Towards an Indexing Calculus for Efficient Distributed Array Computation

Harry B. Hunt III
Computer Science Department
University at Albany
State University of New York
Albany, NY 12222
hunt@cs.albany.edu

Lenore Mullin
Computer Science Department
University at Albany
State University of New York
Albany, NY 12222
lenore@cs.albany.edu

Daniel J. Rosenkrantz
Computer Science Department
University at Albany
State University of New York
Albany, NY 12222
djr@cs.albany.edu

December 1997

1 Introduction

High performance computing and communication is used to solve large scientific problems. Scientific programming and subsequent compilation is significantly complicated when programs are expected to execute on one or many processors for any size or dimensional problems. Although scientific programming languages are sophisticated and powerful, they have been slow to evolve from the level of operations on scalars to data parallel operations applicable to whole arrays or array sections. Moreover, the structure of arrays and the architectural topology to which the arrays are mapped, impact the efficiency, portability and scalability of algorithm design. Programming languages such as High Performance Fortran (HPF) [2] enable the programmer to specify operations on whole arrays and array sections, and to give directives indication the distribution and alignment of arrays. A compiler for such a language must mechanize a systematic method for operating on distributed arrays. In addition, array level transformations can significantly improve the performance of a given program. For instance, many array operations, such as transpose and concatenation, involve the rearrangement and replication of array elements. These operations essentially utilize array indexing, and are independent of the domain of values of the scalars involved. Often it is unnecessary to generate code for these operations. Instead of *materializing* the result of such an operation (i.e. constructing the resulting array at run–time), the compiler can keep track of how elements of the resulting array can be obtained by appropriately addressing the operands of the operation. Subsequent references to the result of the operation can be replaced by suitably modified references to the operands of the operation. We believe that nonmaterialization of partial results of selected operations can be an important compiler optimization.

This paper presents fundamental definitions for an algebraic theory of array indexing, with a focus on issues involving array addressing, distribution, decomposition, layout, and reshaping. The index calculus developed here is based on Mullin's *Psi Calculus* model of array operations [4, 3, 5, 1].

2 Formalization

2.1 Notation for Finite Sequences, Vectors, and Arrays

In this paper, we consider arrays consisting of elements from a set $\mathcal{S}$. Variables ranging over the set $\mathcal{S}$ are denoted by a through z.

Let $\mathcal{SEQ}(\mathcal{S})$ be the set of finite sequences of elements of $\mathcal{S}$. In particular, for the set $\mathcal{N}$ of natural numbers, $\mathcal{SEQ}(\mathcal{N})$ is the set of finite sequences of elements of $\mathcal{N}$. Variables ranging over finite sequences are denoted by $\hat{\alpha}$ through $\hat{\omega}$. We also denote finite sequences by enclosing the elements of a given sequence within angle brackets, e.g. $<1\ 2\ 3>$. $\hat{\Theta}$ denotes the empty sequence $< >$.

Square brackets are used around arrays of two or more dimensions. The symbols $\prec$ and $\succ$ are used around one–dimensional arrays.

2.2 The Selectors Operation

Definition 2.1 *The function* **Selectors** *has domain* $\mathcal{SEQ}(\mathcal{N})$ *and range* $2^{\mathcal{SEQ}(\mathcal{N})}$, *and is defined as follows. Let* $\hat{\alpha} = <\alpha_0\ \ldots\ \alpha_{m-1}>$ *be in* $\mathcal{SEQ}(\mathcal{N})$. *Then*

$$Selectors(\hat{\alpha}) = \{<i_0\ \ldots\ i_{m-1}> \mid 0 \leq i_j < \alpha_j, \quad for\ 0\ \leq j < m\}.$$

□

Example 2.1

$$Selectors(<3\ 2>) = \{<0\ 0>, <0\ 1>, <1\ 0>, <1\ 1>, <2\ 0>, <2\ 1>\}.$$

The way in which we will use $Selectors(\hat{\alpha})$ is that $\hat{\alpha}$ will be the shape of an array, and each element of $Selectors(\hat{\alpha})$ will be a full index selecting an element of the array.

Observe that $Selectors(\hat{\Theta}) = \{\hat{\Theta}\}$. Note that if for some j, $0 \leq j \leq m-1$, $\alpha_j = 0$, then $Selectors(\hat{\alpha}) = \{\ \}$, the empty set.

2.3 Arrays

We formalize an array as an ordered pair consisting of a *shape sequence* giving the size of each dimension, and a *mapping function* giving the value of each component of the array.

Example 2.2 *The two–dimensional array*

$$\begin{bmatrix} 8 & 3 \\ 4 & 6 \\ 5 & 9 \end{bmatrix}$$

has shape sequence $<3\ 2>$ *and mapping function* ψ*, where*

$$\psi(<0\ 0>) = 8$$

$$\psi(<0\ 1>) = 3$$

$$\psi(<1\ 0>) = 4$$

$$\psi(<1\ 1>) = 6$$

$$\psi(<2\ 0>) = 5$$

$$\psi(<2\ 1>) = 9.$$

Definition 2.2 *An* $\boldsymbol{\mathcal{S}}$**-array** ξ *is a pair* $\xi = (\hat{\alpha}, \psi)$*, where* $\hat{\alpha} \in \mathcal{SEQ}(\mathcal{N})$*, and* ψ *is a function with domain Selectors(*$\hat{\alpha}$*) and range S.* $\boldsymbol{\mathcal{ARRAY}(\mathcal{S})}$ *denotes the set of* $\mathcal{S}$*-arrays. We call* $\hat{\alpha}$ *the* **shape sequence** *of* ξ *and* ψ *the* **mapping function** *of* ξ*.* □

Note that $\mathcal{ARRAY}(\mathcal{N})$ denotes the set of $\mathcal{N}$*-arrays*. Often, in this document, $\mathcal{N}$*-arrays* will be arrays of indices.

As will be defined in Definition 4.3, the dimension of an $\mathcal{S}$*-array* $\xi = (\hat{\alpha}, \psi)$ is equal to the number of components of $\hat{\alpha}$.

2.4 Scalars

Definition 2.3 *An* $\boldsymbol{\mathcal{S}}$**-scalar** *is an* $\mathcal{S}$*-array whose first component is* $\hat{\Theta}$*.* □

Note: Recall that $Selectors(\hat{\Theta})$ is the set containing the single element $\hat{\Theta}$. Thus, from Definition 2.2, for an $\mathcal{S}$*-scalar* $(\hat{\Theta}, \psi)$, mapping function ψ is only defined on $\hat{\Theta}$, and $\psi(\hat{\Theta}) = a$ for some element $a \in \mathcal{S}$. Hence, there is a natural bijection between the set of $\mathcal{S}$*-scalars* and $\mathcal{S}$.

2.5 Vectors

Definition 2.4 *An* **$\mathcal{S}$-vector** *is an $\mathcal{S}$-array of the form $\xi = (\hat{\alpha}, \psi)$ where $\hat{\alpha}$ has one component.* **$\mathcal{VEC}(\mathcal{S})$** *denotes the set of $\mathcal{S}$-vectors.* $\Box$

There is a natural bijection $\Xi_{\mathcal{S}}$ from $\mathcal{VEC}(\mathcal{S})$ to $\mathcal{SEQ}(\mathcal{S})$, defined as follows. For $\mathcal{S}$-*vector* $\xi = (<m>, \psi)$,

$$\Xi_{\mathcal{S}}(\xi) \text{ equals the sequence } \hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}> \text{ where } \alpha_i = \psi(i), \text{ for } 0 \ \leq i < m.$$

In particular, for $\mathcal{N}$-*vectors*, $\Xi_{\mathcal{N}}$ is a bijection from $\mathcal{VEC}(\mathcal{N})$ to $\mathcal{SEQ}(\mathcal{N})$. For example, consider the $\mathcal{N}$-*vector* $\xi = \prec 47 \ 85 \ 14 \succ = (<3>, \psi)$ where

$$\psi(<0>) = 47$$

$$\psi(<1>) = 85$$

$$\psi(<2>) = 14.$$

Then $\Xi_{\mathcal{N}}(\xi)$ is equal to $<47 \ 85 \ 14>$, and $\Xi_{\mathcal{N}}^{-1}(<47 \ 85 \ 14>)$ is equal to ξ.

For functions defined on $\mathcal{S}$-*vectors*, we will often want to apply such a function to a sequence, and will accomplish this by applying $\Xi_{\mathcal{S}}^{-1}$ to the sequence, and then applying the function to the resulting $\mathcal{S}$-*vector*. For an example where this is done, see Observation 4.3

2.6 Empty Arrays

Given an $\mathcal{S}$-*array* $\xi = (\hat{\alpha}, \psi)$, if at least one component of $\hat{\alpha}$ is zero, then $Selectors(\hat{\alpha}) = \{ \ \}$, so ψ has empty domain.

Definition 2.5 *An* **empty array** *is an $\mathcal{S}$-array $\xi = (\hat{\alpha}, \psi)$ such that at least one component of $\hat{\alpha}$ is zero. We let $\vec{\Theta}$ denote the $\mathcal{S}$-vector $(<0>, \psi)$, where ψ has empty domain.* $\Box$

Observation 2.1

$$\Xi(\vec{\Theta}) \ = \ \hat{\Theta}.$$

$\Box$

Given $\hat{\alpha} \in \mathcal{SEQ}(\mathcal{N})$ such that at least one component of $\hat{\alpha}$ is zero, we let $\Theta_{\hat{\alpha}}$ denote the array $(\hat{\alpha}, \psi)$, where ψ has empty domain. Note that $\vec{\Theta}$ is the same as $\Theta_{<0>}$, and that $\vec{\Theta}$ is the only empty one–dimensional array. For any $n > 1$, there are an infinite number of empty n–dimensional arrays. For instance, $\Theta_{<0 \ 0>}$, $\Theta_{<a \ 0>}$ for any $a \in \mathcal{N}$, and $\Theta_{<0 \ b>}$ for any $b \in \mathcal{N}$, are all empty two–dimensional arrays.

3 Operations on Finite Sequences

3.1 product – $\boldsymbol{\pi}$

Definition 3.1 *The function* **product***, denoted by* $\boldsymbol{\pi}$*, has domain* $\mathcal{SEQ}(\mathcal{N})$ *and range* $\mathcal{N}$*, and is defined as follows. Let* $m \in \mathcal{N}$*, and let* $\hat{\alpha}$ *in* $\mathcal{SEQ}(\mathcal{N})$ *be* $<\alpha_0 \ \ldots \ \alpha_{m-1}>$*. Then*

$$\pi(\hat{\alpha}) \ = \ \prod_{i=0}^{(m-1)} \alpha_i \ \ \textit{when } m \ > \ 0,$$

and

$$\pi(\hat{\Theta}) \ = \ 1 \qquad \textit{when } m = 0.$$

□

3.2 Partitioning Functions – Take and Drop

Definition 3.2 *The partial function* **Take***, has domain* $\mathcal{SEQ}(\mathcal{S}) \times \mathcal{Z} \times \mathcal{Z}$ *and range* $\mathcal{SEQ}(\mathcal{S})$*. Let* $\hat{x} \ = < x_0 \ \ldots \ x_{m-1} > \in \mathcal{SEQ}(\mathcal{S})$*. Let* p *and* $q \in \mathcal{Z}$ *satisfy the constraints that either* $0 \le p \le q < m$ *or* $p = m$ *and* $q = m - 1$*. Then*

$$\textit{Take}(\hat{x}, p, q) \ = \ < x_p \ \ldots \ x_q >.$$

□

Definition 3.3 *The partial function* **Drop***, has domain* $\mathcal{SEQ}(\mathcal{S}) \times \mathcal{Z} \times \mathcal{Z}$ *and range* $\mathcal{SEQ}(\mathcal{S})$*. Let* $\hat{x} \ = < x_0 \ \ldots \ x_{m-1} > \in \mathcal{SEQ}(\mathcal{S})$*. Let* p *and* $q \in \mathcal{Z}$ *satisfy the constraints that either* $0 \le p \le q < m$ *or* $p = 0$ *and* $q = -1$*. Then*

$$\textit{Drop}(\hat{x}, p, q) \ = \ < x_0 \ \ldots \ x_{p-1} \ x_{q+1} \ \ldots \ x_{m-1} >.$$

□

Observation 3.1 *For* $\hat{x} \ \in \mathcal{SEQ}(\mathcal{S})$*, let* $m = \tau(\hat{x})$*. Then, the following hold.*

$$\textit{Take}(\hat{x}, 0, m-1) = \hat{x}.$$

$$\textit{Take}(\hat{x}, m, m-1) = \hat{\Theta}.$$

$$\textit{Drop}(\hat{x}, 0, -1) = \hat{x}.$$

$$\textit{Drop}(\hat{x}, 0, m-1) = \hat{\Theta}.$$

□

4 Operations on Arrays

4.1 Structural Functions

4.1.1 shape – $\boldsymbol{\rho}$

Definition 4.1 *The function* **shape***, denoted by* $\boldsymbol{\rho}$*, with domain* $\mathcal{ARRAY}(\mathcal{S})$ *and range* $\mathcal{SEQ}(\mathcal{N})$ *is defined by* $\rho((\hat{\alpha}, \psi)) = \hat{\alpha}$*.* □

4.1.2 total – $\boldsymbol{\tau}$

The function **total** gives the number of elements in an array.

Definition 4.2 *The function* **total**, *denoted by* $\boldsymbol{\tau}$, *with domain* $\mathcal{ARRAY}(\mathcal{S})$ *and range* $\mathcal{N}$ *is defined by* $\tau((\hat{\alpha}, \psi)) = \pi(\hat{\alpha})$. $\Box$

Observation 4.1 *For* $\hat{x} \in \mathcal{SEQ}(\mathcal{S})$, $p \in \mathcal{Z}$, *and* $q \in \mathcal{Z}$ *such that* $Take(\hat{x}, p, q)$ *is defined,*

$$\tau(\Xi_{\mathcal{S}}^{-1}(\mathit{Take}(\hat{x}, p, q))) = q - p + 1.$$

$\Box$

Observation 4.2 *For* $\hat{x} \in \mathcal{SEQ}(\mathcal{S})$, $p \in \mathcal{Z}$, *and* $q \in \mathcal{Z}$ *such that* $Drop(\hat{x}, p, q)$ *is defined,*

$$\tau(\Xi_{\mathcal{S}}^{-1}(\mathit{Drop}(\hat{x}, p, q))) = \tau(\Xi_{\mathcal{S}}^{-1}(\hat{x})) - q + p - 1.$$

$\Box$

4.1.3 dimension – $\boldsymbol{\delta}$

Definition 4.3 *The function* **dimension**, *denoted by* $\boldsymbol{\delta}$, *with domain* $\mathcal{ARRAY}(\mathcal{S})$ *and range* $\mathcal{N}$ *is defined by* $\delta((\hat{\alpha}, \psi)) = m$ *where* $\hat{\alpha} = \ <\alpha_0 \ \ldots \ \alpha_{m-1}>$. $\Box$

Observe for an $\mathcal{S}$-*scalar* $\xi = (\hat{\Theta}, \psi)$, $\delta(\xi) = 0$ and $\tau(\xi) = 1$. For an $\mathcal{S}$-*vector* $\xi = (<a>, \psi)$, $\delta(\xi) = 1$ and $\tau(\xi) = a$. For the array ξ in Example 2.2, $\delta(\xi) = 2$ and $\tau(\xi) = 6$.

Observation 4.3 *For any* $\mathcal{S}$-*array* $(\hat{\alpha}, \psi)$,

$$\delta((\hat{\alpha}, \psi)) = \tau(\Xi_{\mathcal{N}}^{-1}(\hat{\alpha})).$$

$\Box$

For notational convenience in the rest of this paper, we will usually drop the explicit use of $\Xi_{\mathcal{N}}^{-1}$, and write an expression such as occurs in Observation 4.3 in the simpler form $\delta((\hat{\alpha}, \psi)) = \tau(\hat{\alpha})$. Thus, we will use the shorthand notation $\tau(\hat{\alpha})$ for $\hat{\alpha}$ in $\mathcal{SEQ}(\mathcal{S})$, instead of the longer $\tau(\Xi_{\mathcal{S}}^{-1}(\hat{\alpha}))$.

4.1.4 sequence–concatenation and vector–concatenation

Definition 4.4 *The function* **sequence–concatenation**, *denoted by* $+\!\!+_{\boldsymbol{s}}$, *has domain* $\mathcal{SEQ}(\mathcal{N}) \times \mathcal{SEQ}(\mathcal{N})$ *and range* $\mathcal{SEQ}(\mathcal{N})$, *and is defined as follows.*

$$<\alpha_0 \ \ldots \ \alpha_{m-1}> +\!\!+_{\boldsymbol{s}} <\beta_0 \ \ldots \ \beta_{n-1}> \ = \ <\alpha_0 \ \ldots \ \alpha_{m-1} \ \beta_0 \ \ldots \ \beta_{n-1}>.$$

The function **vector–concatenation**, *denoted by* $+\!\!+_{\boldsymbol{v}}$, *has domain* $\mathcal{VEC}(\mathcal{S}) \times \mathcal{VEC}(\mathcal{S})$ *and range* $\mathcal{VEC}(\mathcal{S})$, *and is defined as follows. Let* $\xi_1 = (<r_1>, \psi_1)$ *and* $\xi_2 = (<r_2>, \psi_2) \in \mathcal{VEC}(\mathcal{S})$. *Then* $\xi_1 +\!\!+_v \xi_2 = \xi_3$ *where* $\xi_3 = (<r_1 + r_2>, \psi_3)$, *with*

$$\psi_3(<i>) = \begin{cases} \psi_1(<i>) & \text{if } 0 \leq i < r_1 \\ \psi_2(<i - r_1>) & \text{if } r_1 \leq i < r_1 + r_2. \end{cases}$$

$\Box$

Observation 4.4 *For any $\hat{\alpha}$ and $\hat{\beta}$ in $\mathcal{SEQ}(\mathcal{S})$,*

$$\hat{\alpha} \ \#_s \ \hat{\beta} = \Xi_{\mathcal{S}}(\Xi_{\mathcal{S}}^{-1}(\hat{\alpha}) \ \#_v \ \Xi_{\mathcal{S}}^{-1}(\hat{\beta})).$$

$\square$

Observation 4.5 *For any $\hat{\alpha}$ in $\mathcal{SEQ}(\mathcal{S})$,*

$$\hat{\alpha} \ \#_s \ \hat{\Theta} \ = \ \hat{\Theta} \ \#_s \ \hat{\alpha} \ = \ \hat{\alpha}.$$

$\square$

4.2 index–generator – $\boldsymbol{\iota}$

Definition 4.5 *The function* **index–generator***, denoted by $\boldsymbol{\iota}$, has domain $\mathcal{SEQ}(\mathcal{N})$ and range $\mathcal{ARRAY}(\mathcal{N})$, and is defined as follows. Let $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}> \ \in \mathcal{SEQ}(\mathcal{N})$. Then $\iota(\hat{\alpha}) = (\hat{\alpha}^{'}, \psi^{'})$, where $\hat{\alpha}^{'} = <\alpha_0 \ \ldots \ \alpha_{m-1} \ m>$ and $\psi^{'}$ is as follows. For $\hat{\beta} = <i_0 \ \ldots \ i_m> \ \in$ $Selector(\hat{\alpha}^{'})$, $\psi^{'}(\beta) = i_{i_m}$.* $\square$

Observation 4.6 *For any $\hat{\alpha} \in \mathcal{SEQ}(\mathcal{N})$, $\iota(\hat{\alpha}) = (\hat{\alpha}^{'}, \psi^{'})$, where $\hat{\alpha}^{'} = \hat{\alpha} \#_s \rho(\hat{\alpha})$.* $\square$

Note that in writing Observation 4.6, we are using the shorthand expression $\rho(\hat{\alpha})$ instead of the more explicit expression $\rho(\Xi_{\mathcal{N}}^{-1}(\hat{\alpha}))$.

For example, suppose $\hat{\alpha} = <3 \ 5>$. Then $\iota(<3 \ 5>) = (<3 \ 5 \ 2>, \psi^{'})$, as follows.

$$\iota(<3 \ 5>) = \begin{bmatrix} \begin{bmatrix} 0 \ 0 \\ 0 \ 1 \\ 0 \ 2 \\ 0 \ 3 \\ 0 \ 4 \end{bmatrix} \\ \\ \begin{bmatrix} 1 \ 0 \\ 1 \ 1 \\ 1 \ 2 \\ 1 \ 3 \\ 1 \ 4 \end{bmatrix} \\ \\ \begin{bmatrix} 2 \ 0 \\ 2 \ 1 \\ 2 \ 2 \\ 2 \ 3 \\ 2 \ 4 \end{bmatrix} \end{bmatrix}$$

For example, $\psi^{'}(<2 \ 4 \ 0>) = 2$, $\psi^{'}(<2 \ 4 \ 1>) = 4$, $\psi^{'}(<1 \ 3 \ 0>) = 1$, and $\psi^{'}(<1 \ 3 \ 1>) = 3$.

4.3 The Index Function $\mathbf{\Psi}$

Definition 4.6 *The partial function* **Index–Via–Sequence**, *denoted by* $\mathbf{\Psi}$, *has domain* $\mathcal{SEQ}(\mathcal{N}) \times \mathcal{ARRAY}(\mathcal{S})$ *and range* $\mathcal{ARRAY}(\mathcal{S})$. *Let* $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$, *where* $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{n-1}>$. *Let* $\hat{i} = <i_0 \ \ldots \ i_{m-1}> \ \in \mathcal{SEQ}(\mathcal{N})$, *where* $\hat{i}$ *satisfies the constraints that* $m \le n$ *and* $0 \le i_j < \alpha_j$ *for* $0 \le j < m$. *Let* $p = n - m$. *Then* $\hat{i}\mathbf{\Psi}\xi = (\hat{\alpha}', \psi')$ *is defined as follows.*

(a) $\hat{\alpha}' = \ <\alpha'_0 \ \ldots \ \alpha'_{p-1}>$ *where* $\alpha'_j = \alpha_{m+j}$ *for* $0 \le j < p$.

(b) for each $\hat{\beta} \in$ *Selectors*$(\hat{\alpha}')$, $\psi'(\hat{\beta}) = \psi(\hat{i} \mathbin{+\!\!+}_s \hat{\beta})$. □

Later, in Section 4.5, we will define a generalization of $\mathbf{\Psi}$ where the left operand is an $\mathcal{N}$*-array*.

Suppose that $\tau(\hat{i}) = \delta(\xi)$. Then $p = 0$, so $\hat{\alpha}' = \hat{\Theta}$. Note that *Selectors*$(\hat{\Theta}) = \{\hat{\Theta}\}$, and that $\psi'(\hat{\Theta}) = \psi(\hat{i})$. Therefore, in this case the $\mathbf{\Psi}$ operation returns an $\mathcal{S}$*-scalar*, as selected from ξ by $\hat{i}$.

Suppose that $\hat{i} = \hat{\Theta}$. Then $p = n$, so $\hat{\alpha}' = \hat{\alpha}$ and $\hat{\Theta}\mathbf{\Psi}\xi = \xi$, that is the whole array, yielding the following identity.

Observation 4.7 *For any* $\mathcal{S}$*-array* ξ, $\hat{\Theta}\mathbf{\Psi}\xi = \xi$. □

Observation 4.8 *For* $\hat{i} \in \mathcal{SEQ}(\mathcal{S})$ *and* $\xi \in \mathcal{ARRAY}(\mathcal{S})$, *such that* $\hat{i}\mathbf{\Psi}\xi$ *is defined,*

$$\rho(\hat{i}\mathbf{\Psi}\xi) \ = \ \mathit{Take}(\rho(\xi), \tau(\hat{i}), \delta(\xi) - 1) \ = \ \mathit{Drop}(\rho(\xi), 0, \tau(\hat{i}) - 1).$$

□

Example 4.1 *Consider the three–dimensional array*

$$\xi = (<2\ 3\ 4>, \psi) = \begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \\ \\ \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}.$$

Then

$$\hat{\Theta}\mathbf{\Psi}\xi = \begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \\ \\ \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}.$$

$$<0>\mathbf{\Psi}\xi = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix}.$$

$$<1\ 2>\mathbf{\Psi}\xi = \prec 20\ 21\ 22\ 23 \succ.$$

$$<0\ 1\ 2>\mathbf{\Psi}\xi = (\hat{\Theta}, \psi') \text{ where } \psi'(\hat{\Theta}) = 6.$$

Definition 4.7 *Let R be a binary relation on S. Then R_* is the binary relation on $\mathcal{SEQ}(\mathcal{S})$, defined as follows. Let $\hat{v} = <v_0, \ldots, v_{m-1}>$ and $\hat{w} = <w_0, \ldots, w_{p-1}>$. Then $\hat{v}\ R_*\ \hat{w}$ is true iff $m \leq p$ and $v_j\ R\ w_j$ for all j, $0 \leq j < m$.* □

For example, consider the relation $<_*$ on $\mathcal{SEQ}(\mathcal{N})$. $<5\ 3\ 4>\ <_*\ <6\ 4\ 5>$, and $<5\ 3\ 4>\ <_*\ <6\ 4\ 5\ 1>$. Note that that it is not the case that $<5\ 3\ 4>\ <_*\ <5>$, so $<_*$ does not correspond to lexicographic ordering based on $<_*$.

Observation 4.9 *For all $\hat{\alpha} \in \mathcal{SEQ}(\mathcal{N})$, $\hat{\Theta} <_* \hat{\alpha}$.* □

In particular, $\hat{\Theta} <_* \hat{\Theta}$. Thus, although $<$ on $\mathcal{N}$ is irreflexive, $<_*$ on $\mathcal{SEQ}(\mathcal{N})$ is not irreflexive.

Definition 4.8 *$\hat{i} \in \mathcal{SEQ}(\mathcal{N})$ is a* **valid index** *for an array ξ if $\hat{i} <_* \rho(\xi)$.* □

Definition 4.9 *$\hat{i} \in \mathcal{SEQ}(\mathcal{N})$ is a* **full index** *for an array ξ if $\hat{i}$ is a valid index for ξ and $\tau(\hat{i}) = \delta(\xi)$.* □

Observation 4.10 *For $\hat{i} \in \mathcal{SEQ}(\mathcal{N})$ and $\mathcal{S}$-array ξ, $\hat{i}\mathbf{\Psi}\xi$ is defined iff $\hat{i}$ is a valid index for ξ.* □

Observation 4.11 *If $\hat{i}$ and $\hat{j} \in \mathcal{SEQ}(\mathcal{N})$ are such that $\hat{i} +\!\!+_s \hat{j}$ is a valid index for $\mathcal{S}$-array ξ, then $(\hat{i} +\!\!+_s \hat{j})\mathbf{\Psi}\xi = \hat{j}\mathbf{\Psi}(\hat{i}\mathbf{\Psi}\xi)$.* □

Observation 4.12 *For any $\hat{i} \in \mathcal{SEQ}(\mathcal{N})$ and $\mathcal{S}$-array ξ such that $\hat{i}\mathbf{\Psi}\xi$ is defined,*

$$\rho(\hat{i}\mathbf{\Psi}\xi) = Drop(\rho(\xi), 0, \tau(\hat{i}) - 1).$$

□

Observation 4.13 *For $\hat{\alpha} \in \mathcal{SEQ}(\mathcal{N})$,*

$$Selectors(\hat{\alpha}) = \{\ \hat{i} \mid \hat{i} \in \mathcal{SEQ}(\mathcal{N}), \tau(\hat{i}) = \tau(\hat{\alpha}),\ and\ \hat{i} <_* \hat{\alpha}\}.$$

□

Observation 4.14 *For $\hat{i}$ and $\hat{\alpha} \in \mathcal{SEQ}(\mathcal{N})$ such that $\hat{i} \in Selectors(\hat{\alpha})$,*

$$\hat{i}\mathbf{\Psi}\iota(\hat{\alpha}) = \hat{i}.$$

□

4.4 Arrays of Sequences

Recall that given a set $\mathcal{S}$, $\mathcal{SEQ}(\mathcal{S})$ is the set of finite sequences of elements of $\mathcal{S}$. Such finite sequences can themselves be elements of an array. Thus, $\boldsymbol{\mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))}$ denotes the set of arrays whose components are members of $\mathcal{SEQ}(\mathcal{S})$.

Definition 4.10 *An array $\zeta \in \mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))$ is* **uniform** *if there is a $k \in \mathcal{N}$ such that each component of ζ contains k elements.* **Uniform–$\boldsymbol{\mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))}$** *denotes the set of $\mathcal{S}$-arrays that are uniform.* $\square$

There is a natural bijection between $\mathcal{ARRAY}(\mathcal{S})$ and *Uniform–*$\mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))$, defined as follows.

Definition 4.11 *The function* **ArrayToArrayOfSeq** *has domain $\mathcal{ARRAY}(\mathcal{S})$ and range Uniform–$\mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))$. Let $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$, where $\hat{\alpha} = <\alpha_0\ \ldots\ \alpha_{m-1}>$. Then $ArrayToArrayOfSeq(\xi) = (\hat{\alpha}', \psi')$, where $\hat{\alpha}' = <\alpha_0\ \ldots\ \alpha_{m-2}>$ and ψ' is as follows. For $\hat{i}\ \in Selector(\hat{\alpha}')$, $\psi'(\hat{i}) = \Xi(\hat{i}\boldsymbol{\Psi}\xi)$.*

The function **ArrayOfSeqToArray** *has domain Uniform–$\mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))$ and range $\mathcal{ARRAY}(\mathcal{S})$. Let $\zeta = (\hat{\alpha}, \psi) \in$ Uniform–$\mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))$, where $\hat{\alpha} = <\alpha_0\ \ldots\ \alpha_{m-1}>$, and each component of ζ is a sequence containing k elements. Then $ArrayOfSeqToArray(\zeta) = (\hat{\alpha}', \psi')$, where $\hat{\alpha} = <\alpha_0\ \ldots\ \alpha_{m-1}\ k>$, and ψ' is as follows. For $\hat{i} = <i_0\ \ldots\ i_{m-1}\ i_m> \in Selector(\hat{\alpha}')$, $\psi'(\hat{i}) =$ Take$(\psi(<i_0\ \ldots\ i_{m-1}>), i_m, i_m)$.* $\square$

Example 4.2 *Consider the three–dimensional array*

$$\xi = (<2\ 3\ 4>, \psi) = \left[\begin{array}{c} \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{array}\right] \\ \\ \left[\begin{array}{cccc} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{array}\right] \end{array}\right].$$

Then $ArrayToArrayOfSeq(\xi) = \zeta$, where

$$\zeta = (<2\ 3>, \psi') = \left[\begin{array}{ccc} <0\ 1\ 2\ 3> & <4\ 5\ 6\ 7> & <8\ 9\ 10\ 11> \\ <12\ 13\ 14\ 15> & <16\ 17\ 18\ 19> & <20\ 21\ 22\ 23> \end{array}\right].$$

Observation 4.15 *For $\xi \in \mathcal{ARRAY}(\mathcal{S})$,*

$$ArrayOfSeqToArray(ArrayToArrayOfSeq(\xi)) = \xi.$$

For $\zeta \in$ Uniform–$\mathcal{ARRAY}(\mathcal{SEQ}(\mathcal{S}))$,

$$ArrayToArrayOfSeq(ArrayOfSeqToArray(\zeta)) = \zeta.$$

$\square$

4.5 Generalizing the Index Function – $\tilde{\mathbf{\Psi}}$

The following definition generalizes the function *Index–Via–Sequence* (i.e., $\mathbf{\Psi}$ from Definition 4.6) so that its left operand can be an array of full indices. This left operand, denoted as ζ in Definition 4.12, is an m–dimensional array of elements of $\mathcal{N}$. But it is used as a $(m-1)$–dimensional array of $\mathcal{N}$*-vectors*. Each of these $\mathcal{N}$*-vectors* is interpreted as a sequence that is a full index of the right operand, denoted as ξ in Definition 4.12. The result of the operation is a $(m-1)$–dimensional array of elements from the right operand ξ, as selected by these full indices.

The last dimension of ζ must be equal to the number of dimensions of ξ. For instance, suppose that ξ is a 3–dimensional $\mathcal{S}$*-array* (so that $\delta(\xi) = 3$) and $\rho(\xi) = <50\ 30\ 70>$. Then the last component of the shape sequence of ζ must be 3. For instance, if $\rho(\zeta) = <800\ 180\ 60\ 200\ 3>$, then the result of the operation is an $\mathcal{S}$*-array* whose shape sequence is $<800\ 180\ 60\ 200>$. In addition to the constraint on the last component of the shape sequence of ζ, there is a constraint on the values in ζ. Letting m be $\delta(\zeta)$, this constraint is that each $m-1$–component valid index for ζ must select a 1–dimensional sub–array of ζ that can be used as a full index for ξ. In the above example, where ζ is a 5–dimensional array, this constraint requires that each 1–dimensional sub–array of ζ selected by a 4–component valid index for ζ can be used as a full index for ξ. More precisely, for each $<i_0\ i_1\ i_2\ i_3>$ that is a valid index for ζ, $\Xi(<i_0\ i_1\ i_2\ i_3>\mathbf{\Psi}\ \zeta)$ must be a full index of ξ. Let ω be the mapping function of ζ. Then $\Xi(<i_0\ i_1\ i_2\ i_3>\mathbf{\Psi}\ \zeta) = <\omega(i_0\ i_1\ i_2\ i_3\ 0)\ \omega(i_0\ i_1\ i_2\ i_3\ 1)\ \omega(i_0\ i_1\ i_2\ i_3\ 2)>$. Thus, the constraint on the values in ζ is that $\omega(i_0\ i_1\ i_2\ i_3\ 0) < 50$, $\omega(i_0\ i_1\ i_2\ i_3\ 1) < 30$, and $\omega(i_0\ i_1\ i_2\ i_3\ 2) < 70$. Suppose we express $\rho(\xi)$ as $<\alpha_0\ \alpha_1\ \alpha_2>$. (In the current example, $\alpha_0 = 50$, $\alpha_1 = 30$, and $\alpha_2 = 70$.) Then the constraint on the values in ζ can be expressed succinctly as $\omega(i_0\ i_1\ i_2\ i_3\ i_4) < \alpha_{i_4}$.

Definition 4.12 *The partial function* **Index–Via–Array**, *denoted by* $\tilde{\mathbf{\Psi}}$, *has domain* $\mathcal{ARRAY}(\mathcal{N}) \times \mathcal{ARRAY}(\mathcal{S})$ *and range* $\mathcal{ARRAY}(\mathcal{S})$. *Let* $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$, *where* $\hat{\alpha} = <\alpha_0\ \ldots\ \alpha_{n-1}>$. *Let* $\zeta = (\hat{\sigma}, \omega) \in \mathcal{ARRAY}(\mathcal{N})$, *where* $\hat{\sigma} = <\sigma_0\ \ldots\ \sigma_{m-1}>$. *Let* ζ *satisfy the constraints that* $\sigma_{m-1} = n$, *and for all* $\hat{i} = <i_0\ \ldots\ i_{m-1}> \in Selectors(\hat{\sigma})$, $\omega(\hat{i}) < \alpha_{i_{m-1}}$. *Then* $\zeta\tilde{\mathbf{\Psi}}\xi = (\hat{\sigma}', \psi')$ *is defined as follows.*

(a) $\hat{\sigma}' = <\sigma'_0\ \ldots\ \sigma'_{m-2}>$ *where* $\sigma'_j = \sigma_j$ *for* $0 \le j \le m-2$.

(b) for each $\hat{\beta} = <\beta_0\ \ldots\ \beta_{m-2}> \in Selectors(\hat{\sigma}')$, $\psi'(\hat{\beta}) = \psi(\Xi(\hat{\beta}\mathbf{\Psi}\zeta))$. □

Observation 4.16 *The constraints in Definition 4.12 are equivalent to requiring that for every* $(m-1)$*–component valid index for* ζ, $\hat{i} = <i_0\ \ldots\ i_{m-2}>$,

$$\Xi(\hat{i}\mathbf{\Psi}\zeta) \text{ is a full index for } \xi.$$

Also, in Definition 4.12(b),

$$\Xi(\hat{\beta}\mathbf{\Psi}\zeta) \;=\; <\omega(\hat{\beta} +\!\!+_s <0>)\ \ldots\ \omega(\hat{\beta} +\!\!+_s <n-1>)>.$$

□

For examples of $\tilde{\mathbf{\Psi}}$, let

$$\xi = (<2\ 3\ 4>, \psi) = \begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \\ \\ \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}.$$

Suppose

$$\zeta = \begin{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 1 & 3 \\ 1 & 2 & 2 \end{bmatrix} \\ \\ \begin{bmatrix} 1 & 1 & 3 \\ 0 & 0 & 2 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 2 & 0 \end{bmatrix} \\ \\ \begin{bmatrix} 0 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \\ 0 & 1 & 1 \end{bmatrix} \end{bmatrix}.$$

Since $\rho(\zeta) = <3\ 5\ 3>$, $\zeta\tilde{\mathbf{\Psi}}\xi$ has shape sequence $<3\ 5>$, and is the following array.

$$\zeta\tilde{\mathbf{\Psi}}\xi = \begin{bmatrix} 23 & 0 & 6 & 7 & 22 \\ 19 & 2 & 18 & 5 & 8 \\ 11 & 23 & 22 & 23 & 5 \end{bmatrix}.$$

To see how a typical value in the above array is computed, note that

$$\psi'(<2\ 4>) = \psi(\Xi(<2\ 4>\mathbf{\Psi}\zeta)) = \psi(\Xi(\prec 0\ 1\ 1\succ)) = \psi(<0\ 1\ 1>) = 5.$$

Now, suppose

$$\zeta = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 2 & 1 \\ 1 & 2 & 0 \\ 0 & 2 & 1 \\ 1 & 2 & 3 \\ 1 & 2 & 2 \end{bmatrix}.$$

Since $\rho(\zeta) = <6\ 3>$, $\zeta \tilde{\mathbf{\Psi}} \xi$ has shape sequence $<6>$, and is the following array.

$$\zeta \tilde{\mathbf{\Psi}} \xi = \prec 6\ 9\ 20\ 9\ 23\ 22 \succ.$$

Now, suppose $\zeta = \prec 0\ 1\ 2 \succ$. Since $\rho(\zeta) = <3>$, the shape sequence of $\zeta \tilde{\mathbf{\Psi}} \xi$ is $\hat{\Theta}$, and $\zeta \tilde{\mathbf{\Psi}} \xi$ is a $\mathcal{S}$-*scalar*, as follows.

$$\prec 0\ 1\ 2 \succ \tilde{\mathbf{\Psi}} \xi = (\hat{\Theta}, \psi') \text{ where } \psi'(\hat{\Theta}) = 6.$$

Observation 4.17 *For* $\xi \in \mathcal{ARRAY}(\mathcal{S})$,

$$\iota(\rho(\xi))\ \tilde{\mathbf{\Psi}}\ \xi\ =\ \xi.$$

□

Suppose that ξ is a $\mathcal{S}$-*scalar*, $\xi = (\hat{\Theta}, \psi)$, where $\psi(\hat{\Theta}) = a$ for some element $a \in \mathcal{S}$. Suppose that ζ is an empty array, and the last component of its shape vector is zero. For concreteness, suppose that ζ is $\Theta_{<5\ 3\ 0>}$ Then, $\zeta \tilde{\mathbf{\Psi}} \xi = (<5\ 3>, \psi')$, where for each $\hat{i} \in$ *Selectors*$(<5\ 3>)$, $\psi'(\hat{i}) = a$. I.e.,

$$\Theta_{<5\ 3\ 0>} \tilde{\mathbf{\Psi}} \xi = \begin{bmatrix} a & a & a \\ a & a & a \\ a & a & a \\ a & a & a \\ a & a & a \end{bmatrix}.$$

Observation 4.18 *Let ξ be an $\mathcal{S}$-scalar, $(\hat{\Theta}, \psi)$ where $\psi(\hat{\Theta}) = a$ for some element $a \in \mathcal{S}$. For any $\hat{\alpha} \in \mathcal{SEQ}(\mathcal{N})$, let ζ be the empty array with shape vector $\hat{\alpha} +\!\!+_s <0>$. Then $\zeta \tilde{\mathbf{\Psi}} \xi = (\hat{\alpha}, \psi')$, where for each $\hat{i} \in$* Selectors(α), *$\psi'(\hat{i}) = a$.* □

The function $\tilde{\mathbf{\Psi}}$ can be generalized so that its left operand can be an array of valid indices, which are not necessarily full indices.

Definition 4.13 *The partial function* **Index–Via–Array**, *denoted by $\tilde{\mathbf{\Psi}}$, has domain $\mathcal{ARRAY}(\mathcal{N}) \times \mathcal{ARRAY}(\mathcal{S})$ and range $\mathcal{ARRAY}(\mathcal{S})$. Let $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$, where $\hat{\alpha} = < \alpha_0\ \ldots\ \alpha_{n-1} >$. Let $\zeta = (\hat{\sigma}, \omega) \in \mathcal{ARRAY}(\mathcal{N})$, where $\hat{\sigma} = < \sigma_0\ \ldots\ \sigma_{m-1} >$. Let ζ satisfy the constraints that $\sigma_{m-1} \leq n$, and for each $\hat{i} = < i_0\ \ldots\ i_{m-2} >$ that is a valid index for ζ, $\Xi(\hat{i} \mathbf{\Psi} \zeta)$ is a valid index for ξ. Then $\zeta \tilde{\mathbf{\Psi}} \xi = (\hat{\sigma}', \psi')$ is defined as follows. Let $p = m - 1 + n - \sigma_{m-1}$.*

(a)

$$\hat{\sigma}' = < \sigma'_0\ \ldots\ \sigma'_{p-1} > \text{ where } \sigma'_j = \begin{cases} \sigma_j & \text{for } 0 \leq j \leq m-2 \\ \alpha_{j-m+1+\sigma_{m-1}} & \text{for } m-2 < j < p. \end{cases}$$

(b) for each $\hat{\beta} = <\beta_0\ \ldots\ \beta_{p-1}> \in$ Selectors$(\hat{\sigma}')$,

$$\psi'(\hat{\beta}) = \psi(\ \Xi(<\beta_0\ \ldots\ \beta_{m-2}> \mathbf{\Psi} \zeta)\ +\!\!+_s <\beta_{m-1}\ \ldots\ \beta_{p-1}>\ ).$$

□

4.6 Mapping Functions for Layout

Computer languages vary in the way they lay out data in memory. For example, C stores arrays in row major order, while FORTRAN stores arrays in column major order. Here, we formalize mapping functions that correspond to row major and column major layouts in memory. More generally, we want to define mapping functions appropriate to various layouts of array elements into memory, or appropriate to various processor interconnection network topologies.

We introduce mapping functions between the full indices of an array, and the offsets of the elements of the array when the array is layed out linearly using row major or column major ordering. These mapping functions represent the correspondence between the full index selecting a given array element, and the offset of that array element in the layout. For a given layout method λ, function *Index–to–Offset*$_\lambda$ takes a full index and a shape, and returns an offset based on layout method λ. Function *Offset–to–Index*$_\lambda$ takes an offset and a shape, and returns a full index sequence based on layout method λ. Note that the mappings between full index and offset are different for row and column major ordering.

Example 4.3 *Consider the following two–dimensional array:*

$$\xi = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}.$$

In row major order, the layout is 0, 1, 2, 3, 4, 5. In column major order, the layout is 0, 3, 1, 4, 2, 5. Let $@\xi$ denote the address in memory of the first element of ξ. Suppose we want the element 3 in ξ, i.e. the element in row 1 and column 0. Then, in row major order the element is stored in $@\xi+3$ and in column major order the element is stored in $@\xi+1$.

The following two mapping functions are based on row major order. In the following definition, $\hat{\alpha}$ is a shape vector, and $\hat{i}$ is a full index.

Definition 4.14 *The partial function* **Index–to–Offset$_r$**, *denoted by* $\boldsymbol{\gamma_r}$, *has domain* $\mathcal{SEQ}(\mathcal{N}) \times \mathcal{SEQ}(\mathcal{N})$ *and range* $\mathcal{N}$. *Let* $\hat{i} = < i_0 \ \ldots \ i_{m-1}>$ *and* $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}> \ \in \mathcal{SEQ}(\mathcal{N})$, *where* $\hat{i}$ *and* $\hat{\alpha}$ *satisfy the constraint that* $\hat{i} \ \in Selectors(\hat{\alpha})$. *Then,* $\gamma_r(\hat{i},\ \hat{\alpha})$ *is defined as follows.*

$$\gamma_r(\hat{\Theta},\hat{\Theta}) \ = \ 0 \ \textit{if}\ m=0,$$

$$\gamma_r(\hat{i},\hat{\alpha}) = \ i_{m-1} + (\alpha_{m-1} * \gamma_r(< i_0 \ \ldots \ i_{m-2}>, <\alpha_0 \ \ldots \ \alpha_{m-2}>)) \ \textit{if}\ m>0.$$

□

Definition 4.15 *The partial function* **Offset–to–Index$_r$**, *denoted by* $\boldsymbol{\gamma_r'}$, *has domain* $\mathcal{N} \times \mathcal{SEQ}(\mathcal{N})$ *and range* $\mathcal{SEQ}(\mathcal{N})$. *Let* $q \in \mathcal{N}$ *and* $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}> \ \in \mathcal{SEQ}(\mathcal{N})$, *where* q *and* $\hat{\alpha}$ *satisfy the constraint that* $q < \pi(\hat{\alpha})$. *Then,* $\gamma_r'(q,\hat{\alpha})$ *is defined as follows.*

$$\gamma_r'(0,\hat{\Theta}) \ = \hat{\Theta} \ \textit{if}\ m=0,$$

$$\gamma_r'(q,\hat{\alpha}) = \ \gamma_r'(q \ div \ \alpha_{m-1}, <\alpha_0 \ \ldots \ \alpha_{m-2}>) \ +\!\!+_s < q \ mod \ \alpha_{m-1} > \textit{if}\ m>0.$$

□

Suppose that in Definition 4.15, $m = 1$, so that $\hat{\alpha} = <\alpha_0>$. Then the constraint on q and $\hat{\alpha}$ is that $q < \alpha_0$. Therefore, $q \ mod \ \alpha_0 = q$. Furthermore, $q \ div \ \alpha_0 = 0$ and $<\alpha_0 \ \ldots \ \alpha_{m-2}> \ = \hat{\Theta}$. Therefore $\gamma_r^{'}(q \ div \ \alpha_{m-1}, <\alpha_0 \ \ldots \ \alpha_{m-2}>) = \gamma_r^{'}(0, \hat{\Theta}) \ = \hat{\Theta}$. Hence, from Observation 4.5,

$$\gamma_r^{'}(q, <\alpha_0>) \ = \ <q>.$$

The following two mapping functions are based on column major order.

Definition 4.16 *The partial function* **Index–to–Offset$_c$**, *denoted by* $\boldsymbol{\gamma_c}$, *has domain* $\mathcal{SEQ}(\mathcal{N}) \times \mathcal{SEQ}(\mathcal{N})$ *and range* $\mathcal{N}$. *Let* $\hat{i} = < i_0 \ \ldots \ i_{m-1}>$ *and* $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}> \ \in \mathcal{SEQ}(\mathcal{N})$, *where* $\hat{i}$ *and* $\hat{\alpha}$ *satisfy the constraint that* $\hat{i} \ \in Selectors(\hat{\alpha})$. *Then,* $\gamma_c(\hat{i}, \ \hat{\alpha})$ *is defined as follows.*

$$\gamma_c(\hat{\Theta}, \hat{\Theta}) \ = \ 0 \ \text{if} \ m > 0,$$

$$\gamma_c(\hat{i}, \hat{\alpha}) = \ i_0 + (\alpha_0 * \gamma_c(< i_1 \ \ldots \ i_{m-1}>, <\alpha_1 \ \ldots \ \alpha_{m-1}>)) \ \text{if} \ m > 0.$$

□

Definition 4.17 *The partial function* **Offset–to–Index$_c$**, *denoted by* $\boldsymbol{\gamma_c^{'}}$, *has domain* $\mathcal{N} \times \mathcal{SEQ}(\mathcal{N})$ *and range* $\mathcal{SEQ}(\mathcal{N})$. *Let* $q \in \mathcal{N}$ *and* $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}> \ \in \mathcal{SEQ}(\mathcal{N})$, *where* q *and* $\hat{\alpha}$ *satisfy the constraint that* $q < \pi(\hat{\alpha})$. *Then,* $\gamma_c^{'}(q, \hat{\alpha})$ *is defined as follows.*

$$\gamma_c^{'}(0, \hat{\Theta}) \ = \hat{\Theta} \ \text{if} \ m = 0,$$

$$\gamma_c^{'}(q, \hat{\alpha}) = \ < q \ mod \ \alpha_0 > \ +\!\!+_s \ \gamma_c^{'}(q \ div \ \alpha_0, <\alpha_1 \ \ldots \ \alpha_{m-1}>) \ \text{if} \ m > 0.$$

□

Suppose that in Definition 4.17, $m = 1$, so that $\hat{\alpha} = <\alpha_0>$. Then

$$\gamma_c^{'}(q, <\alpha_0>) \ = \ <q>.$$

Observation 4.19 *For* $\hat{\alpha} \ \in \mathcal{SEQ}(\mathcal{N})$ *and* $\hat{i} \ \in Selectors(\hat{i})$,

$$\gamma_r^{'}(\gamma_r(\hat{i}, \hat{\alpha}), \hat{\alpha}) = \hat{i} \ \text{and}$$

$$\gamma_c^{'}(\gamma_c(\hat{i}, \hat{\alpha}), \hat{\alpha}) = \hat{i}.$$

□

Observation 4.20 *For* $\hat{\alpha} \ \in \mathcal{SEQ}(\mathcal{N})$ *and* $n \ \in \ \mathcal{N}$ *such that* $n < \pi\,(\hat{\alpha})$,

$$\gamma_r(\gamma_r^{'}(n, \hat{\alpha}), \hat{\alpha}) = n \ \text{and}$$

$$\gamma_c(\gamma_c^{'}(n, \hat{\alpha}), \hat{\alpha}) = n.$$

□

Consider the array ξ from Example 4.3. $\gamma_r(<1\ 0>, <2\ 3>) = 0 + (3 * 1) = 3$, indicating that for row major order, the $<1\ 0>$ element of ξ is allocated position $@\xi \ + \ 3$.

$\gamma_c(<1\ 0>, <2\ 3>) = 1 + (2 * 0) = 1$, indicating that for column major order, the $<1\ 0>$ element of ξ is allocated position $@\xi \ + \ 1$.

4.7 Transformation Functions Based on FORTRAN Intrinsics

4.7.1 Transpose

Definition 4.18 *A sequence $\hat{\beta}$ in $\mathcal{SEQ}(\mathcal{N})$ is a* **permutation sequence** *for $m \in \mathcal{N}$ if $\hat{\beta} = <\beta_0 \ \ldots \ \beta_{m-1}>$ where $0 \leq \beta_i < m$ for $0 \leq i < m$, and $\beta_i = \beta_j \Rightarrow i = j$ for $0 \leq i, j < m$.* □

In the following definition of *Transpose*$(\hat{\beta}, \xi)$, ξ is the array being transposed, and $\hat{\beta}$ is a permutation sequence specifying how ξ is to be transposed.

Definition 4.19 *The partial function* **Transpose**, *has domain $\mathcal{SEQ}(\mathcal{N}) \times \mathcal{ARRAY}(\mathcal{S})$ and range $\mathcal{ARRAY}(\mathcal{S})$. Let $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$, where $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}>$. Let $\hat{\beta} = <\beta_0 \ \ldots \ \beta_{m-1}> \ \in \mathcal{SEQ}(\mathcal{N})$, where $\hat{\beta}$ satisfies the constraint that $\hat{\beta}$ is a permutation sequence for m. Then Transpose$(\hat{\beta}, \xi) = (\hat{\alpha}', \psi')$ is defined as follows.*

(a) $\hat{\alpha}' = <\alpha'_0 \ \ldots \ \alpha'_{m-1}>$ where $\alpha'_j = \alpha_k$ for that value k such that $\beta_k = j$.

(b) for each $\hat{i} = <i_0 \ i_1 \ \ldots \ i_{m-1}> \ \in$ Selectors$(\hat{\alpha}')$,

$$\psi'(\hat{i}) = \psi(<i_{\beta_0} \ i_{\beta_1} \ \ldots \ i_{\beta_{m-1}}>).$$

□

For instance, suppose that $\xi = (\hat{\alpha}, \psi)$ is an array where $\hat{\alpha} = <\alpha_0 \ \alpha_1 \ \alpha_2> = <30 \ 40 \ 50>$. Let $\hat{\beta} = <\beta_0 \ \beta_1 \ \beta_2> = <2 \ 0 \ 1>$. Let $\xi' = (\hat{\alpha}', \psi') =$ *Transpose*$(\hat{\beta}, \xi)$. Let $\hat{\alpha}' = <\alpha'_0 \ \alpha'_1 \ \alpha'_2>$. First consider α'_0. Since $\beta'_1 = 0$, $\alpha'_0 = \alpha_1 = 40$. For α'_1, since $\beta'_2 = 1$, $\alpha'_1 = \alpha_2 = 50$. For α'_2, since $\beta'_0 = 2$, $\alpha'_2 = \alpha_0 = 30$. Thus,

$$\hat{\alpha}' = <\alpha_1 \ \alpha_2 \ \alpha_0> = <40 \ 50 \ 30>.$$

Furthermore,

$$\psi'(< i_0 \ i_1 \ i_2>) = \psi(< i_2 \ i_0 \ i_1>).$$

For instance, $\psi'(<37 \ 45 \ 8>) = \psi(<8 \ 37 \ 45>)$.

As another example, suppose that $\xi = (\hat{\alpha}, \psi)$ is an array where $\hat{\alpha} = <\alpha_0 \ \alpha_1 \ \alpha_2 \ \alpha_3> = <30 \ 40 \ 50 \ 60>$. Let $\hat{\beta} = <\beta_0 \ \beta_1 \ \beta_2 \ \beta_3> = <0 \ 3 \ 1 \ 2>$. Let $\xi' = (\hat{\alpha}', \psi') =$ *Transpose*$(\hat{\beta}, \xi)$. Then,

$$\hat{\alpha}' = <\alpha_0 \ \alpha_2 \ \alpha_3 \ \alpha_1> = <30 \ 50 \ 60 \ 40>.$$

Furthermore,

$$\psi'(< i_0 \ i_1 \ i_2 \ i_3 >) = \psi(< i_0 \ i_3 \ i_1 \ i_2 >).$$

Note that the relationship between $\hat{\alpha}$ and $\hat{\alpha}'$ in Definition 4.19(a) can be restated as $\alpha_k = \alpha'_{\beta_k}$ for $0 \leq k < m$.

4.7.2 Reshape

In the following definition of $\widehat{\rho}_{\lambda}(\xi, \hat{\beta}, f)$, ξ is the array being reshaped, $\hat{\beta}$ is the new shape sequence, and f is a fill–in value to be used if the new array has more elements than the old array.

Definition 4.20 *Let λ be a layout method for which functions γ_λ and $\gamma_\lambda^{'}$ are defined. The function* **Reshape$_{\boldsymbol{\lambda}}$**, *denoted by* $\boldsymbol{\widehat{\rho}_{\lambda}}$, *has domain $\mathcal{ARRAY}(\mathcal{S}) \times \mathcal{SEQ}(\mathcal{N}) \times \mathcal{S}$ and range $\mathcal{ARRAY}(\mathcal{S})$. Let $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$. Let $\hat{\beta} = \in \mathcal{SEQ}(\mathcal{N})$. Let $f \in \mathcal{S}$. Then $\widehat{\rho}_{\lambda}(\xi, \hat{\beta}, f) = (\hat{\beta}, \psi')$ where for each $\hat{i} \in Selectors(\hat{\beta})$,*

$$\psi'(\hat{i}) = \begin{cases} \psi(\gamma_\lambda^{'}(\gamma_\lambda(\hat{i}, \hat{\beta}), \hat{\alpha})) & \text{if } \gamma_\lambda(\hat{i}, \hat{\beta}) < \tau(\xi) \\ f & \text{if } \gamma_\lambda(\hat{i}, \hat{\beta}) \geq \tau(\xi). \end{cases}$$

□

4.7.3 Cshift

In the following definition of $Cshift(\xi, a, q)$, ξ is the array being shifted, a is the amount of the shift, and q is the dimension to be shifted. *Cshift* does a circular left shift if a is positive, and a circular right shift if a is negative.

Definition 4.21 *The partial function* **Cshift** *has domain $\mathcal{ARRAY}(\mathcal{S}) \times \mathcal{Z} \times \mathcal{N}$ and range $\mathcal{ARRAY}(\mathcal{S})$. Let $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$, where $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}>$. Let $a \in \mathcal{Z}$. Let $q \in \mathcal{N}$, where q satisfies the constraint that $q < m$. Then $Cshift(\xi, a, q) = (\hat{\alpha}, \psi')$, where ψ' is defined as follows. Let $\hat{k} = <k_0 \ k_1 \ \ldots \ k_{m-1}> \in Selectors(\hat{\alpha})$. For $0 \leq i < m$, let*

$$k_i^{'} = \begin{cases} k_i & \text{if } i \neq q \\ (k_q + a) \ mod \ \alpha_q & \text{if } i = q. \end{cases}$$

Then

$$\psi'(\hat{k}) = \psi(<k_0^{'} \ k_1^{'} \ \ldots \ k_{m-1}^{'}>).$$

□

For example,

$$Cshift(\prec 20 \ 21 \ 22 \ 23 \ 24 \ 25 \succ, 2, 0) = \prec 22 \ 23 \ 24 \ 25 \ 20 \ 21 \succ.$$

$$Cshift(\prec 20 \ 21 \ 22 \ 23 \ 24 \ 25 \succ, -2, 0) = \prec 24 \ 25 \ 20 \ 21 \ 22 \ 23 \succ.$$

Observation 4.21 *For $\hat{\alpha} \in \mathcal{VEC}(\mathcal{S})$, let $m = \tau(\hat{\alpha})$, and let $k \in \mathcal{N}$ be such that $0 \leq k < m$. Then*

$$\begin{aligned} Cshift(\hat{\alpha}, k, 0) &= \Xi_{\mathcal{S}}^{-1}(Take(\Xi(\hat{\alpha}), k, m-1) +\!\!+_{\boldsymbol{s}} Take(\Xi(\hat{\alpha}), 0, k-1)) \\ Cshift(\hat{\alpha}, -k, 0) &= \Xi_{\mathcal{S}}^{-1}(Take(\Xi(\hat{\alpha}), m-k+1, m-1) +\!\!+_{\boldsymbol{s}} Take(\Xi(\hat{\alpha}), 0, m-k)) \end{aligned}$$

□

4.7.4 EOSshift

In the following definition of *EOSshift*(ξ, a, q, f), ξ is the array being shifted, a is the amount of the shift, q is the dimension to be shifted, and f is a fill–in value to be used for array positions vacated by the shift. *EOSshift* does a noncircular left shift if a is positive, and a noncircular right shift if a is negative.

Definition 4.22 *The partial function* **EOSshift** *has domain* $\mathcal{ARRAY}(\mathcal{S}) \times \mathcal{Z} \times \mathcal{N} \times \mathcal{S}$ *and range* $\mathcal{ARRAY}(\mathcal{S})$*. Let* $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$*, where* $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{m-1}>$*. Let* $a \in \mathcal{Z}$*. Let* $q \in \mathcal{N}$*, where* q *satisfies the constraint that* $q < m$*. Let* $f \in \mathcal{S}$*. Then* *EOSshift*$(\xi, a, q, f) = (\hat{\alpha}, \psi')$*, where* ψ' *is defined as follows. Let* $\hat{k} = <k_0 \ k_1 \ \ldots \ k_{m-1}> \in$ *Selectors*$(\hat{\alpha})$*.*

(a) Suppose $(a \geq 0$ *and* $k_q < \alpha_q - a)$ *or* $(a < 0$ *and* $k_q \geq -a)$*. For* $0 \leq i < m$*, let*

$$k_i' = \begin{cases} k_i & \text{if } i \neq q \\ (k_q + a) & \text{if } i = q. \end{cases}$$

Then

$$\psi'(\hat{k}) = \psi(<k_0' \ k_1' \ \ldots \ k_{m-1}'>).$$

(b) Suppose $(a \geq 0$ *and* $k_q \geq \alpha_q - a)$ *or* $(a < 0$ *and* $k_q < -a)$*. Then*

$$\psi'(\hat{k}) = f.$$

$\square$

For example,

$$\mathit{EOSshift}(\prec 20 \ 21 \ 22 \ 23 \ 24 \ 25 \succ, 2, 0, 8) = \prec 22 \ 23 \ 24 \ 25 \ 8 \ 8 \succ.$$

$$\mathit{EOSshift}(\prec 20 \ 21 \ 22 \ 23 \ 24 \ 25 \succ, -2, 0, 8) = \prec 8 \ 8 \ 20 \ 21 \ 22 \ 23 \succ.$$

4.8 Array Sections

Definition 4.23 *Given set* $\mathcal{S}$*, we let* $\boldsymbol{\mathcal{S}}^*$ *denote the set* $\mathcal{S} \cup \{*\}$*.* $\square$

Definition 4.24 *The partial function* **Index–Via–Sequence****, denoted by* $\mathbf{\Psi}^*$*, has domain* $\mathcal{SEQ}(\mathcal{N}^*) \times \mathcal{ARRAY}(\mathcal{S})$ *and range* $\mathcal{ARRAY}(\mathcal{S})$*. Let* $\xi = (\hat{\alpha}, \psi) \in \mathcal{ARRAY}(\mathcal{S})$*, where* $\hat{\alpha} = <\alpha_0 \ \ldots \ \alpha_{n-1}>$*. Let* $\hat{i} = <i_0 \ \ldots \ i_{n-1}> \ \in \mathcal{SEQ}(\mathcal{N}^*)$*, where* $\hat{i}$ *satisfies the constraint that for* $0 \leq j < n$*, either* α_j *is* $*$*, or* $\alpha_j \in \mathcal{N}$ *and* $i_j < \alpha_j$*. Let* p *equal the number of occurrences of* $*$ *in* $\hat{i}$*. Let* f *be the injective function from* $\{0, \ 1, \ \ldots \ p{-}1\}$ *to* $\{0, \ 1, \ \ldots \ n{-}1\}$*, defined by*

$$f(j) = k \mid i_k = * \text{ and } Take(\hat{i}, 0, k) \text{contains exactly } k+1 \text{ occurrences of } * .$$

Then $\hat{i}\mathbf{\Psi}^*\xi = (\hat{\alpha}', \psi')$ *as follows.*

(a) $\hat{\alpha}' = \ <\alpha'_0 \ \ldots \ \alpha'_{p-1}>$ *where* $\alpha'_j = \alpha_{f(j)}$ *for* $0 \leq j < p$.

(b) Given $\hat{\beta} = <\beta_0 \ \ldots \ \beta_{p-1}> \in$ *Selectors*$(\hat{\alpha}')$, *let* $\hat{\beta}' = <\beta'_0 \ \ldots \ \beta'_{n-1}>$ *where*

$$\beta'_j = \begin{cases} i_j & \text{if } i_j \in \mathcal{N} \\ \beta_{f^{-1}(j)} & \text{if } i_j = *. \end{cases}$$

Then, $\psi'(\hat{\beta}) = \psi(\hat{\beta}')$. □

For example, consider the three–dimensional array from Example 4.1.

$$\xi = (<2\ 3\ 4>, \psi) = \begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \\ \\ \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}.$$

$$<1\ *\ *>\ \mathbf{\Psi}^*\ \xi \ = \ \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix}.$$

$$<*\ 1\ *>\ \mathbf{\Psi}^*\ \xi \ = \ \begin{bmatrix} 4 & 5 & 6 & 7 \\ 16 & 17 & 18 & 19 \end{bmatrix}.$$

$$<*\ *\ 2>\ \mathbf{\Psi}^*\ \xi \ = \ \begin{bmatrix} 2 & 6 & 10 \\ 14 & 18 & 22 \end{bmatrix}.$$

$\mathbf{\Psi}^*$ can be expressed using *Transpose* and $\mathbf{\Psi}$. For instance, for a five–dimensional array ξ,

$$<9\ *\ 6\ 7\ *>\ \mathbf{\Psi}^*\ \xi \ = \ <9\ 6\ 7>\ \mathbf{\Psi}\ \mathit{Transpose}(<0\ 3\ 1\ 2\ 4>, \xi).$$

As another example, for an eight–dimensional array ξ,

$$<*\ 5\ *\ *\ 2\ *\ 8\ *>\ \mathbf{\Psi}^*\ \xi \ = \ <5\ 2\ 8>\ \mathbf{\Psi}\ \mathit{Transpose}(<3\ 0\ 4\ 5\ 1\ 6\ 2\ 7>, \xi).$$

Observation 4.22 *Let* $\hat{i} = <i_0 \ \ldots \ i_{n-1}> \in \mathcal{SEQ}(\mathcal{N}^*)$ *and* $\xi \in \mathcal{ARRAY}(\mathcal{S})$ *be such that* $\hat{i}\mathbf{\Psi}^*\xi$ *is defined. Let* p *equal the number of occurrences of* $*$ *in* $\hat{i}$. *Let* $\hat{i}'$ *be* $\hat{i}$ *with all occurrences of* $*$ *deleted (so that* $\tau(\hat{i}') = n - p$*). Let* $\hat{\beta} = <\beta_0 \ \ldots \ \beta_{n-1}> \in \mathcal{SEQ}(\mathcal{N})$ *be as follows.*

$$\beta_j = \begin{cases} j - (\text{number of } * \text{'s in } \mathit{Take}(\hat{i}, 0, j) & \text{if } i_j \in \mathcal{N} \\ p + j & \text{if } i_j = *. \end{cases}$$

Then,

$$\hat{i}\mathbf{\Psi}^*\xi \ = \ \hat{i}'\ \mathbf{\Psi}\ \mathit{Transpose}(\hat{\beta}, \xi).$$

□

Acknowledement

We acknowledge Richard E. Stearns for a number of helpful discussions and suggestions for developing this formalism, and Shi–Yu Chen for preparing an early version of this manuscript.

References

[1] L. Mullin et al. The pgi-psi project: Preprocessing optimizations for existing and new f90 intrinsics in hpf using compositional symmetric indexing of the psi calculus. In M. Gerndt, editor, *Proceedings of the 6th Workshop on Compilers for Parallel Computers*. Forschungszentrum Julich GmbH, 1996.

[2] High Performance Fortran Forum. *High Performance Fortran Language Specification, Version 1*, May 1993.

[3] L. Mullin. The psi compiler project. In *Workshop on Compilers for Parallel Computers*. TU Delft, Holland, 1993.

[4] L. M. R. Mullin. *A Mathematics of Arrays*. PhD thesis, Syracuse University, December 1988.

[5] L.R. Mullin, D. Dooling, E. Sandberg, and S. Thibault. Formal methods for scheduling and communication protocol. In *Proceedings of the Second International Symposium on High Performance Distributed Computing(HPDC-2)*. IEEE Computer Society, July 1993.