Parallelisation with a Grid Abstraction

Sven DE SMET [a,1]
[a] *Student at Ghent University*

Abstract. This paper describes a new technique for automatic parallelisation in the $\mathcal{Z}$-polyhedral model. The presented technique is applicable to arbitrarily nested loopnests with iteration spaces that can be represented as unions of $\mathcal{Z}$-polyhedra and affine modular data-access functions. The technique partitions both iteration and data spaces of the computation. The maximal amount of parallelism that can be represented using grid partitions is extracted.

Keywords. Parallelisation, Z-polyhedral model, Lattice, Grid

Introduction

In order to optimally benefit from the incessant increase in parallel computational capacity in modern architectures, the available parallelism in our algorithms must increase accordingly. Since manual parallelisation requires considerable effort from the developer, we must resort to automatic parallelisation.

In recent decades, much research in the field of automatic parallelisation has focused on structured computations that can be represented in the polyhedral model. This model allows to represent and transform an important class of computationally intensive algorithms and many parallelisation techniques have been developed for it. The Farkas algorithm allows to extract the maximal amount of parallelism that can be represented with multi-dimensional affine scheduling functions [1,2,3]. By extracting space partitions before every time partition during the recursive construction of the affine partitioning functions, the coarseness of the available parallelism can be detected [4,5]. This results in a specification that is well-suited for current hierarchical, parallel architectures.

It is well known that some computations contain more parallelism than can be extracted with affine spacetime mappings [6], and several techniques have been proposed that aim to address this issue [7,8,9,10,11].

In this paper a new technique is presented that is applicable to computations with general $\mathcal{Z}$-polyhedral iteration spaces and affine modular data access functions. This technique extends Lim and Lam's affine spacetime partitioning technique [4,5] by enabling the extraction of more parallelism through the use of affine modular partitioning functions. The technique also extends the unimodular transformation based approach by Yu and D'Hollander [9] by making it applicable to general loopnests. Since the Farkas algorithm allows to compute a schedule in polynomial time, a complete multi-dimensional spacetime mapping that extracts maximal parallelism for the grid abstraction can be computed in polynomial time.

[1] sven.desmet@cubiccarrot.com, http://www.cubiccarrot.com/salmoc/

1. Mathematical Background

This section briefly reviews the mathematics required in the rest of the paper.

1.1. Notation

Let $a..b$ with $a, b \in \mathbb{Z}$ denote the set of integers from a up to and including b: $a..b \triangleq \{i | i \in \mathbb{Z} \wedge a \leq i \leq b\}$. For a vector or tuple x, let $|x|$ denote the number of elements it contains.

For a set A, let $\mathcal{P}A$ denote the set of all its subsets and $\mathcal{L}A \triangleq \mathcal{P}(A^2)$ denote the set of binary relations on A. The **Kleene Closure** of A is denoted with $A^* \triangleq \bigcup_i^{\mathbb{N}} A^i$.

For a set of vectors X, the set **finitely generated** by X through the coefficient set Y is the set of finite combinations of vectors in X with coefficients from Y, denoted with $\langle X \rangle_Y$:

$$\langle X \rangle_Y \triangleq \{ \sum_z^Z c(z) z | (Z \subseteq X) \wedge (\#Z \in \mathbb{N}) \wedge (c \in Z \to Y) \} \,, \tag{1}$$

where $\#Z$ denotes the cardinality of the set Z.

For a relation $\lessdot$ on $\mathbb{A}$ and a set $\mathbb{B} \subseteq \mathbb{A}$, let $\mathcal{U}_\lessdot^\mathbb{A}\mathbb{B}$ denote the **earliest $\lessdot$-successor** of $\mathbb{B}$ on $\mathbb{A}$, which is defined by the axiom

$$\forall_\mathbb{B}^{\mathcal{P}\mathbb{A}} \mathcal{U}_\lessdot^\mathbb{A}\mathbb{B} \in \mathbb{A} \wedge \forall_b^\mathbb{B} b \lessdot \mathcal{U}_\lessdot^\mathbb{A}\mathbb{B} \wedge \forall_a^\mathbb{A} (\forall_b^\mathbb{B} b \lessdot a) \Rightarrow \mathcal{U}_\lessdot^\mathbb{A}\mathbb{B} \lessdot a \tag{2}$$

1.2. Basic Definitions

An **affine modular function** is a function of the form $\lambda x.(c^T x + b) \bmod d$ on $\mathbb{Z}^n$ with $c \in \mathbb{Z}^n$ and $b, d \in \mathbb{Z}$ where $\bmod$ denotes the modulo operation. We use Boute's E-definition of the modulo operation based on Euclid's theorem [12], i.e.

$$a \bmod b = c \quad \equiv \quad c \in 0..(|b| - 1) \wedge \exists_k^\mathbb{Z} a - c = kb \quad . \tag{3}$$

Let $\mathbb{F}_n$ denote the set of affine modular functions on $\mathbb{Z}^n$.

For a set F of functions, let $\mathcal{E} : F^* \to \mathcal{P}F$ be a function that maps any m-dimensional function $f \in F^m$ to a set containing its single-dimensional constituent functions $\mathcal{E}f \triangleq \{f_i | i \in 1..m\}$.

Let $\pi_S(L)$ denote the projection of a set $L \subseteq \mathbb{Z}^m$ onto the dimensions in the set $S \subseteq 1..m$. For a set $L \subseteq A \times B$ we also let $\pi_B(L)$ denote the projection of L onto the respective subspace B of the product.

Let $\mathrm{Im}\, f$ further denote the **image** of a function f and $\mathrm{Ker}\, f$ denote its **kernel**. Let us further define the function $\mathcal{Z} : \mathcal{P}\mathbb{Z}^n \to \mathcal{P}\mathbb{F}_n$ that maps a subset S of $\mathbb{Z}^n$ to the set of affine modular functions that contain S in their kernel:

$$\forall_S^{\mathcal{P}\mathbb{Z}^n} \mathcal{Z}(S) \triangleq \{f | f \in \mathbb{F}_n \wedge S \subseteq \mathrm{Ker}\, f\} \tag{4}$$

1.3. Partitions

1.3.1. Basic definitions

A **partition** $\mathbf{B}$ of a set $\mathbb{A}$ is a set of subsets of $\mathbb{A}$ such that their disjoint union is equal to $\mathbb{A}$, i.e. $\mathbb{A} \triangleq \bigsqcup_{\mathbb{B}}^{\mathbf{B}} \mathbb{B}$. The **cells** of a partition are the subsets of which it consists. Let $\mathcal{C}\mathbb{A}$ denote the set of all partitions of $\mathbb{A}$.

A partition $\mathbf{P}$ is **at least as fine as** a partition $\mathbf{Q}$, denoted as $\mathbf{P} \preceq \mathbf{Q}$ (or, $\mathbf{Q}$ is **at least as coarse as** $\mathbf{P}$, $\mathbf{Q} \succeq \mathbf{P}$), if every cell of $\mathbf{P}$ is contained in a cell of $\mathbf{Q}$:

$$\mathbf{P} \preceq \mathbf{Q} \triangleq \bigforall_{P}^{\mathbf{P}} \bigexists_{Q}^{\mathbf{Q}} P \subseteq Q \tag{5}$$

A partition $\mathbf{P}$ is **finer than** a partition $\mathbf{Q}$, denoted as $\mathbf{P} \prec \mathbf{Q}$ (or, $\mathbf{Q}$ is **coarser than** $\mathbf{P}$, $\mathbf{Q} \succ \mathbf{P}$) if $\mathbf{P} \preceq \mathbf{Q} \wedge \mathbf{P} \neq \mathbf{Q}$.

The **refining** operation is defined as $\otimes \triangleq \mathcal{U}_{\succeq}^{\mathcal{C}\mathbb{A}}$ and the **coarsening** operation is defined as $\odot \triangleq \mathcal{U}_{\preceq}^{\mathcal{C}\mathbb{A}}$. The **bottom** of $\mathcal{C}\mathbb{A}$ is $\bot_{\mathbb{A}} \triangleq \mathcal{U}_{\succeq}^{\mathcal{C}\mathbb{A}} \mathcal{C}\mathbb{A} = \{\{a\} | a \in \mathbb{A}\}$, the finest element, while the **top** of $\mathcal{C}\mathbb{A}$ is $\top_{\mathbb{A}} \triangleq \mathcal{U}_{\preceq}^{\mathcal{C}\mathbb{A}} \mathcal{C}\mathbb{A} = \{\mathbb{A}\}$, the coarsest element. For every set $\mathbb{A}$, the structure $(\mathcal{C}\mathbb{A}, \preceq, \otimes, \odot, \bot, \top)$ forms a **complete lattice**.

1.3.2. Constructions

Let $\mathbf{B}|_{\mathbb{A}} \triangleq \{B \cap \mathbb{A} | B \in \mathbf{B}\}$ denote the **restriction** of a partition $\mathbf{B} \in \mathcal{C}\mathbb{B}$ to a set $\mathbb{A} \subseteq \mathbb{B}$.

A **partition bijection** is a bijective relation between the cells of two partitions. A partition bijection on the sets C and D also specifies a partition of $C \cup D$ where each cell of the partition of $C \cup D$ is the union of a cell of the partition of C with its related cell of the partition of D.

A partition on a set $\mathbb{A}$ can be specified using a function $f \in \mathbb{A} \to \mathbb{B}$ to another set $\mathbb{B}$ by allocating all elements that are mapped to the same value by f to a cell unique to this value. Let us denote the resulting partition with $\mathbb{A}_{[f]}$,

$$\mathbb{A}_{[f]} \triangleq \{\{a | a \in \mathbb{A} \wedge f(a) = i\} | i \in \operatorname{Im} f\} \tag{6}$$

1.4. Affine grids, diophantine equations and modular functions

A **linear grid** $\mathcal{G}A \subseteq \mathbb{Z}^n$ is the set of all integral combinations of a finite set of integer vectors $A \subseteq \mathbb{Z}^n$, i.e. $\mathcal{G}A \triangleq \langle A \rangle_{\mathbb{Z}}$. The set A is a set of **generators** of $\mathcal{G}A$. For a matrix B we extend the notation to let $\mathcal{G}B$ denote the grid generated by the rows of B. Let $\mathbb{G}_n$ denote the set of linear grids on $\mathbb{Z}^n$.

An **affine grid** $\mathcal{G}_a A \subseteq \mathbb{Z}^n$ is the translation of a linear grid $\mathcal{G}A$ by an integer vector a, i.e. $\mathcal{G}_a A \triangleq \{a + b \mid b \in \mathcal{G}A\}$. The **homogeneous space** $\mathbb{Z}^{n+1} \triangleq \mathbb{Z} \times \mathbb{Z}^n$ extends $\mathbb{Z}^n$ with a homogeneous dimension h (we here choose its dimension to be positioned before the other dimensions). Let $\widetilde{V}$ denote the set that results from prepending the set V with a homogeneous dimension in this way. An affine grid in $\mathbb{Z}^n$ can be represented in the homogeneous space as the intersection of the **homogeneous plane** that contains the points of $\mathbb{Z}^n$ for which $h = 1$ with the linear grid $\mathcal{G}(\{(1) \bowtie b\} \cup \bigcup_{a}^{A} \{(0) \bowtie a\})$, where $\bowtie$ denotes the concatenation of two vectors. Let $\mathbb{G}_n^*$ denote the set of affine grids on $\mathbb{Z}^n$.

A **unimodular matrix** U is a square integer matrix for which $|\det U| = 1$. This implies U^{-1} is integral as well, and therefore that the transformation of multiplying an element of $S \subseteq \mathbb{Z}^n$ by a unimodular matrix gives a bijection between S and its image under the unimodular transformation.

Let $l(z)$ denote the index of the first non-zero element of a vector $z \in \mathbb{Z}^n$ (or $+\infty$ for the zero vector), then a matrix $A \in \mathbb{Z}^{m \times n}$ is in **echelon form** iff

$$\bigvee_{r}^{2..m} (A_{r,*} = 0) \vee (l(A_{r-1,*}) < l(A_{r,*})) \tag{7}$$

A matrix A is in **hermite normal form** (HNF) if it is in echelon form, does not contain zero rows and satisfies

$$\bigvee_{r}^{2..m} \bigvee_{q}^{1..(r-1)} A_{r,l(A_{r,*})} > A_{q,l(A_{r,*})} \geq 0 \tag{8}$$

The HNF of a matrix can be obtained by left-multiplying it with an appropriate unimodular matrix U and retaining the non-zero rows, i.e. $UA = \begin{bmatrix} \text{HNF}(A) \\ 0 \end{bmatrix}$. Note that $\mathcal{G}\text{HNF}(A) = \mathcal{G}A$ since every row of HNF(A) is an integral combination of rows of A and the integrality of U^{-1} ensures that every row of A is an integral combination of rows of HNF(A). The HNF can thus be used as a normal form representation for a grid. An efficient algorithm exists to compute the HNF and a unimodular matrix that transforms the input matrix into HNF [13].

If the generators of an affine grid are represented in homogeneous coordinates with the homogeneous coordinate in the first position, the HNF form provides an easy way to test whether the affine grid is empty. The affine grid contains an element only if the top-left element $A_{0,0}$ is equal to one[2]. The remaining matrix then provides the generators of the translated linear grid, with an additional column of zeroes for the homogeneous coordinate.

A **system of linear diophantine equations** is an equation of the form $Az = 0$ where A is a integral matrix and z is an unknown integral vector. The set of solutions is equal to the kernel of the multi-dimensional integral linear function $\lambda z.Az$ defined on $\mathbb{Z}^n$. Since every linear combination of elements in the kernel is contained in the kernel too, the kernel is a linear grid. A set of generators of this grid can be obtained by computing the HNF of A^T and an associated unimodular matrix U that transforms A^T into HNF(A^T), since then $UA^T = \begin{bmatrix} U_X \\ U_0 \end{bmatrix} A^T = \begin{bmatrix} \text{HNF}(A^T) \\ 0 \end{bmatrix}$ with $U = \begin{bmatrix} U_X \\ U_0 \end{bmatrix}$, i.e. $U_X A^T = \text{HNF}(A^T)$ and $U_0 A^T = 0$. The rows of U_0 then form a set of generators of the kernel of A. Indeed, any integral linear combination of rows lies in the kernel too due to linearity. Furthermore, since $U^{-1}U = I$ we have HNF$(U) = I$ and thus that $\mathcal{G}U = \mathcal{G}I$ so that any vector $z \in \mathbb{Z}^n$ can be written as $d + y$ with $d \in \mathcal{G}U_X$ and $y \in \mathcal{G}U_0$. Since $yA^T = 0$, z lies in the kernel only if $dA^T = 0$. Since $d \in \mathcal{G}U_X$, we have $d = w^T U_X$ for some $w \in \mathbb{Z}^{\text{rows}(U_X)}$ such that $dA^T = 0$ implies $dA^T = (w^T U_X)A^T = w^T \text{HNF}(A^T) = 0$. Since the rows of HNF$(A^T)$ are linearly independent, we see that $w = 0$ such that $d = w^T U_X = 0$ and $zA^T = 0 \Rightarrow z \in \mathcal{G}U_0$.

[2] It thus suffices to test whether the gcd of the coefficients of the first column of the original matrix is equal to one.

Let $\mathcal{U}_0(A)$ denote a matrix so that $\mathcal{U}_0(A)A = 0$ and let $\mathcal{U}_X(A)$ denote the matrix so that $\mathcal{U}_X(A)A = \text{HNF}(A)$ and so that $\begin{bmatrix}\mathcal{U}_X(A)\\ \mathcal{U}_0(A)\end{bmatrix}$ is unimodular.

A **system of affine diophantine equations** $Az + c = 0$ (with $c \in \mathbb{Z}^m$) can be solved by solving the system of linear diophantine equations $[c \mid A]\begin{bmatrix}h\\ z\end{bmatrix} = 0$. The result is a linear grid in the homogeneous space that represents the affine grid equal to the kernel of the **affine function** $\lambda z.Az + c$ defined on $\mathbb{Z}^n$.

A **system of modular equations** is an equation of the form $Az \bmod d = 0$ with $d \in \mathbb{Z}^m$. For each $i \in 1..m$ the equation $A_{i,*}z \bmod d_i = 0$ forms a separate modular equation. Since $a \bmod b = 0$ iff $\underset{k}{\overset{\mathbb{Z}}{\exists}} a = kb$, the system of modular equations can be solved by solving the system of linear diophantine equations $Az + \text{diag}(d)y = 0$, where $\text{diag}(d)$ denotes the diagonal matrix with elements from d, and projecting the result onto the first n dimensions. The generators of the projected linear grid can be obtained by retaining the first n dimensions of the generators of the grid. A **system of affine modular equations** can then be solved by solving the corresponding system of modular equations in the homogenised space.

Representing an affine grid as the integral combination of a set of integral vectors is equivalent to representing it as the solution of a system of affine modular equations. A constraining representation as a system of affine modular equations can be converted to its generating representation as described above, while converting the generating representation to a constraining representation can be done through matrix inversion if the matrix of generators is square [14].

Practically, we can use the same algorithm that allows to convert a matrix to its HNF and also produces the unimodular transformation that converts the matrix to HNF. Notice first that we can obtain a set of vectors that generate the linear functions that contain a linear grid $\mathcal{G}A$ in their kernels as the rows of $\mathcal{U}_0(A^T)$. If the rows of the matrix A are linearly independent, then $\mathcal{U}_X(A^T)A^T$ is a full-rank upper-triangular matrix. We can convert $\mathcal{U}_X(A^T)A^T$ to a diagonal matrix $\text{diag}(d)$ by applying unitary row transformations and the transformation of multiplying a row by a non-zero integer. Let D denote the matrix that captures this sequence of transformations that transform $\mathcal{U}_X(A^T)A^T$ to a diagonal form, i.e. $\text{diag}(d) = D(\mathcal{U}_X(A^T)A^T)$. We then have $((D\mathcal{U}_X(A^T))_{i,*}A^T)_j = d_i\delta_{i,j} = d_i(i = j)$. Every row of $D\mathcal{U}_X(A^T)$ thus contains a vector with coefficients of a linear function that contains all of the generators of $\mathcal{G}A$ in its kernel except for the generator that corresponds to the diagonal entry. This immediately allows to derive the modular functions that contain $\mathcal{G}A$ in their kernel since every element of $\mathcal{G}A$ will be contained in the kernel of $\lambda x.(D\mathcal{U}_X(A^T))_{i,*}x \bmod d_i$ for every row i. Since the linear parts of all obtained functions are linearly independent, we obtain all solutions. The functions for which $d_i = 1$ vanish everywhere and can therefore be discarded. This method can also be adapted for the affine version, by taking care that the affine supporting generator is included exactly once.

The **affine grid hull** of a set $S \subseteq \mathbb{Z}^n$ is denoted with $\mathcal{H}S \triangleq \mathcal{U}_{\subseteq}^{\mathbb{G}_n^*} S$. The affine grid hull of a union of affine grids can be computed by taking the union of the homogeneous generators of the affine grids.

1.5. Lemma on affine grids and modular functions

A basic problem when considering spacetime and data partitioning in the polyhedron model has the general form

$$\forall_{z}^{Q} f(z) = 0 \Rightarrow g(z) = 0 \tag{9}$$

where Q is a known set, f is a known function and $g \in \mathbb{F}_{n,m}$ is an unknown affine modular function. This is easily rewritten as

$$\forall_{z}^{Q \cap \operatorname{Ker} f} g(z) = 0 \tag{10}$$

which, by definition of $\mathcal{Z}$, is equivalent to $g \in \mathcal{Z}(Q \cap \operatorname{Ker} f)$. In this subsection we introduce some lemma which can be used to work with this type of problem.

The following lemma allows us to determine a subset of $\mathcal{Z}A$ if we know a superset of a set A:

Lemma 1.

$$\forall_{A,B}^{(\mathcal{P}\mathbb{Z}^n)^2} A \subseteq B \Rightarrow \mathcal{Z}A \supseteq \mathcal{Z}B \tag{11}$$

The following lemma allows us to replace the set P in $\mathcal{Z}P$ by $\mathcal{H}P$ and thus also identifies the maximal precision that is required to obtain optimal solutions:

Lemma 2.

$$\mathcal{Z} = \mathcal{Z} \circ \mathcal{H} \tag{12}$$

The following lemma closely resembles the Farkas lemma for polyhedra in that it allows to write an affine modular function that vanishes on an affine lattice as an integral combination of the affine modular constraints that define the affine lattice:

Lemma 3.

$$\forall_{h}^{\mathbb{F}_n} \mathit{Ker}\, h \neq \emptyset \Rightarrow \forall_{g}^{\mathbb{F}_{n,1}} \left[\begin{array}{l} \forall_{z}^{\mathbb{Z}^n} h(z) = 0 \Rightarrow g(z) = 0 \\ \equiv \\ g \in \langle \mathcal{E}h \rangle_{\mathbb{Z}} \end{array} \right. \tag{13}$$

Lemma 2 and lemma 3 imply that we can write every $g \in \mathcal{Z}(P)$ in an explicit form if we know any affine modular function vanishing precisely on the affine lattice hull of P. In particular, we can write this solution without redundant degrees of freedom by deriving a basis for the corresponding solution space. Notice that when taking an integral sum of affine modular functions, the modular coordinates of the summed components are independent and are mapped to separate coordinates in the result. As a result of Bézout's

lemma, these coordinates can be replaced by a single coordinate with the gcd of the coefficients of the original modular coordinates as coefficient.

The following convenient lemma resemble De Morgan's laws. This lemma allows to convert between a union in the generating representation and an intersection in the constraining representation:

Lemma 4.

$$\underset{T}{\overset{\mathcal{P}\mathcal{P}\mathbb{Z}^n}{\forall}} \mathcal{Z}(\bigcup_S^T S) = \bigcap_S^T \mathcal{Z}(S) \tag{14}$$

This lemma allows to convert between an intersection in the generating representation and an integral sum in the constraining representation:

Lemma 5.

$$\underset{T}{\overset{\mathcal{P}\mathbb{G}_n^*}{\forall}} (\bigcap_S^T S \neq \emptyset) \Rightarrow (\mathcal{Z}(\bigcap_S^T S) = \sum_S^T \mathcal{Z}(S)) \tag{15}$$

1.6. Grid partitions and grid partition bijections

A linear grid $\mathcal{G}A$ is a **subgrid** of a linear grid $\mathcal{G}B$ iff $\mathcal{G}A \subseteq \mathcal{G}B$. A linear subgrid can be used to specify a **grid partition** of the containing **supergrid** by considering all translations of $\mathcal{G}A$ in $\mathcal{G}B$ and allocating two elements of the grid to the same cell iff they are contained in the same translation of $\mathcal{G}A$. Every affine grid contained in $\mathcal{G}B$ that is a translation of $\mathcal{G}A$ then forms a cell of the partition. A linear grid $\mathcal{G}A$ can thus also be used to specify a partition of an affine grid $\mathcal{G}_b B$ if $\mathcal{G}A \subseteq \mathcal{G}B$ by partitioning $\mathcal{G}B$.

Since the cell grid of the partition that results from coarsening two grid partitions defined by the cell grids $\mathcal{G}A$ and $\mathcal{G}B$ is the smallest cell grid that contains both original cell grids, this cell grid is given by $\mathcal{H}(\mathcal{G}A \cup \mathcal{G}B) = \mathcal{G}(A \cup B)$. The cell grid of the partition that results from refining two grid partitions of two grid partitions defined by the cell grids $\mathcal{G}A$ and $\mathcal{G}B$ is $\mathcal{G}A \cap \mathcal{G}B$.

A non-empty affine grid $\mathcal{G}_a A \subseteq C \times D$ with $C \triangleq \mathbb{Z}^p$ and $D \triangleq \mathbb{Z}^q$ implicitly specifies a **grid partition bijection** on two grid partitions $\mathbf{P}$ of $\mathcal{G}_{\pi_C(a)}\pi_C(A)$ and $\mathbf{Q}$ of $\mathcal{G}_{\pi_D(a)}\pi_D(A)$. The subsets of homogeneous generators of $\mathcal{G}_a A$ with coordinates that are non-zero only for either the dimensions of C or D give the generators of the cell grid of the associated partition. Let us denote the generators of the cell grid of $\mathbf{P}$ as $\mathcal{T}_C(\mathcal{G}_a A)$, i.e.

$$\mathcal{T}_C(\mathcal{G}_a A) \triangleq \text{rows}(\mathcal{U}_0(\pi_{\widetilde{D}}(\begin{bmatrix} 1 & a^T \\ 0 & A \end{bmatrix}))\pi_C(A)) \tag{16}$$

The generators of the cell grid of $\mathbf{Q}$ can be obtained in the same (but mirrored) way. The same technique can also be applied when an affine grid $\mathcal{G}_a A \subseteq \mathbb{Z}^p \times \mathbb{Z}^q \times \ldots \times \mathbb{Z}^r$ specifies a multi-tuple grid partition bijection. The cell of the partition of the grid in C_q for a grid $\mathcal{G}_a A \subseteq \overset{k}{\underset{i=1}{\times}} C_i$ that specifies a multi-tuple grid partition bijection on this product space is then given by

$$\mathcal{T}_C(\mathcal{G}_a A) \triangleq \bigcup_{i=1, i\neq q}^{k} \text{rows}(\mathcal{U}_0(\pi_{\widetilde{C_k}}(\begin{bmatrix} 1 & a^T \\ 0 & A \end{bmatrix}))\pi_{C_q}(A)) \tag{17}$$

Notice that, if we want to determine the cell grids for each space in the product space, it suffices to apply the HNF algorithm once for each of the spaces.

Coarsening grid partition bijections can be done by taking the union of the generators of the cell grids.

2. Parallelisation with a Grid Abstraction

2.1. Representation of Computations

A **computation** is a structure $(\Omega, \Gamma, \Delta, \rightsquigarrow)$ where

- Ω is the set of **operations** executed during the computation.
- Δ is the **data space**, the set of data elements accessed during the computation. Information can be stored in a data element and can later be retrieved from it. The data space is specified implicitly by Ω and Γ.
- $\Gamma \subseteq \Omega \to \Delta$ is the set of **access functions** that map an operation to a data element accessed by it. (In the full version of this paper, partial functions will be accounted for.)
- $\rightsquigarrow \in \mathcal{L}\Omega$ is a minimal partial order of the operations such that any execution of the computation must respect $\rightsquigarrow$ to obtain valid results

For the static specification and analysis of computations, Ω and Δ are often given as the finite union of sets of a specific type. In this case, the finite union of operations is indexed by $\mathcal{S}$, the set of **statements** , $\Omega \triangleq \bigcup_{s}^{\mathcal{S}} \Omega_s$, while the finite union of data elements is indexed by $\mathcal{V}$, the set of **variables**, $\Delta \triangleq \bigcup_{v}^{\mathcal{V}} \Delta_v$. Let $\Gamma_s^v \subseteq \Omega_s \to \Delta_v$ denote the access functions from $s \in \mathcal{S}$ to $v \in \mathcal{V}$.

2.2. Spacetime Partitioning

2.2.1. Space Partitioning

A partition $\mathbf{Q} \in \mathcal{C}(\Omega \cup \Delta)$ such that every operation is contained in the same cell as all data elements that are accessed by it,

$$\forall_{f}^{\Gamma} \forall_{\omega,\delta}^{\Omega\times\Delta} \delta = f(\omega) \Rightarrow \exists_{Q}^{\mathbf{Q}} \{\omega, \delta\} \subseteq Q \tag{18}$$

allows to describe communication-free parallelism.

A partition of $\Omega \cup \Delta$ can be specified as a partition bijection between Ω and Δ. If we specify this partition bijection with a pair of functions $(\Phi, \Lambda) \in (\Omega \to \mathbb{A}) \times (\Delta \to \mathbb{A})$ such that the related partitions are $\Omega_{[\Phi]}$ and $\Delta_{[\Lambda]}$ and such that the elements in the cells that are related by the bijection are mapped to the same value of the set $\mathbb{A}$, then the partitioning functions must satisfy

$$\bigvee_{f}^{\Gamma} \bigvee_{\omega,\delta}^{\Omega\times\Delta} \delta = f(\omega) \Rightarrow \Lambda(\delta) = \Phi(\omega) \tag{19}$$

If we consider single dimensional affine modular functions then we can interpret δ and ω as the projections of a vector $(\delta, \omega) = m \in \Delta \times \Omega$ and rewrite the expression as

$$\bigvee_{f}^{\Gamma} \bigvee_{\omega,\delta}^{\Omega\times\Delta} (\mathbf{1}_\Delta - f)(m) = 0 \Rightarrow (\Lambda - \Phi)(m) = 0 \tag{20}$$

where $\mathbf{1}_X$ is the identity function on X. We thus see that this expression has the same form as the expression discussed in section 1.5 and can rewrite it as

$$(\Lambda - \Phi) \in \bigcap_{f}^{\Gamma} \mathcal{Z}(\Delta \times \Omega \cap \text{Ker}\,(\mathbf{1}_\Delta - f)) \tag{21}$$

By lemma 4, we can further rewrite the expression as

$$(\Lambda - \Phi) \in \mathcal{Z}(\bigcup_{f}^{\Gamma}(\Delta \times \Omega \cap \text{Ker}\,(\mathbf{1}_\Delta - f))) \tag{22}$$

Let us define the argument to $\mathcal{Z}$ as Ψ:

$$\Psi \triangleq \bigcup_{f}^{\Gamma}(\Delta \times \Omega \cap \text{Ker}\,(\mathbf{1}_\Delta - f)) \tag{23}$$

We will assume for convenience that the sets Δ and Ω are full-dimensional so that the term $\Delta \times \Omega$ can be discarded.

Since lemma 2 allows to compute the solution space by first determining $\mathcal{H}\Psi$, we can efficiently compute the argument to $\mathcal{Z}$ by taking the union of the generators of $\text{Ker}\,(\mathbf{1}_\Delta - f)$ over the access function $f \in \Gamma$.

To detect maximal parallelism, we must find the finest partition that satisfies equation (18)

If Δ and Ω consist of finite unions of sets indexed by $\mathcal{V}$ and $\mathcal{S}$, then we first compute a set $\Psi_{v,s}$ for every pair $(v, s) \in \mathcal{V} \times \mathcal{S}$,

$$\bigvee_{v,s}^{\mathcal{V}\times\mathcal{S}} \Psi_{v,s} \triangleq \bigcup_{f}^{\Gamma_s^v}(\Delta_v \times \Omega_s \cap \text{Ker}\,(\mathbf{1}_{\Delta_v} - f)) \tag{24}$$

Since the space partitions induced by a single variable $v \in \mathcal{V}$ only interact through the coarsening of the partition of Δ_v, we proceed by first determining the se of generators Υ_v of the cell lattice of the partition of Δ_v that results from combining the relations of space and data partitions $\Psi_{v,s}$ for all statements $s \in \mathcal{S}$:

$$\bigvee_{v}^{\mathcal{V}} \Upsilon_v \triangleq \bigcup_{s}^{\mathcal{S}} \mathcal{T}_{\Delta_v}(\Psi_{v,s}) \tag{25}$$

and subsequently add these generators to the initial lattice partition relations to obtain lattice partition relations that take all references to a specific variable into account:

$$\underset{(v,s)}{\overset{\mathcal{V}\times\mathcal{S}}{\forall}} \Psi^*_{v,s} = \Psi^G_{v,s} \cup \Upsilon_v \ , \tag{26}$$

where $\Psi^G_{v,s}$ denotes the set of generators of $\Psi_{v,s}$. In this union, the generators contained in Υ_v are automatically extended with zeroes for the iteration space and homogeneous dimensions.

From these relations we may now derive a multi-tuple lattice partition bijection on the product of all iteration spaces of statements that access a variable v that takes all references to v into account for every variable v. To this end we compute the HNF of $\Psi^*_{v,s}$ for each $(v,s) \in \mathcal{V} \times \mathcal{S}$ where $\Gamma^v_s \neq \emptyset$. Since the resulting generators will be ordered by the generators of the partition of Δ_v because these dimensions occur first, the generators of the multi-tuple lattice partition bijection can be extracted by concatenating the resulting generators of the partitions of the iteration space of the corresponding rows.

We thus obtain a multi-tuple lattice partition bijection on the iteration spaces for every variable where each of these relations takes the coarsening induced by all accesses to the respective variables into account. These lattice partition bijections are subsequently coarsened by taking the union of all their sets of homogeneous generators to obtain a lattice partition bijection that takes all accesses in the computation into account.

Since the variables are often not accessed by all statements in the computation, the multi-tuple relations of a variable may apply only to a subset of the set of statements. In this case, the lattice partition bijections must be aligned with each other while coarsening the multi-tuple relations of individual variables. This can be done by finding a common element in one of the iteration spaces, which can be done by solving a system of diophantine equations. If no such common element is found for any variable, then we can consider the non-overlapping subsets of statements separately. This results in a constant partition of the iteration spaces.

From the multi-tuple lattice partition bijection, we can extract the cell lattice of the resulting partition of the iteration spaces. We then add the generators of these cell lattices to $\Psi^*_{v,s}$ for each $(v,s) \in \mathcal{V} \times \mathcal{S}$ so that we obtain lattice partition bijection between the data and iteration spaces that take all accesses in the computation into account.

Finally, we determine the affine modular functions by first converting each of the generating representations of the resulting pairwise lattice partition bijections to their constraining representation using the method described in section 1.4. To obtain the the multi-tuple of affine modular functions, we combine the corresponding rows of $\mathcal{D}_X(A^T)$ where A is a matrix in HNF-form containing the generators of the pairwise lattice partition bijection and we combine the corresponding rows of $\text{HNF}(\mathcal{U}_0(A^T))$.

$$\underset{(v,s)}{\overset{\mathcal{V}\times\mathcal{S}}{\forall}} \Psi_{v,s} \triangleq \mathcal{Z}\big(\overset{\Gamma^v_s}{\underset{f}{\cup}} (\Delta_v \times \Omega_s \cap \text{Ker}\,(\mathbf{1}_{\Delta_v} - f))\big) \ . \tag{27}$$

2.2.2. Time Partitioning

For time partitioning, we can use affine partitioning which uses a double dualisation through Fourier-Motzkin projection and the specially designed Algorithm A []. Fourier-Motzkin projection has an exponential time complexity. It must however be observed that the conjunction of time constraints imposed on the pairs of time partitioning functions amounts to intersecting the cones of affine functions that result from applying Farkas'

Lemma to the individual dependence polyhedra. When considering more than two statements, the coefficients for all the functions can be gathered in a vector and the intersection can be performed on the product space of the function coefficient vectors for all statements. Preliminary experiments suggest that performing this intersection using the Parma Polyhedra Library[3] [15] is significantly faster than the original approach.

While obtaining the complete set of solutions using the affine partitioning approach is interesting, in particular for the manual analysis of specific algorithms, performing the intersection does not scale well since the size of the dual representation for polyhedra can be exponentially larger than the original representation. For this reason, it seems more interesting to use an approach like the Farkas algorithm if we are only interested in a single solution (such as for parallelisation in a production compiler). However, this algorithm uses Fourier-Motzkin projection to eliminate the Farkas multipliers which has exponential time complexity and may also significantly increase the size of the representation. Furthermore, it uses parametric integer programming (PIP) [16], which also has an exponential time complexity.

The intersection of the cones of affine functions obtained using Farkas' Lemma is equivalent to applying Farkas' Lemma after computing the convex hull of the individual dependence polyhedra. The maximally independent set of legal affine time partitioning functions is obtained by eliminating redundant constraints from this polyhedron (which can be done in polynomial time). As shown by Lim and Lam, an affine time partitioning function that maximizes parallelism can be obtained as a combination of all constraints of this polyhedron. This combination ensures that the maximal number of half-planes that constrain this polyhedron are not included in the affine sets of the resulting finer cells of operations. We can use an approach resembling the Farkas algorithm to obtain a single time partitioning solution that maximizes parallelism without computing a convex hull (or intersection) by constructing a linear programming problem that contains a variable z_i such that $0 \leq z_i \leq 1$ for every constraint of the individual dependence polyhedra and use the sum of these variables to maximize the number of these constraints that are not included. A reasoning similar as that used in the Farkas Algorithm can be used to show that $z_i \in \{0, 1\}$ in the resulting solution such that z_i can be interpreted as a binary variable while using linear programming to solve the problem. The z_i variables describe the problem with a finer granularity than in the Farkas Algorithm since a variable is used for every constraint of the dependence polyhedra rather than using a variable for every dependence and might thus lead to more parallelism.

It can be seen that, for the purpose of the extraction of maximal parallelism (in number of dimensions), the value added by a parametric solution of the problem using PIP (as in [1]) is limited. Indeed, the coefficients of the parametric parts of the scheduling functions of distinct statements are required to be equal by the Farkas Algorithm in order to ensure that the resulting time partition is no more than one dimension finer than the previous partition. The parametric part of the scheduling functions thus affects neither the structure of the time partition nor the ordering of its cells. While the quast-solution obtained by the Farkas Algorithm allows to split the context (the parameter space) into a finite union of subspaces and specify a distinct scheduling solution for each of these subspaces, we can use linear progamming instead of PIP if we use a single scheduling solution that is valid for all possible values of the parameters (as in [3]). Since we can

[3] PPL also uses double dualisation, but the dualisation is performed using Chernikova's algorithm, which is significantly more efficient than the Fourier-Motzkin based approach.

avoid the elimination of Farkas Multipliers by simply discarding the value of the Farkas multipliers in the obtained solution, the resulting method is completely polynomial.

For multi-dimensional time partitioning, the generators of the affine function space must be orthogonalized to the function space generated by the already obtained, coarser partitions prior to including them in the linear programming problem to ensure that the resulting solution is orthogonal to the affine partitioning functions of the coarser partitions.

3. Conclusion

This paper is a slightly modified version of a draft paper that was submitted to ParCo 2011 (with added proofs) and is very preliminary. Since I do not have the resources to complete this paper by increasing its clarity, adding examples, adding an experimental evaluation and adding a section on related work, I'm making it available so that it may be useful to others.

Acknowledgements

Useful information or feedback was gratefully received from Maurice Bruynooghe and Harald Devos.

Initial research that provided the starting point for this paper was supported in part by a PhD grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen)[4] and a BOF/GOA project[5] and was also morally supported by the Flexware (IWT/060068) project.

References

[1] Feautrier, P.: Some efficient solutions to the affine scheduling problem. Part I. One-dimensional time. International Journal of Parallel Programming **21**(5) (October 1992) 313–348

[2] Feautrier, P.: Some efficient solutions to the affine scheduling problem. Part II. Multidimensional time. International Journal of Parallel Programming **21**(6) (December 1992) 389–420

[3] Feautrier, P.: Scalable and modular scheduling. In: Computer Systems: Architectures, Modeling and Simulation (SAMOS). Volume 3133 of Lecture Notes in Computer Science., Springer-Verlag Berlin (2004) 433–442

[4] Lim, A.W., Lam, M.S.: Maximizing parallelism and minimizing synchronization with affine transforms. In: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM Press (1997) 201–214

[5] Lim, A.W., Lam, M.S.: Maximizing parallelism and minimizing synchronization with affine partitions. Parallel Computing **24**(3–4) (May 1998) 445–475

[6] Beletska, A., San Pietro, P.: Extracting coarse-grained parallelism with the affine transformation framework and its limitations. Electronic Modelling **5** (2006) 1–14

[7] Beletska, A., Bielecki, W., San Pietro, P.: Extracting coarse-grained parallelism in program loops with the slicing framework. In: Proceedings of the Sixth International Symposium on Parallel and Distributed Computing, Washington, DC, USA, IEEE Computer Society (2007) 29

[4] from 01/01/2008 to 31/08/2009

[5] from 01/07/2006 to 31/12/2007

[8] Yu, Y., D'Hollander, E.H.: Non-uniform dependences partitioned by recurrence chains. In: Proceedings of the 2004 International Conference on Parallel Processsing (ICPP'04), IEEE (8 2004) 100–107
[9] Yu, Y., D'Hollander, E.H.: Partitioning loops with variable dependence distances. In Lilja, D., ed.: Proceedings of the 2000 29th International Conference on Parallel Processing. Volume I., Toronto, Canada, The IEEE Computer Society (8 2000) 209–218
[10] Griebl, M., Feautrier, P.A., Lengauer, C.: Index set splitting. Int. J. Parallel Programming **28**(6) (2000) 607–631
[11] Griebl, M., Feautrier, P.A., Lengauer, C.: On index set splitting. (October 1999) 274–282
[12] Boute, R.: The euclidean definition of the functions div and mod. ACM Transactions on Programming Languages and Systems **14** (1992) 127–144
[13] Storjohann, A., Labahn, G.: Asymptotically fast computation of hermite normal forms of integer matrices. (1996) 259–266
[14] Bagnara, R., Dobson, K., Hill, P.M., Mundell, M., Zaffanella, E.: Grids: A domain for analyzing the distribution of numerical values. **4407** (2007) 219–235
[15] Bagnara, R., Hill, P.M., Zaffanella, E.: The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Sci. Comput. Program. **72** (June 2008) 3–21
[16] Feautrier, P.: Parametric integer programming. RAIRO Recherche Op'erationnelle **22** (1988)

A. Proofs

A.1. Lemma 1

$$\underset{A,B}{\overset{(\mathcal{P}\mathbb{Z}^n)^2}{\forall}} A \subseteq B \Rightarrow \mathcal{Z}A \supseteq \mathcal{Z}B \tag{28}$$

Proof. By rewriting the lemma using the definition of $\mathcal{Z}$,

$$\underset{A,B}{\overset{(\mathcal{P}\mathbb{Z}^n)^2}{\forall}} A \subseteq B \Rightarrow \{f | f \in \mathbb{F}_n \wedge A \subseteq \operatorname{Ker} f\} \supseteq \{f | f \in \mathbb{F}_n \wedge B \subseteq \operatorname{Ker} f\} \tag{29}$$

we see that the constraints on the functions in $\mathcal{Z}(A)$ is at least as strong as the constraints on the functions in $\mathcal{Z}(B)$ if $A \subseteq B$ so that we have $\mathcal{Z}(A) \supseteq \mathcal{Z}(B)$ under this condition. □ □

A.2. Lemma 2

$$\mathcal{Z} = \mathcal{Z} \circ \mathcal{H} \tag{30}$$

Proof. Since for any $S \subseteq \mathbb{Z}^n$ we have $S \subseteq \mathcal{H}S$, lemma 1 immediately gives

$$\underset{S}{\overset{\mathcal{P}\mathbb{Z}^n}{\forall}} \mathcal{Z}(S) \supseteq (\mathcal{Z} \circ \mathcal{H})(S)\ . \tag{31}$$

In order to prove

$$\underset{S}{\overset{\mathcal{P}\mathbb{Z}^n}{\forall}} \mathcal{Z}(S) \subseteq (\mathcal{Z} \circ \mathcal{H})(S) \tag{32}$$

we must show that

$$\overset{\mathcal{P}\mathbb{Z}^n}{\underset{S}{\forall}} \overset{\mathbb{F}_n}{\underset{g}{\forall}} (\overset{S}{\underset{s}{\forall}} g(s) = 0) \Rightarrow (\overset{\mathcal{H}S}{\underset{t}{\forall}} g(t) = 0) \; . \tag{33}$$

Any $g \in \mathbb{F}_n$ can be written using its linear part g_L, its constant coefficient g_C and its modular coefficient g_M where $\overset{\mathbb{Z}^n}{\underset{v}{\forall}} g(v) = \langle g_L | v \rangle + g_C \bmod g_M$. This allows us to rewrite the expression as

$$\overset{\mathcal{P}\mathbb{Z}^n}{\underset{S}{\forall}} \overset{\mathbb{F}_n}{\underset{g}{\forall}} (\overset{S}{\underset{s}{\forall}} \langle g_L | s \rangle + g_C \bmod g_M = 0) \Rightarrow (\overset{\mathcal{H}S}{\underset{t}{\forall}} \langle g_L | t \rangle + g_C \bmod g_M = 0) \; . \tag{34}$$

Any $t \in \mathcal{H}S$ must, by definition of $\mathcal{H}$, be an integral, affine combination of points in S. This means that it must be an integral combination so that the coefficient of the combination sum to 1. Assume that the set $U \subseteq S$ is affinely combined to form t, that is $t = \sum_u^U c(u)u$ and $\sum_u^U c(u) = 1$, then

$$\begin{aligned}
\langle g_L | t \rangle + g_C &= \langle g_L | \sum_u^U c(u)u \rangle + g_C \\
&= \sum_u^U c(u) \langle g_L | u \rangle + \sum_u^U c(u) g_C \\
&= \sum_u^U c(u) (\langle g_L | u \rangle + g_C) \\
&= \sum_u^U c(u) g(u) \\
&= 0
\end{aligned} \tag{35}$$

thereby concluding the proof. □ □

A.3. Lemma 4

$$\overset{\mathcal{P}\mathcal{P}\mathbb{Z}^n}{\underset{T}{\forall}} \mathcal{Z}(\overset{T}{\underset{S}{\bigcup}} S) = \overset{T}{\underset{S}{\bigcap}} \mathcal{Z}(S) \tag{36}$$

Proof. Indeed, for any $T \subseteq \mathcal{P}\mathbb{Z}^n$,

$$
\begin{aligned}
\mathcal{Z}(\bigcup_S^T S) &= \{f|(f \in \mathbb{F}_n) \wedge (\forall_s^{\bigcup_S^T S} f(s) = 0)\} \\
&= \{f|(f \in \mathbb{F}_n) \wedge (\forall_S^T \forall_s^S f(s) = 0)\} \\
&= \bigcap_S^T \{g|(g \in \mathbb{F}_n) \wedge (\forall_s^S g(s) = 0)\} \\
&= \bigcap_S^T \mathcal{Z}(S)
\end{aligned}
\tag{37}
$$

□

A.4. Lemma 5

$$
\forall_T^{\mathcal{P}\mathbb{G}_n^*} (\bigcap_S^T S \neq \emptyset) \Rightarrow (\mathcal{Z}(\bigcap_S^T S) = \sum_S^T \mathcal{Z}(S)) \tag{38}
$$

Proof. It is clear that

$$
\mathcal{Z}(\bigcap_S^T S) \supseteq \sum_S^T \mathcal{Z}(S) \ , \tag{39}
$$

since a sum of affine modular functions that each vanish on every set S in T will vanish on the intersection of those sets.

We now consider two cases:

- If the intersection of the sets in T is the empty set, then there are no constraints on the affine modular functions in the first set.
- If the intersection is not empty, let us take any function $h_\diamond : T \to \mathbb{F}_n$ for which

$$
\forall_S^T (h_S \in \mathcal{Z}(S)) \wedge (\text{Ker}\, h_S = S) \ . \tag{40}
$$

Since any affine lattice is equal to the kernel of an affine modular function, such a function exists. We can then write

$$
\forall_S^T S = \{v|(v \in \mathbb{Z}^n) \wedge (h_S(v) = 0)\} \tag{41}
$$

and

$$
\begin{aligned}
\bigcap_S^T S &= \{v|(v \in V) \wedge \forall_S^T h_S(v) = 0\} \\
&= \{v|(v \in V) \wedge (\bowtie_S^T h_S)(v) = 0)\} \ . \\
&= \text{Ker} \bowtie_S^T h_S
\end{aligned}
\tag{42}
$$

By lemma 3 we can then infer that for any $g \in \mathbb{F}_n$

$$\begin{aligned} g \in \mathcal{Z}(\bigcap_S^T S) &\equiv \mathcal{E}g \subseteq \langle \mathcal{E} \bowtie_S^T h_S \rangle_{\mathbb{Z}} \\ &\equiv \mathcal{E}g \subseteq \langle \bigcup_S^T \mathcal{E}h_S \rangle_{\mathbb{Z}} \end{aligned} \tag{43}$$

Note that

$$\langle \bigcup_S^T \mathcal{E}h_S \rangle_{\mathbb{Z}} \subseteq \mathcal{E} \sum_S^T \mathcal{Z}(S) \ . \tag{44}$$

where we have overloaded the function $\mathcal{E}$ to be applicable to a set of functions in the sense that $\mathcal{E}F \triangleq \bigcup_f^F \mathcal{E}f$ for any $F \subseteq \mathbb{F}_n$. Indeed, since $\mathcal{E}h_S \subseteq \mathcal{E}\mathcal{Z}(S)$ we can always find a function in the second set that is equal to a function in the first set by choosing a sum of elements in $\mathcal{E}h_S$ as the chosen term of $\mathcal{Z}(S)$. We can therefore derive

$$g \in \mathcal{Z}(\bigcap_S^T S) \Rightarrow \mathcal{E}g \subseteq \mathcal{E} \sum_S^T \mathcal{Z}(S) \ . \tag{45}$$

By combining this with

$$\mathcal{E}g \subseteq \mathcal{E} \sum_S^T \mathcal{Z}(S) \Rightarrow g \in \sum_S^T \mathcal{Z}(S) \tag{46}$$

we may conclude

$$\mathcal{Z}(\bigcap_S^T S) \subseteq \sum_S^T \mathcal{Z}(S) \tag{47}$$

if the intersection of the sets in T is non-emtpy.

Combining (39) with (47) then proves the lemma. □