On the Validity Problem for Unimodular Transformations *

Peter M.W. Knijnenburg

High Performance Computing Division,
Dept. of Computer Science, Leiden University,
Niels Bohrweg 1, 2333 CA Leiden, the Netherlands.
peterk@cs.leidenuniv.nl

Abstract

In this paper a new approach to the validity problem of unimodualr transformations is proposed. First, we define a representation of data dependences by means of convex polytopes. Second, it is shown how this representation can be used to test the validity of unimodular transformations. This method is strictly more powerful than methods based on dependence direction vectors, as proposed by Wolfe, or interval vector, as proposed by Wolf and Lam. We show how our method can be extended to cover the situations where affine guards are present in the loop body, where multiple transformations or displacements are used to transform the loop, as well.

1 Introduction

The application of loop level transformations is one of the major techniques for restructuring programs in order to exploit their inherent parallelism and achieve high performance on current vector and massively parallel architectures [Ban93, PW86, Pol88, Wol91, ZC90]. Recently significant progress was made by the recognition that many important transformations can be described by means of unimodular matrices [AI91, Ban93, Ban91, WL91]. The theory has also been extended to cover general non-singular integer matrices [LP92]. In this paper we focus on unimodular transformations. In section 5 we show how the techniques can conservatively be extended to the case of non-singular integer matrices.

One of the main advantages of using matrices lies in the ability to test for the legality of a transformation in a very concise way. Suppose that there exists a dependence from iteration $\vec{\mathtt{I}}$ to $\vec{\mathtt{I}}'$. Then it holds that in the transformed iteration space there is a dependence from iteration $\mathcal{U}\vec{\mathtt{I}}$ to $\mathcal{U}\vec{\mathtt{I}}'$, where $\mathcal{U}$ is the matrix specifying the transformation. Hence the distance of the dependence in the transformed iteration space equals $\mathcal{U}\vec{\mathtt{I}}' - \mathcal{U}\vec{\mathtt{I}} = \mathcal{U}(\vec{\mathtt{I}}' - \vec{\mathtt{I}})$. That is, dependence distances are also transformed by the same transformation as used for the iteration space. From this it follows that a transformation $\mathcal{U}$ is valid, that is, respects the dependence structure of a loop, iff for each dependence distance $\vec{\mathtt{d}}$ it is the case that $\mathcal{U}\vec{\mathtt{d}}$ is lexicographically positive. This last assertion means that the sink of a dependence is executed after the source.

So the validity problem of a unimodular transformation boils down to the ability to compute a (compact representation of) the collection of dependence distances in a loop. If the dependence is uniform, this can easily be done. In this case, there exists a fixed dependence distance vector $\vec{\mathtt{d}}$ such that for each iteration $\vec{\mathtt{I}}$, the iteration $\vec{\mathtt{I}} + \vec{\mathtt{d}}$ is dependent on it. Hence we can check for the validity of the transformation by checking whether $\mathcal{U}\vec{\mathtt{d}}$ is lexicographically positive. There exist a number of algorithms for computing uniform dependences. The reader is referred to [Ban93, Ban88, Pol88, Wol91, ZC90] for background on the theory.

The situation is different if the dependence is not uniform. In this case there exists a possibly large number of different dependence distances for the given non-uniform dependence. It is difficult for a compiler to construct this collection explicitly. Hence it is important to be able to device a compact representation for this collection. Traditionally, there are two such representations in wide use. The first is the abstraction of the *dependence direction vector* [Wol91]. In this approach, all possible signs of the distances in each dimension are collected to form a vector over $\{<, >, \leq, \geq, =, *\}$. The penalty we have to pay for this abstraction is that we loose precision. On the other hand, for a number of important transforma-

*This research was partially supported by Esprit BRA APPARC under grant no. 6634

tions like loop interchange this precision is not necessary and direction vectors are precise enough to decide their applicability [YAI94]. The second abstraction one encounters in the literature is proposed by Wolf and Lam and consists of giving the intervals in which the components of the distance vectors are contained [WL91]. They have given a calculus for such intervals thus enabling to conservatively check for the validity of a transformation. But like in the previous case all distance vectors are lumped together which makes these interval vectors imprecise. Hence using these intervals, one may decide that a certain transformation is not valid where in fact it is. In section 5 we give an example in which both approaches conservatively but wrongly conclude that a dependence is violated by a transformation, where the representation we propose decides it is not.

The representation we propose is the following. For a dependence δ we collect every source and sink of the different individual dependences in convex polytopes. These polytopes can be described by a system of inequalities. Hence we only need to store $2n$ bounds, where n is the dimension of the iteration space. These bounds can be obtained by Fourier-Motzkin elimination [DE73, Ban93, BW94, LP92]. Next, to check the validity of the application of a unimodular transformation $\mathcal{U}$, we transform these polytopes by an associated transformation $\mathcal{U}^*$. The resulting polytopes exactly contain the sources and sinks of the transformed dependence. We then show how to check whether all these transformed dependences are lexicographically positive. We again use Fourier-Motzkin elimination for this.

The representation and the validity test discussed in this paper are very flexible. In section 4 we show how to deal with so-called affine guards, multiple mappings, and displacements in the framework. Once the basic techniques have been developed, these extensions can be incorporated almost without effort.

The way we obtain the collection of dependence distance vectors is closely related to some other approaches. The Omega test proposed by Pugh [Pug92] is also based on Fourier-Motzkin elimination of a system of inequalities like the system we consider here. Wolfe and Tseng [WT90] propose an extension of Banerjee's Generalized GCD test [Ban88] using Fourier-Motzkin elimination. Feautrier has proposed an integer programming technique for analyzing the dependence structure of a loop [Fea91] which has strong similarities with the method described in this paper. However, these dependence tests decide whether a dependence exists, possibly yielding a direction vector. In contrast, our technique is tailored towards the validity test for unimodular transformations. That is, we do not explicitly construct dependence distance or direction vectors, but construct polytopes containing all dependence information. These polytopes are then transformed using the unimodular transformation in question and the resulting polytopes are inspected for violation of the dependence structure of the loop. To our knowledge, this approach is new. The test we propose has an intuitively clear formulation. Moreover, the technical machinery we use for both the computation of the depencence structure and the validity test for the transformation is Fourier-Motzkin elimination. Since this is a basic algorithm for applying a unimodular transformation and hence will be present in a state-of-the-art restructuring compiler, our technique can be easily implemented.

The paper is organized as follows. In section 2 we discuss some preliminaries and give notation. In section 3 we formulate our compact representation of a dependence and show how this representation can be used to test the validity of a unimodular transformation. In section 4 we show how the technique can be extended to cover some other situations as well. Finally, in section 5, we give a brief discussion.

Acknowledgement. The author wishes to thank Arjan Bik for critically reading a draft version of this paper.

2 Preliminaries

In this section we give some preliminaries we use in the rest of this paper. First, we define lower and upperbounds for the loops we will consider.

Definition 2.1 *Let $\mathcal{I} = \{\mathtt{I}_1, \ldots, \mathtt{I}_n\}$ be a finite collection of variables or loop indices. With respect to this collection $\mathcal{I}$ we define:*

1. *A* basic (lower or upper) bound *is an affine expression*
$$a_0 + a_1\mathtt{I}_1 + \cdots + a_n\mathtt{I}_n$$
where, for all i, $a_i \in \mathbf{Z}$.

2. *Let $a \in \mathbf{N}^+$ and let B be a basic bound. A* simple lowerbound *is an expression $\lceil \frac{1}{a}B \rceil$. A* simple upperbound *is an expression $\lfloor \frac{1}{a}B \rfloor$.*

3. *A* compound lowerbound *is an expression* $\max(L_1, \ldots, L_m)$ *where each L_i is a simple lowerbound. A* compound upperbound *is an expression*

$\min(U_1, \ldots, U_m)$ *where each* U_i *is a simple upperbound.*

Note that each basic bound can be considered a simple lower or upperbound, and each simple bound can be considered a compound bound. In the sequel of the paper we will make this identification when no confusion can arise. Basic, simple and compound bounds are also called *admissible*. All other expressions for bounds, like $\mathtt{I}_1 * \mathtt{I}_2$ or an indirection $\mathtt{IND(I)}$, are called *inadmissible*. The reason for this distinction is that admissible bounds can be used and are obtained in the process of Fourier-Motzkin elimination (see below).

Every perfectly nested loop $\mathcal{L}$ with basic bounds gives rise to a system of inequalities $\mathcal{S}(\mathcal{L})$, given by

$$\mathcal{S}(\mathcal{L}) = \left\{ \begin{array}{ccccc} \mathtt{L}_1 & \leq & \mathtt{I}_1 & \leq & \mathtt{U}_1 \\ & & \vdots & & \\ \mathtt{L}_n & \leq & \mathtt{I}_n & \leq & \mathtt{U}_n \end{array} \right.$$

Such a system should be read as the *conjunction* of the individual clauses. This system has the property that every bound $\mathtt{L}_i$ and $\mathtt{U}_i$ only involves variables $\mathtt{I}_1, \ldots, \mathtt{I}_{i-1}$. We call this the *standard form* of a system of inequalities. Note that for loops with compound lower and upperbounds we also can define such a system of inequalities. Since compound upperbounds may contain floor and minimum fuction, we use the following equivalences to obtain this standard form.

- $\mathtt{I} \leq \lfloor \frac{1}{a} B \rfloor$ iff $a\mathtt{I} \leq B$ (since $\mathtt{I}$ is integer and $a > 0$).
- $\mathtt{I} \leq \min(U_1, \ldots, U_m)$ iff $\mathtt{I} \leq U_1$ & $\cdots$ & $\mathtt{I} \leq U_m$.

For compound lowerbounds a similar standard form can be deduced.

Any system of inequalities involving the variables $\mathtt{I}_1, \ldots, \mathtt{I}_n$ can be brought in standard form using *Fourier-Motzkin elimination* [DE73, Ban93]. We try to give some intuition. Consider a set $\mathcal{C} = \{\varphi_1, \ldots, \varphi_k\}$ of inequalities, where each inequality φ_i is of the form $e \leq e'$ for two affine expressions e and e' over the variables $\mathtt{I}_1, \ldots, \mathtt{I}_n$. Then Fourier-Motzkin elimination consists of the following process. First, rewrite all expressions involving the variable $\mathtt{I}_n$ to the form $L \leq \mathtt{I}_n, \ldots, \mathtt{I}_n \leq U$. Then each inequality obtained in this way bounds $\mathtt{I}_n$ by expressions only involving the variables $\mathtt{I}_1, \ldots, \mathtt{I}_{n-1}$. Hence these expressions can be used to generate loop bounds. Now consider the system obtained by forming all inequalities $L \leq U$, for lowerbounds L and upperbounds U from the previous step, together with all inequalities from the original system not involving $\mathtt{I}_n$. This is a system of inequalities only involving the variables $\mathtt{I}_1, \ldots, \mathtt{I}_{n-1}$. Hence we can recursively continue the process, finally ending with a system of inequalities which consist only of the variable $\mathtt{I}_1$ and contstants.

Let $[\![\mathcal{C}]\!] = \{x \in \mathbf{R}^n : \varphi_1(x) \wedge \cdots \wedge \varphi_k(x)\}$. We say that $\mathcal{C}$ is *inconsistent* iff $[\![\mathcal{C}]\!] = \emptyset$. We say that an inequality φ is *redundant* for $\mathcal{C}$ iff $[\![\mathcal{C} \cup \{\varphi\}]\!] = [\![\mathcal{C}]\!]$. The Fourier-Motzkin elimination algoritm can be used to decide whether $\mathcal{C}$ is inconsistent. We also have that φ is redundant for $\mathcal{C}$ if and only if $\mathcal{C} \cup \{\neg\varphi\}$ is inconsistent.

Fourier-Motzkin elimination deals with systems of inequalities of the form $e \leq e'$ for affine expressions e and e'. Note that we can express other comparison operations on integers by the following identifications.

- $n < m$ iff $n + 1 \leq m$
- $n = m$ iff $n \leq m$ and $m \leq n$
- $n \geq m$ iff $m \leq n$

In the sequel we will freely use these identifications.

Next, we give a definition of dependence that we use in this paper. Given two statements S_1 and S_2 such that both reference a variable and at least one of the references is a write, then we say that there exists a *dependence* between S_1 and S_2. If S_1 is executed prior to S_2, then S_1 is the *source*, and S_2 is the *sink* of the dependence. If S_1 and S_2 are statements in a loop, then there may be a dependence between them for several iteration vectors. For instance, in a single loop with index `I`, S_1 may define `A(I)` and S_2 may use `A(I-1)`. In this case we still say that there exists a dependence between S_1 and S_2, thus lumping all individual dependences for the different iterations together. Note that in some definitions of dependence it is required that there does not exist a statement S_3 which is executed between S_1 and S_2 and which writes to the same variablei [ZC90]. Dependences with this extra condition are called *value based dependences* [PW92]. We do not require this extra condition. The resulting notion of dependence is called *memory based depedence* [PW92].

Finally, given an expression e containing a variable x, the expression $e[e'/x]$ denotes the substitution of e' for x in e. Likewise, given a collection of variables $x_1, \ldots, x_n$ and a collection of expressions $e_1, \ldots, e_n$, the expression $e[\vec{e}/\vec{x}]$ denotes the simultaneous substitution of $e_1, \ldots, e_n$ for $x_1, \ldots, x_n$, respectively, in e.

3 The validity test

In this section we describe a method for checking the validity of the application of a unimodular transfor-

mation $\mathcal{U}$ on a perfectly nested loop $\mathcal{L}$. Such an application is valid if for each dependence distance vector $\vec{\mathtt{d}}$ it is the case that $\mathcal{U}\vec{\mathtt{d}}$ is lexicographically positive [Ban93, Wol91]. The method we propose is the following. For a dependence δ we collect every source and sink of the different distances in a number of convex polytopes. We construct one polytope for each loop that may carry the dependence. These polytopes can be described by a system of inequalities. The bounds of these polytopes can be obtained by Fourier-Motzkin elimination. Next, to check the validity of the application of a unimodular transformation $\mathcal{U}$, we transform these polytopes by an associated transformation $\mathcal{U}^*$. The resulting polytopes exactly contain the transformed sources and sinks of the dependence. We then show how to check whether all these transformed distances are lexicographically positive. We again use Fourier-Motzkin elimination for this.

3.1 Dependence polytopes

Suppose we have a perfectly nested multiple loop $\mathcal{L}$ with loop indices $\mathtt{I}_1, \ldots, \mathtt{I}_n$, respectively. The system of inequalities generated by the bounds of the loop indices is denoted as

$$\mathtt{L}_1 \leq \mathtt{I}_1 \leq \mathtt{U}_1 \quad \ldots \quad \mathtt{L}_n \leq \mathtt{I}_n \leq \mathtt{U}_n$$

Each bound $\mathtt{L}_k$ and $\mathtt{U}_k$ is an admissible bound over the collection of variables $\{\mathtt{I}_1, \ldots, \mathtt{I}_{k-1}\}$. These inequalities define a convex polytope $\mathcal{P}$ which is the iteration space of the loop. Consider two statements in the loop body, S_1 and S_2. We assume that S_1 defines an indexed variable $A(e_1, \ldots, e_d)$, and S_2 uses an indexed variable $A(e'_1, \ldots, e'_d)$. The index expressions e_i and e'_i are assumed to be affine expressions over the loop indices. We denote the index functions by $\vec{e}$ and $\vec{e}'$, respectively. In order to determine whether there exists a dependence between S_1 and S_2, we have to decide whether $\vec{e}$ and $\vec{e}'$ can obtain the same value during execution of the loop.

Our aim is to define a collection of polytopes $\mathbf{P} = \{\mathcal{P}_m : 1 \leq m \leq n\}$. These polytopes are called *dependence polytopes* and they will contain all dependence information. In this paper, we consider memory based dependences [PW92].

To explain the intuition behind the construction of the collection of dependence polytopes, observe that a dependence can be carried by each loop in the nest. Therefore, for each $1 \leq m \leq n$, we define a polytope $\mathcal{P}_m$ which encodes the dependences carried by the mth loop. Consider a dependence from iteration $\vec{\mathtt{I}}$ to iteration $\vec{\mathtt{I}}'$ that is carried by the mth loop. First of all, this means that $e_i(\vec{\mathtt{I}}) = e'_i(\vec{\mathtt{I}}')$ for all $1 \leq i \leq d$. We can collect all these points in a convex polytope $\mathcal{E} \subseteq \mathcal{P} \times \mathcal{P}$, which we call the *equality polytope* for the dependence. First, we define the polytope $\mathcal{P} \times \mathcal{P}$ by introducing new variables $\vec{\mathtt{I}}'$ with the same bounds as $\vec{\mathtt{I}}$. The system of inequalities that defines $\mathcal{P} \times \mathcal{P}$ is given by

$$\begin{cases} \mathtt{L}_1 \leq \mathtt{I}_1 \leq \mathtt{U}_1, \ldots, \mathtt{L}_n \leq \mathtt{I}_n \leq \mathtt{U}_n \\ \mathtt{L}_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq \mathtt{I}'_1 \leq \mathtt{U}_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}], \ldots, \\ \qquad \mathtt{L}_n[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq \mathtt{I}'_n \leq \mathtt{U}_n[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \end{cases} \tag{1}$$

Then we add equalities derived from the index functions to isolate $\mathcal{E}$ inside $\mathcal{P} \times \mathcal{P}$.

$$e_1 = e'_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}], \ldots, e_d = e'_d[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \tag{2}$$

where we have substituted the variables $\vec{\mathtt{I}}'$ for $\vec{\mathtt{I}}$ in the (formal) expressions e'_i.

Second, the direction vector corresponding to the dependence distance vector $\vec{\mathtt{I}}' - \vec{\mathtt{I}}$ is given by

$$\langle \underbrace{=, \ldots, =}_{m-1\times}, <, *, \ldots, * \rangle$$

Hence, the corresponding point $\langle \vec{\mathtt{I}}, \vec{\mathtt{I}}' \rangle$ in $\mathcal{E}$ has the property that its first $m-1$ coordiantes of $\vec{\mathtt{I}}$ are equal to the first $m-1$ coordinates of $\vec{\mathtt{I}}'$, and the mth coordinate of $\vec{\mathtt{I}}$ is smaller than the mth coordinate of $\vec{\mathtt{I}}'$. We can filter these points out by adding an extra constraint, one for each m. These polytopes are called *dependence polytopes* and are denoted by $\mathcal{P}_m$. For each $1 \leq m \leq n$, the constraint for defining $\mathcal{P}_m$ is given by

$$\mathtt{I}_1 = \mathtt{I}'_1, \ldots, \mathtt{I}_{m-1} = \mathtt{I}'_{m-1}, \mathtt{I}_m < \mathtt{I}'_m \tag{3}$$

Observe how this constraint filters out the dependences with the proper direction vector.

Summing up, we arrive at the following definition for $\mathcal{P}_m$.

Definition 3.1 *Let $\mathcal{L}$ be a loop with bounds as above. Let S_1 and S_2 be two statements in the body of $\mathcal{L}$. Suppose that S_1 defines a subscripted variable $A(e_1, \ldots, e_d)$, and that S_2 uses a subscripted variable $A(e'_1, \ldots, e'_d)$. Then, for each $1 \leq m \leq n$, the* dependence polytope *$\mathcal{P}_m$ associated with this definition and use of A is given by the following system of inequalities.*

$$\begin{cases} \mathtt{L}_1 \leq \mathtt{I}_1 \leq \mathtt{U}_1, \ldots, \mathtt{L}_n \leq \mathtt{I}_n \leq \mathtt{U}_n \\ \mathtt{L}_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq \mathtt{I}'_1 \leq \mathtt{U}_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}], \ldots, \\ \qquad \mathtt{L}_n[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq \mathtt{I}'_n \leq \mathtt{U}_n[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \\ e_1 = e'_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}], \ldots, e_d = e'_d[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \\ \mathtt{I}_1 = \mathtt{I}'_1, \ldots, \mathtt{I}_{m-1} = \mathtt{I}'_{m-1}, \mathtt{I}_m < \mathtt{I}'_m \end{cases}$$

We can obtain a dependence polytope $\mathcal{P}_\infty$ for the loop independent dependences by using the equalities $\mathtt{I}_1 = \mathtt{I}'_1, \ldots, \mathtt{I}_n = \mathtt{I}'_n$. In this case, S_1 should textually preceed S_2 in the loop. But, since the loop transformations considered in this paper act on the whole body of a loop, these dependence will always be satisfied in the transformed loop. Hence we need not consider $\mathcal{P}_\infty$ in this paper.

Lemma 3.2 *There does not exists a dependence from S_1 to S_2 if and only if for each $1 \leq m \leq n$ the polytope $\mathcal{P}_m$ does not contain integer points.*

Corollary 3.3 *There does not exists a dependence from S_1 to S_2 if for each $1 \leq m \leq n$ the polytope $\mathcal{P}_m$ is empty.*

Example 1. Consider the following loop.

```
DO  I1 = 1,10
    DO  I2 = 1,10
        S1: A(I1,I2) = ···
        S2: ··· = ··· A(I1 - 1,I2 - 1) ···
    ENDDO
ENDDO
```

The dependence polytope for the flow dependence from S_1 to S_2 $\mathcal{P}_1$ is given by the following system of inequalities.

$$\left\{ \begin{array}{rcl} 1 \leq & \mathtt{I}_1 & \leq 9 \\ 1 \leq & \mathtt{I}_2 & \leq 9 \\ \mathtt{I}_1 + 1 \leq & \mathtt{I}'_1 & \leq \mathtt{I}_1 + 1 \\ \mathtt{I}_2 + 1 \leq & \mathtt{I}'_2 & \leq \mathtt{I}_2 + 1 \end{array} \right.$$

Its is easy to see that every point in this polytope constitutes the source and the sink of a dependence. The dependence polytope $\mathcal{P}_2$ is empty, which reflects the fact that there does not exist a flow dependence between iterations $\langle \mathtt{I}_1, \mathtt{I}_2 \rangle$ and $\langle \mathtt{I}'_1, \mathtt{I}'_2 \rangle$ with $\mathtt{I}_1 = \mathtt{I}'_1$. □

Example 2. Consider the following loop.

```
DO  I1 = 1,10
    DO  I2 = 1,I1
        S1: A(I1) = ···
        S2: ··· = ··· A(I1 - I2) ···
    ENDDO
ENDDO
```

The dependence polytope for the flow dependence from S_1 to S_2 $\mathcal{P}_1$ is given by the following system of inequalities.

$$\left\{ \begin{array}{rcl} 1 \leq & \mathtt{I}_1 & \leq 9 \\ 1 \leq & \mathtt{I}_2 & \leq \mathtt{I}_1 \\ \mathtt{I}_1 + 1 \leq & \mathtt{I}'_1 & \leq 10 \\ \mathtt{I}'_1 - \mathtt{I}_1 \leq & \mathtt{I}'_2 & \leq \mathtt{I}'_1 - \mathtt{I}_1 \end{array} \right.$$

For instance, there exists a dependence from iteration $\langle 3, 2 \rangle$ to $\langle 5, 2 \rangle$. The dependence polytope $\mathcal{P}_2$ is empty. □

3.2 Transforming the dependence polytopes

Suppose we want to transform the loop using a unimodular matrix $\mathcal{U}$. Consider the matrix $\mathcal{U}^*$ given by

$$\mathcal{U}^* = \begin{pmatrix} \mathcal{U} & 0 \\ 0 & \mathcal{U} \end{pmatrix}$$

Lemma 3.4 *If $\mathcal{U}$ is unimodular, then so is $\mathcal{U}^*$.*

Now consider the application of $\mathcal{U}^*$ to a polytope $\mathcal{P}_m$, with $1 \leq m < n$. Given a point $\langle \vec{\mathtt{I}}, \vec{\mathtt{I}}' \rangle \in \mathcal{P}_m$,

$$\mathcal{U}^* \begin{pmatrix} \vec{\mathtt{I}} \\ \vec{\mathtt{I}}' \end{pmatrix} = \begin{pmatrix} \mathcal{U}\vec{\mathtt{I}} \\ \mathcal{U}\vec{\mathtt{I}}' \end{pmatrix}$$

Hence applying $\mathcal{U}^*$ to $\mathcal{P}_m$ produces a polytope $\mathcal{P}^*_m$ which contains exactly the source and sink of each dependence after application of $\mathcal{U}$. The application of $\mathcal{U}^*$ is done in the standard way [Ban93, LP92]. Briefly, suppose a convex polytope (iteration space) $\mathcal{P}$ is defined using a system of inequalities involving indices $\vec{\mathtt{I}}$ and suppose we want to transform $\mathcal{P}$ using a unimodular matrix $\mathcal{U}$. For the iteration space of a loop $\mathcal{L}$ this system is $\mathcal{S}(\mathcal{L})$ as given in section 2. The resulting transformed polytope $\mathcal{P}^*$ will be given by a system of inequalities involving the indices $\vec{\mathtt{J}}$. We then have that $\vec{\mathtt{J}} = \mathcal{U}\vec{\mathtt{I}}$, or that $\vec{\mathtt{I}} = \mathcal{U}^{-1}\vec{\mathtt{J}}$. We substitute $\mathcal{U}^{-1}\vec{\mathtt{J}}$ for $\vec{\mathtt{I}}$ in the original system of inequalities, thereby obtaining a system of inequalities involving the indices $\vec{\mathtt{J}}$. This system defines $\mathcal{P}^*$. We can use Fourier-Motzkin elimination to obtain a standard form for this system.

We denote $\mathcal{P}^*_m$ by a system of inequalities

$$\left\{ \begin{array}{l} \mathtt{L}'_1 \leq \mathtt{J}_1 \leq \mathtt{U}'_1, \quad \ldots \quad \mathtt{L}_n \leq \mathtt{J}_n \leq \mathtt{U}`_n, \\ \mathtt{L}''_1 \leq \mathtt{J}'_1 \leq \mathtt{U}''_1, \quad \ldots \quad ,\mathtt{L}''_n \leq \mathtt{J}'_n \leq \mathtt{U}''_n \end{array} \right.$$

Example 2. (Continued) Suppose we want to transform the loop from Example 2 by the unimodular transformation

$$\mathcal{U} = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$$

The dependence polytope $\mathcal{P}_1$ from Example 2 is transformed to the polytope $\mathcal{P}^*_1$ given by

$$\left\{ \begin{array}{rcl} 1 \leq & \mathtt{J}_1 & \leq 9 \\ 0 \leq & \mathtt{J}_2 & \leq \mathtt{J}_1 - 1 \\ \mathtt{J}_1 + 1 \leq & \mathtt{J}'_1 & \leq 10 \\ \mathtt{I}_1 \leq & \mathtt{J}'_2 & \leq \mathtt{I}_1 \end{array} \right.$$

□

3.3 Checking for validity

In this section we show how to check whether the transformed dependences are lexicographically positive. This means that we have to check that for all $1 \leq k \leq n$, a direction vector of the form

$$\langle \underbrace{=, \ldots, =}_{k-1\times}, >, *, \ldots, * \rangle$$

does not exist in the transformed polytopes $\mathcal{P}^*_m$.

First we define constraints on the indices of the transformed dependence polytopes $\mathcal{P}^*_m$ that check whether there exists a point $\langle \mathtt{J}_1, \ldots, \mathtt{J}_n, \mathtt{J}'_1, \ldots, \mathtt{J}'_n \rangle$ in $\mathcal{P}^*_m$ such that for some $1 \leq k \leq n$ it is the case that $\mathtt{J}'_k < \mathtt{J}_k$ and for all $1 \leq i < k$ that $\mathtt{J}_i = \mathtt{J}'_i$. If this is so, then the transformed dependence distance

$$\langle \mathtt{J}'_1, \ldots, \mathtt{J}'_n \rangle - \langle \mathtt{J}_1, \ldots, \mathtt{J}_n \rangle$$

is lexicographically negative, and hence the dependence is violated.

Definition 3.5 *For each $1 \leq k \leq n$, the constraint C_k is defined as*

$$\mathtt{J}'_1 = \mathtt{J}_1 \ \ \& \ \cdots \ \& \ \ \mathtt{J}'_{k-1} = \mathtt{J}_{k-1} \ \ \& \ \ \mathtt{J}'_k < \mathtt{J}_k$$

For each $1 \leq m \leq n$, and each $1 \leq k \leq n$, let $\mathcal{Q}^k_m$ denote the polytope obtained from $\mathcal{P}^*_m$ by adding the constraint C_k. We have the following proposition.

Proposition 3.6 *The application of a unimodular matrix $\mathcal{U}$ is valid if and only if for each $1 \leq m \leq n$, and each $1 \leq k \leq n$, the polytope $\mathcal{Q}^k_m$ does not contain integer points.*

Corollary 3.7 *The application of a unimodular matrix $\mathcal{U}$ is valid if for each $1 \leq m \leq n$, and each $1 \leq k \leq n$, the polytope $\mathcal{Q}^k_m$ is empty.*

The difference between Proposition 3.6 and Corollary 3.7 is that the condition given in the corollary can be checked using Fourier-Motzkin elimination. This yields a conservative approximation to the solution of the validity problem. We obtain an exact solution if we can check whether or not a non-empty polytope does not contain integer points. The omega-test proposed by Pugh [Pug92] is a well-known test based on Fourier-Motzkin elimination that can test for precisely this condition.

Example 2. (Continued) Given the transformed dependence polytope $\mathcal{P}^*_1$, it is easy to see that both the conditions

$$\mathtt{J}'_1 < \mathtt{J}_1 \ \text{ and } \ \mathtt{J}'_1 = \mathtt{J}_1 \ \& \ \mathtt{J}'_2 < \mathtt{J}_2$$

deliver empty polytopes. Hence the transformation $\mathcal{U}$ does not violate the flow dependence from S_1 to S_2 in the program of Example 2. □

4 Extensions

In this section we show how to extend the theory developed in the previous sections to the cases where the body of the loop contains affine guards, where the two statements involved in the dependence are to be transformed by different unimodular transformations, and where displacements are used. This shows that the techniques presented in the previous sections are very flexible. Once the basic techniques have been developed, these extensions can be incorporated almost without effort.

4.1 Affine guards

In this section we show how to deal with a certain kind of `IF`-statements guarding the statements involved in a dependence. We call these `IF`-statements *affine guards*. These affine guards are, for instance, generated when converting a non-perfectly nested loop into a perfectly nested one, as described in [Kni94]. If one wants to transform the resulting perfectly nested loop by a unimodular transfornmation, it is important to be able to deal with affine guards.

Definition 4.1 *Let $\mathcal{L}$ be a loop nest.*

1. *A condition of the form $e \leq e'$ where e and e' are affine expressions in the loop variables, is called a* simple affine condition *for $\mathcal{L}$.*
2. *The conjunction of one or more simple affine conditions for $\mathcal{L}$ is called an* affine condition *for $\mathcal{L}$.*
3. *An `IF`-statement of which the condition is an affine condition is called an* affine guard *for $\mathcal{L}$.*

Suppose we have a loop nest containing affine guards.

```
DO  I_1 = L_1, U_1
    ...
    DO  I_n = L_n, U_n
            IF (f_1 ≤ f'_1 & ... & f_k ≤ f'_k)  S_1
            IF (g_1 ≤ g'_1 & ... & g_l ≤ g'_l)  S_2
    ENDDO
    ...
ENDDO
```

Then the affine condition $(f_1 \leq f'_1 \ \& \cdots \& \ f_k \leq f'_k)$ determines a subspace $\mathcal{Q}_1$ of the iteration space of this loop. Likewise, the other affine condition determines a subspace $\mathcal{Q}_2$. Since the conditions are affine, these subspaces are convex subspaces. When we construct the dependence polytopes $\mathcal{P}_1, \ldots, \mathcal{P}_n$ as described in section 3.1, we want to restrict these to those points in $\mathcal{Q}_1 \times \mathcal{Q}_2$. That is, the dependence polytopes $\mathcal{P}'_m$ for $1 \leq m \leq n$ are given by

$$\mathcal{P}'_m = \mathcal{P}_m \cap (\mathcal{Q}_1 \times \mathcal{Q}_2)$$

where $\mathcal{P}_m$ is the dependence polytope defined in section 3.1. We can obtain these polytopes by solving the following system of inequalities using Fourier-Motzkin elimination.

$$\begin{cases} \mathtt{L}_1 \leq \mathtt{I}_1 \leq \mathtt{U}_1, \ldots, \mathtt{L}_n \leq \mathtt{I}_n \leq \mathtt{U}_n \\ \mathtt{L}_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq \mathtt{I}'_1 \leq \mathtt{U}_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}], \ldots, \\ \qquad \mathtt{L}_n[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq \mathtt{I}'_n \leq \mathtt{U}_n[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \\ f_1 \leq f'_1, \ldots, f_k \leq f'_k \\ g_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq g'_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}], \ldots, g_l[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \leq g'_l[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \\ e_1 = e'_1[\vec{\mathtt{I}}'/\vec{\mathtt{I}}], \ldots, e_d = e'_d[\vec{\mathtt{I}}'/\vec{\mathtt{I}}] \\ \mathtt{I}_1 = \mathtt{I}'_1, \ldots, \mathtt{I}_{m-1} = \mathtt{I}'_{m-1}, \mathtt{I}'_m < \mathtt{I}_m \end{cases}$$

Analogously to section 3.3 we define the polytopes $\mathcal{Q}^k_m$ using the matrix $\mathcal{U}^*$ and the constraints C_k. We arrive at the following proposition.

Proposition 4.2 *Given a perfectly nested loop $\mathcal{L}$ containing affine guards and a unimodular transformation $\mathcal{U}$. Then the application of $\mathcal{U}$ is valid if and only if for each dependence present in the loop, the associated polytopes $\mathcal{Q}^k_m$ do not conatain integer points.*

4.2 Multiple transformations

Recently it has proposed to allow different transformations for different statements in the loop body [KTA94, KPR94, KP94, KS94]. In this section we show how the theory developed in section 3 can be extended to cover multiple unimodular transformations as well.

Suppose we have a loop nest containing statements S_1 and S_2 and that we want to transform S_1 using a unimodular transformation $\mathcal{U}_1$, and S_2 using $\mathcal{U}_2$. The reader is referred to [KTA94] for a discussion about how to interpret such transformations. The main point is that each iteration $\langle \mathtt{I}_1, \ldots, \mathtt{I}_n \rangle$, is mapped to $\mathcal{U}_1\langle \mathtt{I}_1, \ldots, \mathtt{I}_n \rangle$ for S_1, and to $\mathcal{U}_2\langle \mathtt{I}_1, \ldots, \mathtt{I}_n \rangle$ for S_2.

Now suppose there exists a dependence between S_1 and S_2, from iteration $\langle \mathtt{I}_1, \ldots, \mathtt{I}_n \rangle$ to $\langle \mathtt{I}'_1, \ldots, \mathtt{I}'_n \rangle$. Obviously the transformation is valid if for each such dependence it is the case that

$$\mathcal{U}_2\langle \mathtt{I}'_1, \ldots, \mathtt{I}'_n \rangle \ - \ \mathcal{U}_1\langle \mathtt{I}_1, \ldots, \mathtt{I}_n \rangle$$

is lexicographically positive.

We can test for this last condition as follows. Recall that in section 3 the transformed dependence polytopes were computed using $\mathcal{U}^*$. Now define the following operation on matrices.

$$\mathcal{U}_1 \otimes \mathcal{U}_2 = \begin{pmatrix} \mathcal{U}_1 & 0 \\ 0 & \mathcal{U}_2 \end{pmatrix}$$

Lemma 4.3 *If $\mathcal{U}_1$ and $\mathcal{U}_2$ are unimodular, then so is $\mathcal{U}_1 \otimes \mathcal{U}_2$.*

We can again compute the polytopes $\mathcal{Q}^k_m$, completely analogous to section 3.3. We arrive at the following proposition.

Proposition 4.4 *Let $\mathcal{L}$ be a loop nest containing N statements $S_1, \ldots, S_N$. Let $\mathcal{U}_1, \ldots, \mathcal{U}_N$ be unimodular. Then the transformation which transforms statement S_i using $\mathcal{U}_i$ is valid if and only if for each dependence from S_i to S_j it is the case that the polytopes $\mathcal{Q}^k_m$ obtained as described above from $\mathcal{U}_i \otimes \mathcal{U}_j$ do not contain integer points.*

4.3 Displacements

In this section we show how to add displacements to the techniques discussed in the previous sections. Displacements were introduced in [AT93, TALV93]. The idea behind displacements is to add a vector to the iterations in the transformed iteration space[1]. We allow different displacements for different statements in a loop. This means that for statement S_i iteration $\vec{\mathtt{I}}$ is mapped as

$$\vec{\mathtt{I}} \mapsto \mathcal{U}\vec{\mathtt{I}} + \vec{\mathtt{d}}_i$$

We now show how to deal with displacements in the present framework.

Suppose we are examining a dependence δ from S_i to S_j. First, we compute the dependence polytopes $\mathcal{P}_m$ as in section 3.1. Second, we have to transform these polytopes by the unimodular transformation $\mathcal{U}$ and the dispacements $\vec{\mathtt{d}}_i$ and $\vec{\mathtt{d}}_j$. This means that the transformed dependence polytopes $\mathcal{P}^*_m$ are obtained by

[1] In [AT93, TALV93] a different but equivalent formulation is used.

transforming $\mathcal{P}_m$ using the unimodular transformation $\mathcal{U}^*$ and displacement

$$\vec{\mathtt{d}} = \begin{pmatrix} \vec{\mathtt{d}}_1 \\ \vec{\mathtt{d}}_2 \end{pmatrix}$$

in the way described in [AT93, TALV93].

We briefly review how to transform a polytope $\mathcal{P}$ using a unimodular transformation $\mathcal{U}$ and a displacement $\vec{\mathtt{d}}$ yielding a transformed polytope $\mathcal{P}^*$. For more background and examples, consult [AT93, TALV93]. Suppose that $\mathcal{P}$ is defined using a system of inequalities involving the indices $\vec{\mathtt{I}}$. Like in section 3 we denote the indices for the system of inequalities defining the polytope $\mathcal{P}^*$ by $\vec{\mathtt{J}}$. Then we have that $\vec{\mathtt{J}} = \mathcal{U}\vec{\mathtt{I}} + \vec{\mathtt{d}}$, or that

$$\vec{\mathtt{I}} = \mathcal{U}^{-1}\left(\vec{\mathtt{J}} - \vec{\mathtt{d}}\right) = \mathcal{U}^{-1}\vec{\mathtt{J}} - \mathcal{U}^{-1}\vec{\mathtt{d}}$$

Hence we have to substitute $\mathcal{U}^{-1}\vec{\mathtt{J}} - \mathcal{U}^{-1}\vec{\mathtt{d}}$ for $\vec{\mathtt{I}}$ in the system of inequalities defining $\mathcal{P}$ to obtain $\mathcal{P}^*$. We use Fourier-Motzkin elimination to obtain a standard form for the resulting system of inequalities. Observe that a point $\langle \vec{\mathtt{I}}, \vec{\mathtt{I}}' \rangle$ in $\mathcal{P}_m$ (which thus consists of the source and sink of a dependence) is mapped to the point $\langle \mathcal{U}\vec{\mathtt{I}} + \vec{\mathtt{d}}_i, \mathcal{U}\vec{\mathtt{I}}' + \vec{\mathtt{d}}_j \rangle$ in $\mathcal{P}^*_m$, as desired.

We can again compute the polytopes $\mathcal{Q}^k_m$, completely analogous to section 3.3. We arrive at the following proposition.

Proposition 4.5 *Let $\mathcal{L}$ be a loop nest containing N statements $S_1, \ldots, S_N$. Let $\mathcal{U}$ be unimodular and let $\vec{\mathtt{d}}_1, \ldots, \vec{\mathtt{d}}_N$ be displacement vectors. Then the transformation which transforms statement S_i using $\mathcal{U}$ and displacement $\vec{\mathtt{d}}_i$ is valid if and only if for each dependence from S_i to S_j it is the case that the polytopes $\mathcal{Q}^k_m$ obtained as described above using displacements $\vec{\mathtt{d}}_i$ and $\vec{\mathtt{d}}_j$ do not contain integer points.*

Please observe that we can merge the theory described above to the case where we want to use different transformations to different statements which may be affine guards, together with displacements.

5 Discussion

In this paper we have proposed a new method for obtaining a compact representation of dependences. In this approach, the bounds of the loop to be analysed should be admissible, and the index functions of arrays should be affine expressions. The main advantage of this representation is that it can be used to test the validity of a unimodular transformation exactly. We have shown how the theory can be extended to cover situations, where affine guards are present in the loop or when more than one transformation will be applied to the loop, as well.

First, we have some comments on the space needed to store the proposed representaion of the dependences in a loop. For each dependence we need to store a number of dependence polytopes. Each such polytope can be represented by $4n$ bounds where n is the dimension of the iteration space. There are n such polytopes, so the total of space needed for the proposed representation is $\mathcal{O}(n^2)$ where n is the nesting depth of the loop. Since this nesting depth tends to be small in practical situations, this seems to be an acceptable amount of space. Next, we need $\mathcal{O}(n^2)$ applications of the Fourier-Motzkin elimination algorithm. Although this algorithm is exponential, it has been observed before [Pug92] that it is efficient for practical purposes. Our own experience [BW94] confirms this and shows that the test can be executed in a few seonds in practical situations.

A well-known extension to unimodular transformations are transformations specified by non-singular integer matrices [LP92]. The theory of this paper can immediately be extended to this case. The only point is that the iteration space of the transformed loop may contain holes: the resulting loop has non-unit strides. Hence the statement of Proposition 3.6 does not hold if we would allow these kind of matrices. In case the polytopes $\mathcal{Q}^k_m$ only contain integer points that do not correspond to actual iteration points, then the transformation is valid as well. But we can formulate the following conservative test.

Corollary 5.1 *The application of a non-singular integer matrix $\mathcal{U}$ is valid if for each $1 \leq m \leq n$, and each $1 \leq k \leq n$, the polytope $\mathcal{Q}^k_m$ does not contain integer points (or, is empty, respectively).*

In order to make a comparison with other methods for representing dependences and validating a transformation, consider the following loop.

```
DO  I1 = 1,10
    DO  I2 = 1,10
        S1: A(I1 + I2) = ···
        S2: ··· = ··· A(I1 + I2 − 1) ···
    ENDDO
ENDDO
```

Consider the flow dependence from S_1 to S_2. The dependence distances of this dependence are given by $\langle \alpha, -\alpha + 1 \rangle$ as inspection of the iteration space shows.

The direction vector of this dependence is $\langle \leq, * \rangle$. The interval vector is $\langle [0,\infty], [-\infty, 1] \rangle$. The dependence polytope $\mathcal{P}_1$ is given by

$$\begin{cases} 1 \leq & \mathtt{I}_1 & \leq 9 \\ 1 \leq & \mathtt{I}_2 & \leq 10 \\ \mathtt{I}_1 + 1 \leq & \mathtt{I}'_1 & \leq \min(\mathtt{I}_1 + \mathtt{I}_2, 10) \\ 1 + \mathtt{I}_1 + \mathtt{I}_2 - \mathtt{I}'_1 \leq & \mathtt{I}'_2 & \leq 1 + \mathtt{I}_1 + \mathtt{I}_2 - \mathtt{I}'_1 \end{cases}$$

and the dependence polytope $\mathcal{P}_2$ is given by

$$\begin{cases} 1 \leq & \mathtt{I}_1 & \leq 10 \\ 1 \leq & \mathtt{I}_2 & \leq 9 \\ \mathtt{I}_1 \leq & \mathtt{I}'_1 & \leq \mathtt{I}_1 \\ \mathtt{I}_2 + 1 \leq & \mathtt{I}'_2 & \leq \mathtt{I}_2 + 1 \end{cases}$$

Now we want to apply the unimodular transformation

$$\mathcal{U} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

If we test whether this transformation violates the flow dependence, we compute:

$$\mathcal{U} \begin{pmatrix} \leq \\ * \end{pmatrix} = \begin{pmatrix} * \\ * \end{pmatrix}$$

$$\mathcal{U} \begin{pmatrix} [0,\infty] \\ [-\infty, 1] \end{pmatrix} = \begin{pmatrix} [-\infty,\infty] \\ [-\infty, 1] \end{pmatrix}$$

Hence both direction and interval representation decide that this flow dependence is violated by $\mathcal{U}$.

However, computing $\mathcal{P}_1^*$ and $\mathcal{P}_2^*$ yield that $\mathcal{P}_1^*$ is given by

$$\begin{cases} 2 \leq & \mathtt{J}_1 & \leq 19 \\ \max(\mathtt{J}_1 - 9, 1) \leq & \mathtt{J}_2 & \leq \min(\mathtt{J}_1 - 1, 10) \\ \mathtt{J}_1 + 1 \leq & \mathtt{J}'_1 & \leq \mathtt{J}_1 + 1 \\ \max(\mathtt{J}'_1 - \mathtt{J}_1, \mathtt{J}'_1 - 10) \leq & \mathtt{J}'_2 & \leq \mathtt{J}'_1 + \mathtt{J}_2 - \mathtt{J}_1 - 1 \end{cases}$$

and that $\mathcal{P}_2^*$ is given by

$$\begin{cases} 1 \leq & \mathtt{J}_1 & \leq 19 \\ \max(\mathtt{J}_1 - 10, 1) \leq & \mathtt{J}_2 & \leq \min(\mathtt{J}_1 - 1, 9) \\ \mathtt{J}_1 + 1 \leq & \mathtt{J}'_1 & \leq \mathtt{J}_1 + 1 \\ \mathtt{J}_2 + 1 \leq & \mathtt{J}'_2 & \leq \mathtt{J}_2 + 1 \end{cases}$$

It is easy to see that the conditions C_1 and C_2 both make both polytopes empty. Hence we conclude that the dependence is not violated by $\mathcal{U}$. This shows that our method is strictly more powerful than the two methods based on direction and interval vectors.

References

[AI91] C. Ancourt and F. Irigoin. Scanning polyhedra with DO loops. In *Proc. 3rd ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pages 39–50, 1991.

[AT93] E. Ayguadé and J. Torres. Partioning the statement per iteration space using non-singular matrices. In *Proc. 7th ACM Int. Conf. Supercomputing*, 1993.

[Ban88] U. Banerjee. *Dependence Analysis for Supercomputing.* Kluwer Academic Publishers, Norwell, 1988.

[Ban91] U. Banerjee. Unimodular transformations of double loops. In *Advances in Languages and Compilers for Parallel Processing*, chapter 10. The MIT Press, 1991.

[Ban93] U. Banerjee. *Loop Transformations for Restructuring Compilers.* Kluwer Academic Publishers, Norwell, 1993.

[BW94] A.J.C. Bik and H.A.G. Wijshoff. Implementation of Fourier-Motzkin elimination. Technical Report 94-42, Dept. of Computer Science, Leiden University, 1994.

[DE73] G.B. Dantzig and B.C. Eaves. Fourier-Motzkin elimination and its dual. *J. of Combinatorial Theory*, 14:288–297, 1973.

[Fea91] P. Feautrier. Dataflow analysis of array and scalar references. *Int. J. of Parallel Programming*, 20(1):23–53, 1991.

[Kni94] P.M.W. Knijnenburg. Towards unimodular transformations of non-perfectly nested loops. Technical Report 94-41, Dept. of Computer Science, Leiden University, 1994.

[KP94] W. Kelly and W. Pugh. Finding legal reordering transformations using mappings. In *Proc. 7th Ann. Workshop on Languages and Compilers for Parallel Computing*, 1994.

[KPR94] W. Kelly, W. Pugh, and E. Rosser. Code generation for multiple mappings. Technical Report UMIACS-TR-94-87, Dept. of Computer Science, Univ. of Maryland, 1994.

[KS94] D. Kulkarni and M. Stumm. Computational alignment: A new, unified program transformation for local and global optimization.

Technical Report CSRI-292, CSRI, University of Toronto, 1994.

[KTA94] P.M.W. Knijnenburg, J. Torres, and E. Ayguadé. Multi-transformations for doubly nested loops. Technical Report 94/13, DAC/UPC, 1994.

[LP92] W. Li and K. Pingali. A singular loop transformation framework based on non-singular matrices. In *Proc. 5th Workshop on Language and Compilers for Parallel Computers*, 1992.

[Pol88] C. Polychronolpoulos. *Parallel Programming and Compilers.* Kluwer Academic Publishers, Boston, 1988.

[Pug92] W. Pugh. The Omega test: A fast and practical integer programming algorithm for dependence analysis. *Comm. of the ACM*, 8:102–114, 1992.

[PW86] D.A. Padua and M.J. Wolfe. Advanced compiler optimizations for supercomputers. *Comm. of the ACM*, 29(12):1184–1201, 1986.

[PW92] W. Pugh and D. Wonnacott. Going beyond integer programming with the Omega test to eliminate false data dependences. Technical Report CS-TR-2993, Dept. of Computer Science, Univ. of Maryland, 1992.

[TALV93] J. Torres, E. Ayguadé, J. Labarta, and M. Valero. Align and distribute-based linear loop transformations. In U. Banerjee, D. Gelernter, A. Nicolau, and D. Padua, editors, *Proc. 6th Int. Workshop on Languages and Compilers for Parallel Computing*, volume 768 of *Lecture Notes in Computer Science*, pages 321–339, Berlin, 1993. Springer Verlag.

[WL91] M.E. Wolf and M.S. Lam. A loop transformation theory and an algorithm to maximize parallelism. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):430–439, 1991.

[Wol91] M. Wolfe. *Optimizing Supercompilers for Supercomputers.* The MIT Press, 1991.

[WT90] M. Wolfe and C.-W. Tseng. The Powertest for data dependence. Technical Report CS/E 90-015, Oregon Graduate Institute of Science and Technology, 1990.

[YAI94] Y.-Q. Yang, C. Ancourt, and F. Irigoin. Minimal data dependence abstractions for loop transformations. In *Proc. 7th Ann. Workshop on Languages and Compilers for Parallel Computing*, 1994.

[ZC90] H. Zima and B. Chapman. *Supercompilers for Parallel and Vector Computers.* ACM Press, New York, 1990.