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} . \begin{pmatrix} \vec{x}_{S2} \\ n \\ 1 \end{pmatrix}$$

$$f_{\mathbf{x}}(\vec{x}_{S2}) = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} . \begin{pmatrix} \vec{x}_{S2} \\ n \\ 1 \end{pmatrix}$$

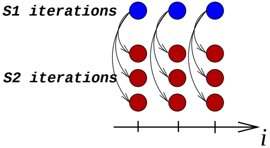
Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)
- Iteration domain: represented as integer polyhedra
- Memory accesses: static references, represented as affine functions of $\vec{x}_S$ and $\vec{p}$
- Data dependence between S1 and S2: a subset of the Cartesian product of $\mathcal{D}_{S1}$ and $\mathcal{D}_{S2}$ (**exact analysis**)

```
for (i=1; i<=3; ++i) {
. s[i] = 0;
. for (j=1; j<=3; ++j)
. . s[i] = s[i] + 1;
}
```

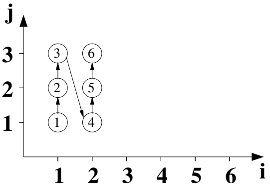
$$\mathcal{D}_{S1\delta S2} : \left[\begin{array}{cccc} 1 & -1 & 0 & 0 \\ \hline 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 3 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 3 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 3 \end{array}\right] \cdot \begin{pmatrix} i_{S1} \\ i_{S2} \\ j_{S2} \\ 1 \end{pmatrix} \begin{array}{c} = 0 \\ \hline \geq \vec{0} \end{array}$$



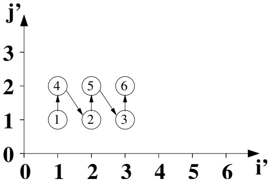
Affine Transformations for Iteration Reordering

Interchange Transformation

The transformation matrix is the identity with a permutation of two rows.



$\Longrightarrow$



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

(a) original polyhedron

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

(b) transformation function

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

(c) target polyhedron

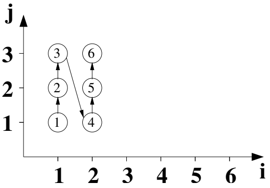
```
do i = 1, 2
  do j = 1, 3
     S(i,j)
```

```
do i' = 1, 3
  do j' = 1, 2
     S(i=j',j=i')
```

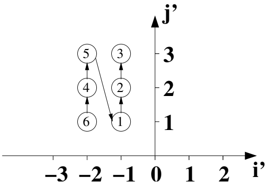
Affine Transformations for Iteration Reordering

Reversal Transformation

The transformation matrix is the identity with one diagonal element replaced by -1.



$\Longrightarrow$



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

(a) original polyhedron

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

(b) transformation function

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

(c) target polyhedron

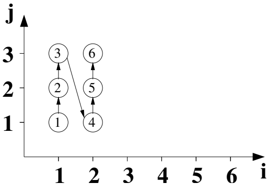
```
do i = 1, 2
  do j = 1, 3
     S(i,j)
```

```
do i' = -1, -2, -1
  do j' = 1, 3
     S(i=3-i',j=j')
```

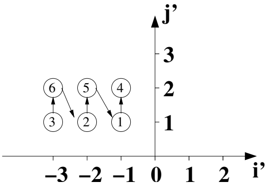
Affine Transformations for Iteration Reordering

Coumpound Transformation

The transformation matrix is the composition of an interchange and reversal



$\Longrightarrow$



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

(a) original polyhedron

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

(b) transformation function

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

(c) target polyhedron

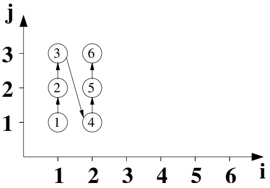
```
do i = 1, 2
  do j = 1, 3
     S(i,j)
```

```
do j' = -1, -3, -1
  do i' = 1, 2
     S(i=4-j',j=i')
```

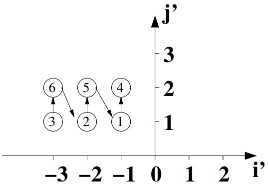
Affine Transformations for Iteration Reordering

Coumpound Transformation

The transformation matrix is the composition of an interchange and reversal



$\Longrightarrow$



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

(a) original polyhedron

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

(b) transformation function

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

(c) target polyhedron

```
do i = 1, 2
  do j = 1, 3
     S(i,j)
```

```
do j' = -1, -3, -1
  do i' = 1, 2
     S(i=4-j',j=i')
```

Affine Schedule

Definition (Affine multidimensional schedule)

Given a statement S, an affine schedule Θ^S of dimension m is an affine form on the d outer loop iterators $\vec{x}_S$ and the p global parameters $\vec{n}$.
$\Theta^S \in \mathbb{Z}^{m\times(d+p+1)}$ can be written as:

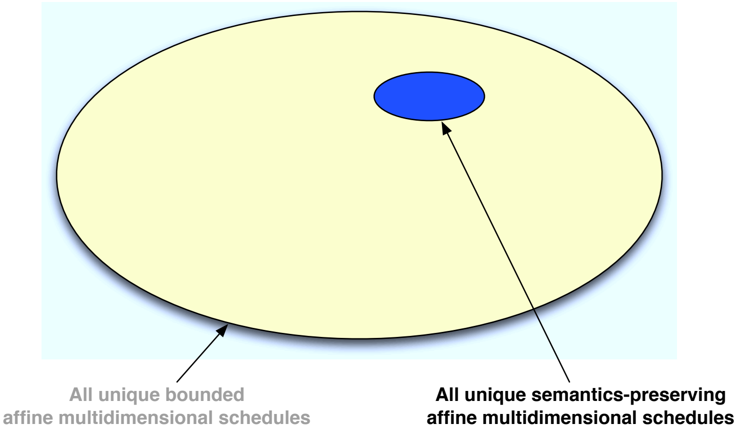
$$\Theta^S(\vec{x}_S) = \begin{pmatrix} \theta_{1,1} & \dots & \theta_{1,d+p+1} \\ \vdots & & \vdots \\ \theta_{m,1} & \dots & \theta_{m,d+p+1} \end{pmatrix} . \begin{pmatrix} \vec{x}_S \\ \vec{n} \\ 1 \end{pmatrix}$$

Θ^S_k denotes the kth row of Θ^S.

Definition (Bounded affine multidimensional schedule)

Θ^S is a bounded schedule if $\theta^S_{i,j} \in [x,y]$ with $x,y \in \mathbb{Z}$

Space of Semantics-Preserving Affine Schedules



1 point $\leftrightarrow$ 1 unique semantically equivalent program (up to affine iteration reordering)

Semantics Preservation

Definition (Causality condition)

Given Θ^R a schedule for the instances of R, Θ^S a schedule for the instances of S. Θ^R and Θ^S preserve the dependence $\mathcal{D}_{R,S}$ if $\forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S}$:

$$\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$$

$\prec$ denotes the *lexicographic ordering*.

$(a_1, \ldots, a_n) \prec (b_1, \ldots, b_m)$ iff $\exists i,\ 1 \leq i \leq min(n,m)$ s.t. $(a_1, \ldots, a_{i-1}) = (b_1, \ldots, b_{i-1})$ and $a_i < b_i$

Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \succ \vec{0}$

Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \succ \vec{0}$
- Considering the row p of the scheduling matrices:

$$\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) \geq \delta_p$$

Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \succ \vec{0}$
- Considering the row p of the scheduling matrices:

$$\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) \geq \delta_p$$

  - $\delta_p \geq 1$ implies no constraints on δ_k, $k > p$
  - $\delta_p \geq 0$ is required if $\nexists k < p$, $\delta_k \geq 1$

Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \succ \vec{0}$
- Considering the row p of the scheduling matrices:

$$\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) \geq \delta_p$$

  - $\delta_p \geq 1$ implies no constraints on δ_k, $k > p$
  - $\delta_p \geq 0$ is required if $\nexists k < p$, $\delta_k \geq 1$

- Schedule lower bound:

Lemma (Schedule lower bound)

Given Θ^R_k, Θ^S_k such that each coefficient value is bounded in $[x,y]$. Then there exists $K \in \mathbb{Z}$ such that:

$$\Theta^S_k(\vec{x}_S) - \Theta^R_k(\vec{x}_R) > -K.\vec{n} - K$$

Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \dots$ of dimension m, the program semantics is preserved if the three following conditions hold:

(i) $\forall \mathcal{D}_{R,S},\ \delta_p^{\mathcal{D}_{R,S}} \in \{0,1\}$

(ii) $\forall \mathcal{D}_{R,S},\ \sum_{p=1}^{m} \delta_p^{\mathcal{D}_{R,S}} = 1$

(iii) $\forall \mathcal{D}_{R,S},\ \forall p \in \{1,\dots,m\},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},$

Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ *of dimension* m*, the program semantics is preserved if the three following conditions hold:*

(i) $\forall \mathcal{D}_{R,S},\ \delta_p^{\mathcal{D}_{R,S}} \in \{0,1\}$

(ii) $\forall \mathcal{D}_{R,S},\ \sum_{p=1}^{m} \delta_p^{\mathcal{D}_{R,S}} = 1$

(iii) $\forall \mathcal{D}_{R,S},\ \forall p \in \{1,\ldots,m\},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},$

Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \dots$ of dimension m, the program semantics is preserved if the three following conditions hold:

(i) $$\forall \mathcal{D}_{R,S},\ \delta_p^{\mathcal{D}_{R,S}} \in \{0,1\}$$

(ii) $$\forall \mathcal{D}_{R,S},\ \sum_{p=1}^{m} \delta_p^{\mathcal{D}_{R,S}} = 1$$

(iii) $$\forall \mathcal{D}_{R,S},\ \forall p \in \{1,\dots,m\},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},$$

$$\Theta_p^S(\vec{x}_S) - \Theta_p^R(\vec{x}_R) \geq \delta_p^{\mathcal{D}_{R,S}}$$

Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \dots$ *of dimension* m, *the program semantics is preserved if the three following conditions hold:*

(i) $\forall \mathcal{D}_{R,S},\ \delta_p^{\mathcal{D}_{R,S}} \in \{0,1\}$

(ii) $\forall \mathcal{D}_{R,S},\ \sum_{p=1}^{m} \delta_p^{\mathcal{D}_{R,S}} = 1$

(iii) $\forall \mathcal{D}_{R,S},\ \forall p \in \{1,\dots,m\},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},$

$$\Theta_p^S(\vec{x}_S) - \Theta_p^R(\vec{x}_R) \geq \delta_p^{\mathcal{D}_{R,S}} - \sum_{k=1}^{p-1} \delta_k^{\mathcal{D}_{R,S}}.(K.\vec{n} + K)$$

Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension m, the program semantics is preserved if the three following conditions hold:

(i) $$\forall \mathcal{D}_{R,S},\ \delta_p^{\mathcal{D}_{R,S}} \in \{0,1\}$$

(ii) $$\forall \mathcal{D}_{R,S},\ \sum_{p=1}^{m} \delta_p^{\mathcal{D}_{R,S}} = 1$$

(iii) $$\forall \mathcal{D}_{R,S},\ \forall p \in \{1,\ldots,m\},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},$$

$$\Theta_p^S(\vec{x}_S) - \Theta_p^R(\vec{x}_R) - \delta_p^{\mathcal{D}_{R,S}} + \sum_{k=1}^{p-1} \delta_k^{\mathcal{D}_{R,S}}.(K.\vec{n}+K) \geq 0$$

$\longrightarrow$ Use **Farkas lemma to build all non-negative functions over a polyhedron** (here, the dependence polyhedra) [Feautrier,92]

Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension m, the program semantics is preserved if the three following conditions hold:

(i) $\forall \mathcal{D}_{R,S},\ \delta_p^{\mathcal{D}_{R,S}} \in \{0,1\}$

(ii) $\forall \mathcal{D}_{R,S},\ \sum_{p=1}^{m} \delta_p^{\mathcal{D}_{R,S}} = 1$

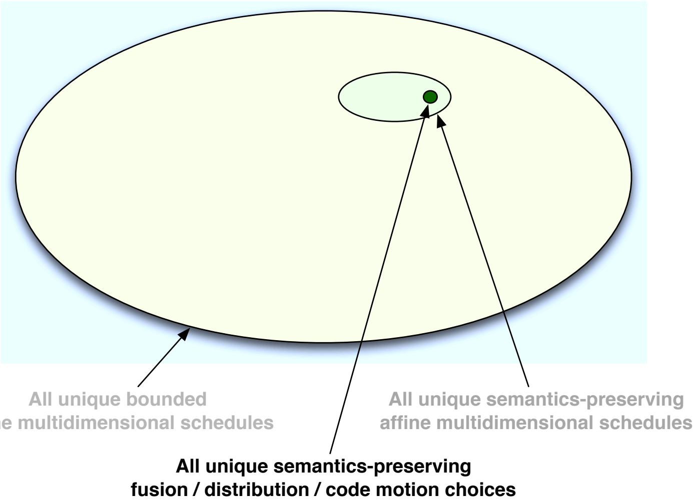
(iii) $\forall \mathcal{D}_{R,S},\ \forall p \in \{1,\ldots,m\},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},$

$$\Theta_p^S(\vec{x}_S) - \Theta_p^R(\vec{x}_R) - \delta_p^{\mathcal{D}_{R,S}} + \sum_{k=1}^{p-1} \delta_k^{\mathcal{D}_{R,S}}.(K.\vec{n} + K) \geq 0$$

→ Use **Farkas lemma to build all non-negative functions over a polyhedron** (here, the dependence polyhedra) [Feautrier,92]

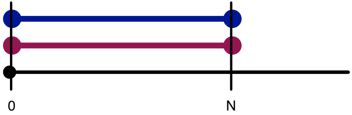
→ Bounded coefficients required [Vasilache,07]

Space of Semantics-Preserving Fusion Choices



1 point $\leftrightarrow$ 1 unique semantically equivalent program
(up to "partial" statement reordering)

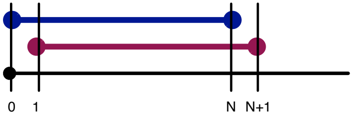
Fusion in the Polyhedral Model




```
for (i = 0; i <= N; ++i) {
  Blue(i);
  Red(i);
}
```

Perfectly aligned fusion

Fusion in the Polyhedral Model

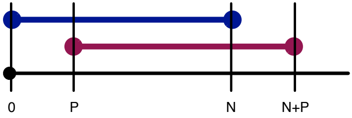



```
Blue(0);
for (i = 1; i <= N; ++i) {
  Blue(i);
  Red(i-1);
}
Red(N);
```

Fusion with shift of 1

Not all instances are fused

Fusion in the Polyhedral Model

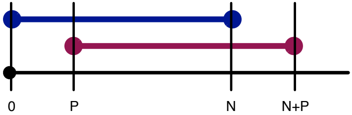



```
for (i = 0; i < P; ++i)
  Blue(i);
for (i = P; i <= N; ++i) {
  Blue(i);
  Red(i-P);
}
for (i = N+1; i <= N+P; ++i)
  Red(i-P);
```

Fusion with parametric shift of P

Automatic generation of prolog/epilog code

Fusion in the Polyhedral Model




```
for (i = 0; i < P; ++i)
  Blue(i);
for (i = P; i <= N; ++i) {
  Blue(i);
  Red(i-P);
}
for (i = N+1; i <= N+P; ++i)
  Red(i-P);
```

Many other transformations may be required to enable fusion: interchange, skewing, etc.

Affine Constraints for Fusibility

- **Two statements can be fused if their timestamp can overlap**

Definition (Generalized fusibility check)

Given v_R (resp. v_S) the set of vertices of $\mathcal{D}_R$ (resp. $\mathcal{D}_S$). R and S are fusible at level p if, $\forall k \in \{1 \ldots p\}$, there exist two semantics-preserving schedules Θ^R_k and Θ^S_k such that

$$\exists(\vec{x}_1, \vec{x}_2, \vec{x}_3) \in v_R \times v_S \times v_R, \qquad \Theta^R_k(\vec{x}_1) \leq \Theta^S_k(\vec{x}_2) \leq \Theta^R_k(\vec{x}_3)$$

- Intersect $\mathcal{L}$ with fusibility and distribution constraints
- **Completeness:** if the test fails, then there is no sequence of affine transformations that can implement this fusion structure

Fusion / Distribution / Code Motion

Our strategy:

1. Build a set containing all unique fusion / distribution / code motion combinations
2. Prune all combinations that do not preserve the semantics

Given two statements R and S, three choices:

1. R is *fully before* S $\longrightarrow$ distribution + code motion
2. R is *fully after* S $\longrightarrow$ distribution + code motion
3. otherwise $\longrightarrow$ fusion

$\Rightarrow$ It corresponds to all total preorders of R and S

Affine Encoding of Total Preorders

Principle:

- Model a total preorder with 3 binary variables

  $p_{i,j} : \ i < j \qquad s_{i,j} : \ i > j \qquad e_{i,j} : \ i = j$
- Enforce totality and mutual exclusion
- Enforce all cases of transitivity through affine inequalities connecting some variables. Ex: $e_{i,j} = 1 \wedge e_{j,k} = 1 \Rightarrow e_{i,k} = 1$

Affine Encoding of Total Preorders

Principle:

- Model a total preorder with 3 binary variables

  $p_{i,j} : \; i < j \qquad s_{i,j} : \; i > j \qquad e_{i,j} : \; i = j$

- Enforce totality and mutual exclusion
- Enforce all cases of transitivity through affine inequalities connecting some variables. Ex: $e_{i,j} = 1 \wedge e_{j,k} = 1 \Rightarrow e_{i,k} = 1$

- **This set contains one and only one point per distinct total preorder of n elements**

Affine Encoding of Total Preorders

Principle:

- Model a total preorder with 3 binary variables

  $p_{i,j} : \; i < j \qquad s_{i,j} : \; i > j \qquad e_{i,j} : \; i = j$
- Enforce totality and mutual exclusion
- Enforce all cases of transitivity through affine inequalities connecting some variables. Ex: $e_{i,j} = 1 \wedge e_{j,k} = 1 \Rightarrow e_{i,k} = 1$

- **This set contains one and only one point per distinct total preorder of n elements**
- Easy pruning: just bound the sum of some variables

  e.g., $e_{1,2} + e_{4,5} + e_{8,12} < 3$
- Automatic removal of supersets of unfusible sets

Convex set of All Unique Total Preorders

$$O = \left\{ \begin{array}{c} 0 \le p_{i,j} \le 1 \\ 0 \le e_{i,j} \le 1 \\ 0 \le s_{i,j} \le 1 \end{array} \right\}$$

constrained to:

$$O = \left\{ \begin{array}{lrl}
 & \left.\begin{array}{r} 0 \le p_{i,j} \le 1 \\ 0 \le e_{i,j} \le 1 \end{array}\right\} & \text{Variables are binary} \\
 & \left. p_{i,j} + e_{i,j} \le 1 \right\} & \text{Relaxed mutual exclusion} \\
\forall k \in]j,n] & \left.\begin{array}{r} e_{i,j} + e_{i,k} \le 1 + e_{j,k} \\ e_{i,j} + e_{j,k} \le 1 + e_{i,k} \end{array}\right\} & \text{Basic transitivity on } e \\
\forall k \in]i,j[& \left. p_{i,k} + p_{k,j} \le 1 + p_{i,j} \right\} & \text{Basic transitivity on } p \\
\begin{array}{l} \forall k \in]j,n] \\ \\ \forall k \in]i,j[\end{array} & \left.\begin{array}{r} e_{i,j} + p_{i,k} \le 1 + p_{j,k} \\ e_{i,j} + p_{j,k} \le 1 + p_{i,k} \\ e_{k,j} + p_{i,k} \le 1 + p_{i,j} \end{array}\right\} & \text{Complex transitivity on } p \text{ and } e \\
\forall k \in]j,n] & \left. e_{i,j} + p_{i,j} + p_{j,k} \le 1 + p_{i,k} + e_{i,k} \right\} & \text{Complex transitivity on } s \text{ and } p
\end{array} \right.$$

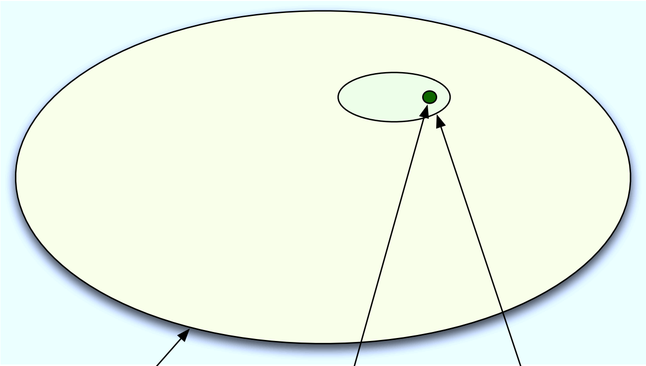
- Systematic construction for a given n, needs n^2 Boolean variables
- **Enable ILP modeling, enumeration, etc.**
- Extension to multidimensional total preorders (i.e., multi-level fusion)

Pruning for Semantics Preservation

Intuition: enumerate the smallest sets of unfusible statements

- Use an intermediate structure to represent sets of statements
  - Graph representation of maybe-unfusible sets (1 node per statement)
  - Enumerate sets from the smallest to the largest
- Leverage dependence graph + properties of fusion / distribution
- Compute properties by intersecting $\mathcal{L}$ with additional fusion / distribution / code motion affine constraints
- Any individual point can be removed from $\mathcal{O}$

Space of Semantics-Preserving Fusion Choices



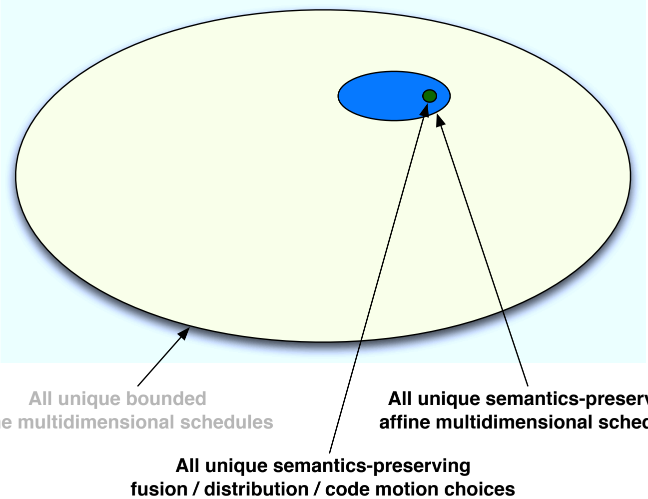
All unique bounded
affine multidimensional schedules

All unique semantics-preserving
affine multidimensional schedules

**All unique semantics-preserving
fusion / distribution / code motion choices**

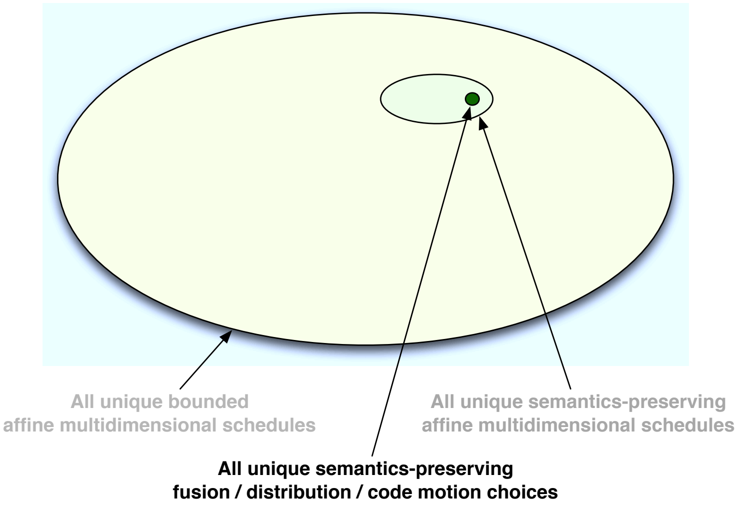
1 point $\leftrightarrow$ 1 unique semantically equivalent program
(up to statement reordering)

Space of Semantics-Preserving Fusion Choices



1 point ↔ **many** unique semantically equivalent programs (up to iteration reordering)

Space of Semantics-Preserving Fusion Choices



1 point $\leftrightarrow$ **1** unique semantically equivalent program
(up to limited iteration reordering)

Objectives for Effective Optimization

Objectives:

- Achieve efficient coarse-grain parallelization
- Combine iterative search of profitable transformations for tiling
  - $\longrightarrow$ loop fusion and loop distribution

Tiling Hyperplane method [Bondhugula,08]

- Model-driven approach for automatic parallelization + locality improvement
- Tiling-oriented
- Poor model-driven heuristic for the selection of loop fusion (not portable)
- Overly relaxed definition of fused statements

Fusibility Restricted to Non-negative Schedules

- Fusibility is not a transitive relation!
  - Example: sequence of matrix-by-vector products $x = Ab,\ y = Bx,\ z = Cy$
  - $x = Ab,\ y = Bx$ can be fused, also $y = Bx,\ z = Cy$
  - They cannot be fused all together

- **Determining the Fusibility of a group of statements is reducible to exhibiting compatible pairwise loop permutations**
  - Extremely easy to compute all possible loop permutations that lead to fuse a pair of statements
  - Never check $\mathcal{L}$ on more than two statements!

- **Stronger definition of fusion**
  - Guarantee **at most c instances are not fused**

$$-c < \Theta_k^R(\vec{0}) - \Theta_k^S(\vec{0}) < c$$

  - No combinatorial choice

The Optimization Algorithm in a Nutshell

Proceeds from the outer-most loop level to the inner-most:

1. Compute the space of valid fusion/distribution/code motion choices
2. **Select a fusion/distribution/code motion scheme** in this space
3. Compute an affine schedule **that implements this scheme**
   - Static cost model to select the schedule
   - Compound of skewing, shifting, fusion, distribution, interchange, tiling and parallelization (OpenMP)
   - **Maximize locality** for each set of statements to be fused

Experimental Results

				$\mathcal{O}$			$\mathcal{F}^1$					
Benchmark	#loops	#stmts	#refs	#dim	#cst	#points	#dim	#cst	#points	Time	perf-Intel	perf-AMD
advect3d	12	4	32	12	58	75	9	43	26	0.82s	1.47×	5.19×
atax	4	4	10	12	58	75	6	25	16	0.06s	3.66×	1.88×
bicg	3	4	10	12	58	75	10	52	26	0.05s	1.75×	1.40×
gemver	7	4	19	12	58	75	6	28	8	0.06s	1.34×	1.33×
ludcmp	**9**	**14**	**35**	**182**	**3003**	$\approx 10^{12}$	**40**	**443**	**8**	**0.54s**	1.98×	1.45×
doitgen	5	3	7	6	22	13	3	10	4	0.08s	15.35×	14.27×
varcovar	**7**	**7**	**26**	**42**	**350**	**47293**	**22**	**193**	**96**	**0.09s**	7.24×	14.83×
correl	5	6	12	30	215	4683	21	162	176	0.09s	3.00×	3.44×

Table: Search space statistics and performance improvement

- **Performance portability:** empirical search on the target machine of the optimal fusion structure
- Outperforms state-of-the-art cost models
- Full implementation in the source-to-source polyhedral compiler PoCC

Conclusion

Take-home message:

⇒ **Clear formalization of loop fusion** in the polyhedral model

⇒ **Formal definition of all semantically equivalent programs** up to:

- statement reordering
- limited affine iteration reordering
- arbitrary affine iteration reordering

⇒ **Effective and portable hybrid empirical optimization algorithm** (parallelization + data locality)

Future work:

- Develop static cost models for fusion / distribution / code motion
- Use statistical techniques to learn optimization algorithms