Specification and Verification of a Wavefront Algorithm for the Algebraic Path Problem

Stephen B. Seidman
Department of Computer Science and Engineering
Auburn University
Auburn, AL 36849

Abstract

In recent papers, Rice and Seidman have proposed a model of wavefront computation on a general interconnection network and used it to develop a language for specifying wavefront algorithms on an *n*-dimensional grid of processors. In this paper, the semantics for the language is used to specify and verify a wavefront algorithm for the algebraic path problem.

MR Subject Classification Numbers: 68Q22, 68Q60, 65Y05

Key Words and Phrases: algebraic path problem, wavefront algorithm, systolic algorithm, verification of parallel algorithms

I. Introduction

In a seminal paper [1], Kung et al proposed that *wavefront computation* could be used to specify and implement parallel algorithms for the mathematical computations needed for high-performance signal processing. In [1], a wavefront computation is regarded as executing in a data-driven fashion on a two-dimensional grid of computational cells that have a north-south-east-west interconnection network. Computations start in the upper left corner of the grid and proceed diagonally toward the lower right corner. The arrival of the data in a wavefront at a cell triggers the computation in that cell, which in turn generates the data needed to propagate the wavefront. Rice and Seidman [3] propose a generalized model of wavefront computation based on an interconnection network that is an *n*-dimensional array of cells and also introduce syntax and semantics for a wavefront language. Rice has recently [2] used methods and ideas from group theory and topology to extend the wavefront model to a broad class of interconnection networks. In this paper, the semantics for the Rice-Seidman *n*-dimensional wavefront language will be discussed in the context of Rice´s general model of wavefront computation. The semantics will then be used to give a formal verification of a wavefront algorithm for the algebraic path problem.

II. Semantics for Wavefront Computation on an n-Dimensional Grid

Assume that, where is the cyclic group on t elements, and that , where s_i is the unit vector whose only nonzero entry is a 1 in dimension i. The d-dimensional grid of computational elements then corresponds to the Cayley graph (G, S). Each element s_i of S corresponds to a boundary set of elements [2] whose ith coordinate is zero. The set S forms a wavefront family [2], and the intersection of the corresponding boundary sets is the cell $D_0 = \{0,...0\}$. A wavefront computation starts at the cell D_0, and external input can be provided by cells in boundary sets corresponding to members of S. In a similar fashion, the cells in boundary sets corresponding to inverses of members of S can provide external output. Each cell can perform an initial *pure cell computation* during which it does not communicate with other cells. Furthermore, the computations performed by each cell may differ, based on the cell´s geometric position or the number of wavefronts that it has processed. All computations within individual cells are performed sequentially. Finally, cell variables can be initialized by values from the environment and the final values of cell variables can be stored in the environment at the conclusion of the wavefront computation.

In the wavefront language [3], the specification of an algorithm consists of a *declaration*, which describes the cell computation, and an *instantiation*, which describes the dimensions and external interface of the computational grid. If the computational grid is described by the instantiation

NETWORK($e_1 : f_1, e_2 : f_2, ..., e_d : f_d$) **OF**
CELL (<External_ports>)
FOR NUMWAVES = g,

define $NumWaves = E(g)$, $m_i = E(e_i)$, and $n_i = E(f_i)$, $1 \le i \le d$. Let **N** and **Z** denote the natural numbers and integers, respectively. We assume that $\{m_i\} \cup \{n_i\} \cup \{NumWaves\}$ y **Z**, $m_i \le n_i$ for all i, and $NumWaves \ge 0$. *NumWaves* represents the number of waves of data processed by each cell during the wavefront computation. If $NumWaves = 0$, no wavefront computation is performed. The integers $\{m_i\}$ and $\{n_i\}$ represent the lower and upper bounds, respectively, of the cell coordinates, and $n_i - m_i + 1 = t_i$.

The semantic rules are based on two expression evaluators which specify the values of global (known in the external environment) and local (cell) expressions. The first of these is E : **GlobalExp** $\to$ **Values**$_{\mathrm{P}}$,where **GlobalExp** denotes the expressions used in the instantiation and the flat domain **Values**$_{\mathrm{P}}$ denotes the set of storable values (including integers and reals) augmented with the undefined element P. The second expression evaluator is [ ] : **LocalExp** $\to$ **Indices** $-\!\!\!\mid\!\!\!\rightarrow$ **Stages** $-\!\!\!\mid\!\!\!\rightarrow$ **Values**$_{\mathrm{P}}$, where **LocalExp** denotes the expressions used in cell declarations, **Stages** = **N** denotes the possible stages of a computation, and **Indices** = $\mathbf{Z}^d$ denotes the possible cell coordinates.

Define $$\mathbf{P} = \ni\{[m_i, n_i] : 1 \le i \le d\}.$$

Given $e \in$ **LocalExp**, the domain of [e] is **P** and for each $p \in$ **P**, the domain of [e]p is [0,*NumWaves*]. In addition,

- [e]ps denotes the *initial value* of e in cell p at the beginning of wavefront s

- $[\, e' \,]ps$ denotes the *final value* of e in cell p after wavefront s has been processed.

If $s = 0$, the preceding values refer to the values preceding and following the cell computation performed prior to the first wavefront, while if $s > 0$, the values refer to the wavefront computation.

The movement of data between the cells and the external environment involves both *external* and *boundary* ports. These ports correspond to d-dimensional *cell* data and (d-1)-dimensional *boundary* data, respectively. An *external* port is a partial mapping **Indices** $\longrightarrow\!\!\!\!\!\!\!+\;\;$ **Values** which has domain **P** and thus corresponds to a variable that has a value in each cell. The variable corresponding to an external port will either be initialized prior to any computation or will have its final value stored in the environment after the completion of the wavefront computation. A *boundary port* is a partial mapping $\mathbf{Z}^{d-1} \rightarrow\!\!\!\!\!\!+\; \mathbf{N} \rightarrow\!\!\!\!\!\!+\; \mathbf{Values}$ which specifies a stream of data values for each cell in some boundary set of **P**. Boundary ports can be distinguished as *input ports* and *output ports*. There is one input port and one output port for each $1 \leq i \leq d$. These ports have domain $\mathbf{P}_i = \ni\{[m_j, n_j]: 1 \leq j \leq d \wedge j \neq i\}$,
and correspond to the following boundary subsets of **P**:

$$\mathbf{IN}_i = \{m_i\} \times \mathbf{P}_i \qquad \text{(corresponding to } s_i\text{)}$$
$$\mathbf{OUT}_i = \{n_i\} \times \mathbf{P}_i \qquad \text{(corresponding to } s_i^{-1}\text{)}$$

For convenience in stating the semantic rules, we will assume that **Indices** = **P** and **Stages** = [0,*NumWaves*]. The semantics of the read-only variables *self* and *stage* are specified as follows. For every $p \in$ **Indices** and $s \in$ **Stages**,

(0)$_\mathbf{i}$ $\quad [\, \mathbf{self}[i] \,]ps = p_i \qquad 1 \leq i \leq d$

(0)$_\mathbf{ii}$ $\quad [\, \mathbf{stage} \,]ps = s.$

The variables **self** and **stage** can be used in conditional statements to express space and time heterogeneity of a cell computation. For example, the statement

COND (*self*[1] < *self*[2]) **->** **{** $c := c + 2$ **}**, (*self*[1] >= *self*[2]) **->** **SKIP**

specifies *space heterogeneity*; when it is executed, the value of c is incremented by 2 in each cell whose first coordinate is less than its second coordinate. In a similar manner,

COND (*stage* = 1) **->** **{** $z := w$**}**, (*stage* > 1) **->** **{** $z, w := w, z$ **}**

specifies *time heterogeneity*; the value of w is assigned to z when the first wavefront is processed and the values of z and w are interchanged when the second and succeeding wavefronts are processed.

Based on a specification

CELL(IMPORT $i_1, i_2, ..., i_u$ **EXPORT** $e_1, e_2, ..., e_v$),

found in the instantiation and a specification

$$\textbf{CELL(IMPORT } g_1, g_2, ..., g_r \textbf{ EXPORT } h_1, h_2, ..., h_s\textbf{)},$$

found in the cell declaration, we assume that $r = u$, $s = v$, and for every $p \in$ **Indices**,

(1)$_i$ $[\, g_m \,]p\, 0 = i_m(p) \qquad 1 \le m \le r$

(1)$_{ii}$ $e_n(p) = [\, h_n' \,](p) NumWaves \qquad 1 \le n \le s.$

Equation $(1)_i$ says that when the pure cell computation is initiated, the variable g_m has the values of the external port i_m. Similarly, equation $(1)_{ii}$ says that at the completion of the wavefront computation, the external port e_n has the final values of the variable h_n.

The communication specification

? *in-list* <Seq_comp> **!** *out-list*

declares two array identifiers *inlist* and *out-list* of size d. Assume that $exp \in$ **LocalExp** does not use any of the references $\{inlist[i]\}$. For every $p \in$ **Indices** and $s \in [1, NumWaves]$,

(2) $[\, exp \,]ps = [\, exp' \,]p(s\text{ -}1).$

Equation (2) states that the wavefront computation in each cell is performed in a sequential manner. The initial value of an identifier that does not refer to *in-list* in a cell at a particular wavefront is the final value of the identifier in the same cell with respect to the preceding wavefront.

For each $1 \le i \le d$, $p \in \mathbf{P}^{(i)} = [m_i + 1, n_i] \times \mathbf{P}_i$, and $s \in [1, NumWaves]$,

(3) $[\, in\text{-}list[i] \,]ps = [\, out\text{-}list[i]' \,]p^{(i)}s,$

where $p^{(i)} = p \oplus \{i \text{ o } p_i \text{ - } 1\}$.

Equation (3) states that the initial value of *in-list*[i] in a non-input boundary cell is the final value of *out-list*[i] in the adjacent cell in dimension i.

Assume now that the corresponding instantiation involves the syntactic construct

USING b, c

This construct specifies two lists of boundary ports: $\{b_i\}$ and $\{c_i\}$. Data movement for the cells in the corresponding boundary sets is then specified as follows: for each $1 \le i \le d$,

(4)$_i$ $[\, in\text{-}list[i] \,]p = b_i(p^i) \qquad (p \in \mathbf{IN}_i)$
(4)$_{ii}$ $c_i(q^i) = [\, out\text{-}list[i]' \,]q \qquad (q \in \mathbf{OUT}_i),$

where $\{p^i, q^i\}$ y $\mathbf{P}_i$ are defined by $p = p^i \oplus \{i \text{ o } m_i\}$ and $q = q^i \oplus \{i \text{ o } n_i\}$, respectively.

Equations $(4)_{i\text{-}ii}$ state that the values of *in-list*[i] are read from the input boundary port b_i and the output boundary port c_i reads its values from *out-list*[i].

It is also possible to communicate on only a subset of the grid dimensions. To do this, the syntactic form

$$\textbf{?}\ [X_1, X_2, ..., X_k]\ \text{<Seq\_comp>}\ \textbf{!}\ [A_1, A_2, ..., A_k]$$

must be used, along with the instantiation construct

USING {(**IN_PORT | OUT_PORT**) (<Dim>, <Id>)}.

Each IN_PORT specification for a given dimension must be matched by an OUT_PORT specification for the same dimension. Each dimension can be used in at most one pair of port specifications. The number of port specifications must be the same as the length of the input and output lists in the communication specification, and the association of ports with identifiers is positional. In this situation, each cell will receive the values of the variables $\{X_i\}$ and will send the values of the variables $\{A_i\}$ along the specified dimensions. There will be no data movement with respect to dimensions other than the ones specified.

The corresponding semantic equations for $m \in \{1,..., k\}$ are

(3) $[\, X_m\,]ps = [\, A_m'\,]p^{(m)}s \qquad (p \in \mathbf{P}^{(m)})$

(4) $[\, X_m\,]p = b_m(p^m) \qquad (p \in \mathbf{IN}_m)$
$c_m(q^m) = [\, A_m'\,]q \qquad (q \in \mathbf{OUT}_m).$

III. A Wavefront Algorithm for the Algebraic Path Problem

In 1981, Zimmermann unified a large number of combinatorial and algebraic problems by showing that they were special cases of a single problem defined on graphs whose edges were weighted with values chosen from a semiring [6]. Zimmermann called such problems algebraic path problems; any algorithm to solve general algebraic path problems can also be used to solve the shortest path problem in graphs, to compute the transitive closure of a binary relation, or to perform Gauss-Jordan elimination on a matrix with values in any field.

The algebraic path problem (APP) is defined as follows: given a weighted graph $G=(V,E, w)$, where V is a finite vertex set, E an arc set, and w is a weight function from E to a semiring $(H, \oplus, \otimes)$ with zero and unity, compute the quantities $d_{ij}=\oplus\{w(p)|\ p \in M(i,j)\}$, where M(i,j) denotes the set of all paths from i to j and the weight of a path p is defined by $w(p) = \oplus\{w(e)|\ e \in p\}$. Suppose that $|V| = N$; the $N\times N$ weight matrix $A = A(i,j)$ is defined by setting $A(i,j) = w(i,j)$ if $(i,j)\in E$ and $A(i,j)= 0$ otherwise. The set of all paths from i to j which contain only vertices p with $1\leq p \leq k$ is denoted by M(i,j,k); we define

$$A(i,j,k) = \oplus\, \{w(p)|\ p\in M(i,j,k)\},$$

where A(i,j,N) is the value d_{ij} that is to be determined.

As observed by Rote [5], the {A(i,j,k)} can be computed using the following recurrences:

(R0) $A(i, j, 0) = A(i, j)$
(R1) $A(i, j, k) = A(i, j, k-1) \oplus (A(i, k, k) \otimes A(k, j, k-1))$ for $1 \leq k \leq N$ and $k \neq i$, $k \neq j$
(R2a) $A(i, j, i) = A(i, i, i) \otimes A(i, j, i-1)$ for $i \neq j$
(R2b) $A(i, j, j) = A(i, j, j-1) \otimes A(j, j, j)$ for $i \neq j$
(R3) $A(i, i, i) = A(i, i, i-1)^*$ for $1 \leq i \leq N$

where $c^* = \oplus \{c^i \mid i \geq 0\}$. The APP can only be solved for a particular semiring if this infinite sum can be defined.

The central role played by this problem has led many investigators to examine sequential and parallel algorithms for the APP. Systolic algorithms have been proposed by Rote [5], and by Robert and Trystram [4]. In this paper, a careful wavefront specification and verification for the Robert-Trystram systolic algorithm will be given. The discussion will be restricted to the case in which the semiring is the ring of real numbers; in this case, the operations ⊕ and ⊗ become ordinary addition and multiplication, and $c^*=(1-c)^{-1}$ if $|c| < 1$. Note that in this case, the APP algorithm can be used to compute the inverse of a nonsingular matrix (more precisely, this is the case if the matrix IA is input to the array). Rote observes that even though the infinite sum that defines c^* does not converge for $|c| \geq 1$, the use of the expression $c^*=(1c)^1$ in the APP algorithm allows the computation of the inverse for all nonsingular matrices.

The systolic APP algorithm presented by Robert and Trystram in [4] executes on a trapezoidal subset of a rectangular grid. Although each computational element is prepared to accept two inputs and produce two outputs at each clock tick, the elements´ communication behavior is not homogeneous in time. Specifically, at each element´s first execution, two data values are received but only one is output. For this reason, while the algorithm inputs 2N waves of data to process a $N \times N$ matrix, it only produces N waves of output data. Robert and Trystram outline an argument for the correctness of their algorithm ([4], pp. 196-7), but in the absence of a formal model of systolic computation, their argument does not constitute a proof.

In this paper, we will specify and verify a wavefront algorithm for the APP that has been modified from the systolic algorithm of Robert and Trystram to fit within the model of wavefront computation described above. In order to solve the APP for a $N \times N$ matrix, the wavefront algorithm is executed on a rectangular $N \times 2N$ grid of computational elements. There are 2N waves of input and output data. The ports **NorthInput** and **WestInput** supply input data to the North and West edges of the grid, respectively. The left half of the i^{th} wave of **NorthInput**´s data supplies the i^{th} column of the input matrix A, while the left half of the $(N+i)^{th}$ wave supplies the i^{th} column of a $N \times N$ identity matrix. The right half of all 2N **NorthInput** data waves consists of zeroes, and the same is true of all of the **WestInput** data waves. These arrangements are made precise by the following definitions:

WestInput (i) (j) = 0.0, $0 \le i \le N-1,\ 1 \le j \le 2N$

NorthInput (i) (j) = {((A(i+1, j),,$0 \le i \le N-1, 1 \le j \le N$,,I(i+1, j-N),,$0 \le i \le N-1$, $N+1 \le j \le 2N$,,0.0,,$N \le i \le 2N-1$, $1 \le j \le 2N$))

where I[r,s] = 1 if $r = s$ and 0 otherwise.

The specification of the wavefront algorithm is given below:

<u>Instantiation</u>:

N = 100 ; North =1; South =1; East =2; West =2

```
USING   IN_PORT   (West, WestInput),
        IN_PORT   (North, NorthInput),
        OUT_PORT  (South, SouthOutput),
        OUT_PORT  (East, EastOutput)
```

EXECUTE NETWORK (0:N-1, 0:2*N-1) **OF** CELL **FOR NUMWAVES** = 2*N

<u>Declaration</u>:

```
(CELL() =

    CONSTANT  xself := self[1];
              yself := self[2];

    VARIABLE  r : REAL;

    {  r := 0.0
```

```
        ? [top, left]

        COND   (  (xself > yself) ∨
                  (xself < yself - N) ∨
                  (xself = yself  ∧  stage ≠ xself + 1) ∨
                  (xself = yself - N  ∧   stage < xself + 1) )

(1)               →  {right, bottom := top, left}

                 (xself = yself)                              -- stage = xself + 1

(2)               →  {    r := 1.0/(1.0 - top) ;
(3)                       right, bottom := r, left }

                 (xself = yself - N ∧  stage = xself + 1)

(4)               →  {    r := left ;
(5)                       right, bottom := top, left }

                 (xself = yself - N ∧  stage > xself + 1)

(6)               →  {    right, bottom := top, r ∗ left }

                 (stage ≤ xself)

(7)               →  {    right, bottom := left, top}

                 (stage = xself + 1)

(8)               →  {    r := top ∗ left ;
(9)                       right, bottom := left, r }

                 (stage > xself + 1)

(10)                →   {    right, bottom := left, top + r ∗ left }

        ! [bottom, right]
```

The arrangement of the computational grid is shown in Figure 1 below, and the output of the algorithm is described by the next result:

<u>Theorem</u>: The output ports of the APP wavefront algorithm have the following values:

EastOutput (i) (j) = 0.0, $0 \le i \le N\text{-}1,\ 1 \le j \le 2N$

SouthOutput (i) (j) = {((0.0,,$0 \le i \le N-1$, $1 \le j \le 2N$,,A(i-N+1, j, j),,$N \le i \le 2N-1$, $1 \le j \le N$,,A(i-N+1, j-N, N),,$N \le i \le 2N-1$, $N+1 \le j \le 2N$))

IV. Verification of the Algorithm

The theorem implies that the right half of waves N+1 through 2N of the **SouthOutput** port consists of the values {A(i,j,N)} for $1 \le i \le N$, $1 \le j \le N$. Since these values are the desired output of the APP algorithm, the correctness of the wavefront APP algorithm follows. In order to prove the theorem, it is necessary to examine the 2N wavefront computations performed by each element of the computational grid. Each such computation outputs two data items and possibly changes the value of the cell´s local variable. The correctness of the theorem follows easily from the following lemma, which describes the results of these computations.

Lemma:

$$
\text{(a) } [\text{bottom}'](i,j)(k) = \begin{cases}
\text{WestInput}(i-j)(k), & j \le i, \quad 1 \le k \le 2N,\\
\text{NorthInput}(j-i-1)(k), & j > N+i, \quad 1 \le k \le 2N,\\
A(j+1, k, k), & i < j < N, \quad 1 \le k \le i+1,\\
A(j+1, k, i+1), & i < j < N, \quad i+1 < k \le N,\\
A(j+1, k-N, i+1), & i < j < N, \quad N+1 \le k \le N+i+1,\\
I(j+1, k-N), & i < j < N, \quad N+i+1 < k \le 2N,\\
A(j+1-N, k, k), & N \le j \le N+i, \quad 1 \le k \le i+1,\\
A(j+1-N, k, i+1), & N \le j \le N+i, \quad i+1 < k \le N,\\
A(j+1-N, k-N, i+1), & N \le j \le N+i, \quad N+1 \le k \le N+i+1,\\
0.0, & N \le j \le N+i, \quad N+i+1 < k \le 2N
\end{cases}
$$

$$
\text{(b) } [\text{right}'](i,j)(k) = \begin{cases}
\text{WestInput}(i-j-1)(k), & j < i, \quad 1 \le k \le 2N,\\
\text{NorthInput}(j-i)(k), & j \ge N+i, \quad 1 \le k \le 2N,\\
A(i+1, k, k), & i \le j < N+i, \quad 1 \le k \le i+1,\\
A(i+1, k, i), & i \le j < N+i, \quad i+1 < k \le N,\\
A(i+1, k-N, i), & i \le j < N+i, \quad N+1 \le k \le N+i,\\
I(i+1, k-N), & i \le j < N+i, \quad N+i < k \le 2N
\end{cases}
$$

$$
\text{(c) } [r'](i,j)(k) = \begin{cases}
0.0, & 0 \le j < i, \quad 1 \le k \le 2N,\\
0.0, & N+i < j \le 2N, \quad 1 \le k \le 2N,\\
0.0, & i \le j \le N+i, \quad 1 \le k \le i,\\
A(j+1, i+1, i+1), & i \le j < N, \quad i < k \le 2N,\\
A(j-N+1, i+1, i+1), & N \le j \le N+i, \quad i < k \le 2N
\end{cases}
$$

Proof of Theorem:

By semantic equation{$S4_{ii}$},

SouthOutput (i) (j) = [bottom´] (N-1, i) (j) and
EastOutput (i) (j) = [right´] (i, 2N-1) (j)

The lemma implies that

SouthOutput (i) (j) = { ((**WestInput** (N-i-1) (j) = 0.0,0 ≤ i ≤ N-1, 1 ≤ j ≤ 2N,,A(i-N+1, j, j),,N ≤ i ≤ 2N-1, 1 ≤ j ≤ N,,A(i-N+1, j-N, N),,N ≤ i ≤ 2N-1, N+1 ≤ j ≤ 2N))

and

EastOutput (i) (j) = **NorthInput** (2N-i-1) (j) = 0.0

Proof of Lemma:

We will refer to the expressions to be verified as

$$\mathcal{E}(i, j, k) = (\mathcal{E}_{bottom}(i, j, k), \mathcal{E}_{right}(i, j, k), \mathcal{E}_{r}(i, j, k)).$$

The proof will use induction on each of these arguments. The organization of the proof is based on the geometry of the computational grid. In Figure 2, the cells are labeled with integers that correspond to the order in which they will be considered in the proof.

Figure 2

The proof is organized into cases that correspond to distinct regions of the computational grid. The grid is divided into three regions by two diagonals: the *lower diagonal* consists of the cells labeled 1 and 5 in Figure 2, while the *upper diagonal* consists of the cells that are labeled 3b and 8. The *lower triangle* consists of the cells to the left of the lower diagonal (labeled 2 and 4), while the *upper triangle* consists of the cells to the right of the upper diagonal (labeled 3c and 4). The remaining cells belong to the *left center* (labeled 3a, 6) and *right center* (labeled 7). The proof will begin by establishing the expressions for the cell in the northwest corner (labeled 1), and will then do so for the cells of the west and north edges (labeled 2, 3a, 3b, 3c). Once that has been accomplished, the expressions will be verified by induction (on the sum of the arguments) for the cells in the other regions, generally moving from the north and west toward the south and east.

The proof will be annotated by references to numbers of statements {Ax} in the algorithm specification, to recurrences {Ry} in the definition of the {A(i,j,k)}, or to numbers of the equations {Sw}.

<u>Case 1</u>: (northwest corner: i = 0, j = 0)

$\mathcal{E}_r$:	[r´] (0,0) (1)	=	1.0 / (1.0 - [top] (0,0) (1))	{A2}
		=	1.0 / (1.0 - **NorthInput** (0) (1))	{$S4_i$}
		=	1.0 / (1.0 - A(1, 1))	definition
		=	A(1,1,1)	{R3}

For $1 < k \leq 2N$,

	[r´] (0,0) (k)	=	[r´] (0,0) (k-1) = A(1,1,1)	{A1}, induction
$\mathcal{E}_{bottom}$:	[bottom´] (0,0) (k)	=	[left] (0,0) (k) = **WestInput** (0) (k)	{A1}, {A3}, {$S4_i$}
$\mathcal{E}_{right}$:	[right´] (0,0) (1)	=	[r´] (0,0) (1) = A(1,1,1) {A3}	

For $1 < k \leq 2N$,

[right´] (0,0) (k)	=	[top] (0,0) (k)	{A1}
	=	**NorthInput** (0) (k),	{$S4_i$}

which is equal to A(1, k) if $2 \le k \le N$ and to I(1, k-N if $N+1 \le k \le 2N$.

<u>Case 2</u> (west edge: j = 0, 0 < i < N)

$\mathcal{E}_r$:	[r´] (i,0) (k)	=	0.0	{A1}
$\mathcal{E}_{bottom}$:	[bottom´] (i,0) (k)	=	[left] (i,0) (k) = **WestInput** (i) (k)	{A1}, {$S4_i$}
$\mathcal{E}_{right}$:	[right´] (i,0) (k)	=	[top] (i,0) (k)	{A1}
		=	[bottom´] (i-1,0) (k)	{S3}
		=	**WestInput** (i-1) (k)	{$S4_i$}

<u>Case 3</u>: (north edge: i = 0, 0 < j < 2N)

(cells with label 3a: 0 < j < N)

$\mathcal{E}_r$:	[r´] (0,j) (1)	=	[top] (0,j) (1) * [left] (0,j) (1)	{A8}
		=	**NorthInput** (j) (1) * [left] (0,j) (1)	{$S4_i$}
		=	A(j+1,1,0) * [right´] (0,j-1) (1)	{S3}
		=	A(j+1,1,0) * A(1,1,1)	{2}
		=	A(j+1,1,1)	{R3}

For $1< k\le 2N$, [r´] (0,j) (k) = [r´] (0,j) (k-1) = A(j+1,1,1) {A10}, {3}

$\mathcal{E}_{bottom}$:	[bottom´] (0,j) (1)	=	[r´] (0,j) (1)	{A9}
		=	A(j+1,1,1)	preceding result

If $1 < k \le N$,

[bottom´] (0,j) (k)	=	[top] (0,j) (k) + ([r´] (0,j) (k)* [left] (0,j) (k))	{A10}
	=	**NorthInput** (j) (k) + (A(j+1,1,1) * [right´] (0, j-1) (k))	{$S4_i$}, {S3}, preceding result
	=	A(j+1,k,0) + (A(j+1,1,1) * A(1,k,0))	{S3}, induction
	=	A(j+1,k,1)	{R1b}

If $N+1 \le k \le 2N$,

$$\begin{aligned}
[\text{bottom}'](0,j)(k) &= [\text{top}](0,j)(k) + && \{A10\} \\
&\quad ([r'](0,j)(k) * [\text{left}](0,j)(k)) && \\
&= \textbf{NorthInput}(j)(k) + && \{S4_i\}\text{, preceding result} \\
&\quad (A(j+1,1,1) * [\text{left}](0,j)(k)) && \\
&= I(j+1,k-N) + && \{S3\}\text{, definition} \\
&\quad (A(j+1,1,1) * [\text{right}'](0,j-1)(k)) && \\
&= I(j+1,k-N) + A(j+1,1,1)*I(1,k-N) && \text{induction} \\
&= A(j+1,1,1) \text{ if } k = N+1 && 0 < j < N < k \\
&\quad \text{and } I(j+1,k-N) \text{ otherwise} &&
\end{aligned}$$

$\mathcal{E}_{\text{right}}$: For $1 \le k \le 2N$,

$$\begin{aligned}
[\text{right}'](0,j)(k) &= [\text{left}](0,j)(k) && \{A1\}, \{A8\}, \{A9\}, \{A10\} \\
&= [\text{right}'](0,j-1)(k), && \{S3\} \\
&= \begin{cases} A(1,1,1) & \text{if } k = 1, \\ A(1,k,0) & \text{if } 1 < k \le N, \\ \in I(1,k-N) & \text{if } k > N \end{cases} && \text{induction}
\end{aligned}$$

(cell with label 3b: $j = N$)

$$\begin{aligned}
\mathcal{E}_r: \quad [r'](0,N)(1) &= [\text{left}](0,N)(1) && \{A4\} \\
&= [\text{right}'](0,N-1)(1) && \{S3\} \\
&= A(1,1,1) && \text{induction}
\end{aligned}$$

For $1 < k \le 2N$,

$$\begin{aligned}
[r'](0,N)(k) &= [r'](0,N)(k-1) = A(j+1,1,1) && \{A1\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{E}_{\text{bottom}}: \quad [\text{bottom}'](0,N)(1) &= [\text{left}](0,N)(1) && \{A5\} \\
&= [\text{right}'](0,N-1)(1) && \{S3\} \\
&= A(1,1,1) && \text{induction}
\end{aligned}$$

For $1 < k \le N$,

$$\begin{aligned}
[\text{bottom}'](0,N)(k) &= [r'](0,N)(k) * [\text{left}](0,N)(k) && \{A6\} \\
&= A(1,1,1) * [\text{right}'](0,N-1)(k) && \{S3\}\text{, previous result} \\
&= A(1,1,1) * A(1,k,0) && \text{induction} \\
&= A(1,k,1) && \{R2a\}
\end{aligned}$$

For $N < k \le 2N$,

[bottom´] (0,N) (k)	=	[r´] (0, N) (k) * [left] (0,N) (k)	{A1}
	=	A(1,1,1) * [right´] (0,N-1) (k)	{S3}
	=	A(1,1,1) * I(1, k-N),	induction

which is equal to A(1,1,1) if k = N+1 and to 0.0 otherwise

$\mathcal{E}_{\text{right}}$: For $1 \le k \le 2N$,

[right´] (0,N) (k)	=	[top] (0,N) (k)	{A1},{A5},{A7}, {$S4_i$}
	=	**NorthInput** (N) (k)	

(cell with label 3c: j > N)

$\mathcal{E}_r$:	[r´] (0,j) (k)	=	0.0 for $1 \le k \le 2N$	{A1}
$\mathcal{E}_{\text{bottom}}$:	[bottom´] (0,j) (k)	=	[left] (0,j) (k)	{A1}
		=	[right´] (0,j-1) (k)	{S3}
		=	**NorthInput** (j-1) (k) for $1 \le k \le 2N$	{2}
$\mathcal{E}_{\text{right}}$:	[right´] (0,j) (k)	=	[top] (0,j) (k)	{A1}
		=	**NorthInput** (j) (k) for $1 \le k \le 2N$	{$S4_i$}

<u>Case 4</u> (lower and upper triangles: $i > j > 0$; $i+N < j < 2N$)

$\mathcal{E}_r$: If $i > j$ or $i+N < j$ and for $1 \le k \le 2N$, [r´] (i,j) (k) = 0.0 by {A1}.

$\mathcal{E}_{\text{bottom}}$: In the lower left and upper right corners, we have either $i > j > 0$ or $0 \le i < jN$, so that

[bottom´] (i , j) (k)	=	[left] (i, j) (k)	{A1}
	=	[right´] (i, j-1) (k)	{S3},

which is equal to **WestInput** (i-j) (k) if $i > j > 0$ and to **NorthInput** (j-i-1) (k) if $0 \le i < j-N$. Both conclusions follow by induction.

$\mathcal{E}_{\text{right}}$: If $i > j$ and $1 \le k \le 2N$, then

[right´] (i,j) (k)	=	[top] (i,j) (k)	{A1}
	=	[bottom´] (i-1,j) (k)	{S3}
	=	**WestInput** (i-j-1) (k)	induction

The same argument is used if $i < j-N$, except that in this case, we have [bottom´] (i-1, j) (k) = **NorthInput** (j-i) (k).

Case 5 (lower diagonal: $0 < i = j < N$):

$\mathcal{E}_r$: If $k \le i$, [r´] (i, i) (k) = 0.0 by {A1}.

[r´] (i, i) (i+1)	=	1.0 / (1.0 - [top] (i, i) (i+1))	{A2}
	=	1.0 / (1.0 - [bottom´] (i-1, i) (i+1)	{S3}
	=	1.0 / (1.0 - A(i+1, i+1, i))	{1}
	=	A(i+1, i+1, i+1)	{R3}

If $k > i+1$,

[r´] (i, i) (k)	=	[r´] (i, i) (k-1)	{A1}
	=	A(i+1, i+1, i+1)	{3}

$\mathcal{E}_{bottom}$: For $1 \le k \le 2N$,

[bottom´] (i, i) (k)	=	[left] (i, i) (k)	{A1}
	=	[right´] (i, i-1) (k)	{S3}
	=	**WestInput** (0) (k)	{2}

$\mathcal{E}_{right}$: As long as $k \neq i+1$,

[right´] (i, i) (k)	=	[top] (i, i) (k)	{A1}
	=	[bottom´] (i-1, i) (k)	{S3}

The value of [bottom] (i-1, i) (k) depends on the value of *k*, as follows:

[bottom´] (i-1, i) (k)	=	A(i+1, k, k)	if $1 \le k \le i$,
	=	A(i+1, k, i)	if $i+1 < k \le N$,
	=	A(i+1, k-N, i)	if $N+1 \le k \le N+i$,
	=	I(i+1, k-N)	if $N+i+1 \le k \le 2N$

All of these conclusions follow by induction and yield the desired results.

If $k = i+1$, we reason as follows:

[right´] (i,i) (i+1)	=	1.0 / (1.0 - [top] (i,i) (i+1))	{A2}, {A3}
	=	1.0 / (1.0 - [bottom´] (i-1, i) (i+1))	{S3}
	=	1.0 / (1.0 - A(i+1, i+1, i))	{1}
	=	A(i+1, i+1, i+1)	{R3}

Case 6 (left center: $0 < i < j < N$):

$\mathcal{E}_r$: If $k \leq i$, [r´] (i,j) (k) = 0.0 by {A7}.

[r´] (i,j) (i+1)	=	[top] (i,j) (i+1) * [left] (i,j) (i+1)	{A9}
	=	[bottom´] (i-1,j) (i+1) * [right´] (i, j-1) (i+1)	{S3}
	=	A(j+1, i+1, i) * A(i+1, i+1, i+1)	{1}, {2}
	=	A(j+1, i+1, i+1)	{R2b}

If k > i+1,

[r´] (i,j)	=	[r´] (i,j) (k-1)	{A10}
	=	A(j+1, i+1, i+1)	induction

$\mathcal{E}_{bottom}$: If $1 \leq k \leq i$,

[bottom´] (i,j) (k)	=	[top] (i, j) (k)	{A7}
	=	[bottom´] (i-1, j) (k)	{S3}
	=	A(j+1,k,k)	induction
[bottom´] (i, j) (i+1)	=	[top] (i, j) (i+1) * [left] (i, j) (i+1)	{A8, A9}
	=	[bottom´] (i-1, j) (i+1) * [right´] (i, j-1) (i+1)	{S3}
	=	A(j+1, i+1, i) * A(i+1, i+1, i+1)	induction
	=	A(j+1,i+1, i+1)	{R2b}

If k > i+1,

[bottom´] (i, j) (k)	=	[top] (i, j) (k) + ([r´] (i, j) (k) * [left] (i, j) (k))	{A10}
	=	[bottom´] (i-1, j) (k) + (A(j+1, i+1, i+1) * [right´] (i, j-1) (k))	{$\mathcal{E}_r$}, {S3}

The evaluation of this expression requires several subcases, depending on wavefront number:

First, if $i+1 < k \leq N$, it is equal to

		A(j+1, k, i) + (A(j+1, i+1, i+1) * A(i+1, k, i))	induction
	=	A(j+1, k, i+1)	{R1}

Next, if $N+1 \le k < N+i+1$, it is equal to

$$A(j+1, k-N, i) + (A(j+1, i+1, i+1) * A(i+1, k-N, i)) \qquad \text{induction}$$

$$= A(j+1, k-N, i+1) \qquad \{R1\}$$

Finally, if $N+i+1 \le k \le 2N$, it is equal to

$$= I(j+1, k-N) + (A(j+1, i+1, i+1) * I(i+1, k-N)) \qquad \text{induction}$$

which is equal to $A(j+1, i+1, i+1)$ if $k = N+i+1$ and to $I(j+1, k-N)$ if $k > N+i+1$.

$\mathcal{E}_{\text{right}}$: [right´] (i, j) (k) = [left] (i,j) (k) {A6, A7, A9, A10}
= [right´] (i, j-1) (k) {S3}

which in turn is equal to

$A(i+1, k, k)$	if $1 \le k \le i+1$,
$A(i+1, k, i)$	if $i+1 < k \le N$,
$A(i+1, k-N, i)$	if $N+1 \le k \le N+i$,
$I(i+1, k-N)$	if $N+i < k \le 2N$.

All of these conclusions follow from induction on the second argument.

<u>Case 7 (right center: $N \le j < N+i$)</u>:

$\mathcal{E}_{\text{r}}$: If $1 \le k \le i$, [r´] (i,j) (k) = 0.0 by {A7}.

[r´] (i,j) (i+1) = [bottom´] (i-1,j) (i+1) * [right´] (i, j-1) (i+1) {A8}, {S3}
= $A(j-N+1, i+1, i) * A(i+1, i+1, i+1)$ induction
= $A(j-N+1, i+1, i+1)$ {R2b}

If $k > i+1$, the argument is identical with that used in Case 6.

$\mathcal{E}_{\text{bottom}}$: As in Case 6, if $1 \le k \le i$,

[bottom´] (i,j) (k) = $A(j-N+1,k,k)$

[bottom´] (i, j) (i+1) = [bottom´] (i-1, j) (i+1) * [right´] (i, j-1) (i+1) {A9, A10}{S3}

= $A(j-N+1, i+1, i) * A(i+1, i+1, i+1)$ induction

$$= \quad A(j\text{-}N\text{+}1, i\text{+}1, i\text{+}1) \qquad \{R2b\}$$

If k > i+1,

$$[\text{bottom}'](i, j)(k) = [\text{bottom}'](i\text{-}1, j)(k) + (A(j\text{-}N\text{+}1, i\text{+}1, i\text{+}1) * [\text{right}'](i, j\text{-}1)(k)) \qquad \{A10\}, \{\mathcal{E}_3\}, \{S3\}$$

As above, the evaluation of this expression requires several subcases, depending on wavefront number:

First, if $i+1 < k \le N$, it is equal to

$$A(j\text{-}N\text{+}1, k, i) + (A(j\text{-}N\text{+}1, i\text{+}1, i\text{+}1) * A(i\text{+}1, k, i)) \qquad \text{induction}$$

$$= \quad A(j\text{-}N\text{+}1, k, i\text{+}1) \qquad \{R1\}$$

Next, if $N+1 \le k < N+i+1$, it is equal to

$$A(j\text{-}N\text{+}1, k\text{-}N, i) + (A(j\text{-}N\text{+}1, i\text{+}1, i\text{+}1) * A(i\text{+}1, k\text{-}N, i)) \qquad \text{induction}$$

$$= \quad A(j\text{-}N\text{+}1, k\text{-}N, i\text{+}1) \qquad \{R1\}$$

If $k = N+i+1$, it is equal to

$$I(j\text{+}1, k\text{-}N) + (A(j\text{-}N\text{+}1, i\text{+}1, i\text{+}1) * I(i\text{+}1, k\text{-}N)) \qquad \text{induction}$$

$$= \quad A(j\text{-}N\text{+}1, i\text{+}1, i\text{+}1) \text{ since } j\text{+}1 \neq i\text{+}1 = k\text{-}N.$$

Finally, if $N+i+1 < k \le 2N$, it is equal to

$$(A(j\text{-}N\text{+}1, i\text{+}1, i\text{+}1) * I(i\text{+}1, k\text{-}N)) \qquad \text{induction}$$

$$= \quad 0.0 \text{ since } i\text{+}1 < k\text{-}N.$$

$\mathcal{E}_{right}$: The argument is identical with that given for Case 6.

<u>Case 8 (upper diagonal: $0 < i < N$; $j = N+i$)</u>:

$\mathcal{E}_r$: If $k \neq i+1$, the argument is identical with that used in Case 7.

$$[r'](i, N\text{+}i)(i\text{+}1) = [\text{left}](i, N\text{+}i)(i\text{+}1) \qquad \{A4\}$$
$$= [\text{right}'](i, N\text{+}i\text{-}1)(i\text{+}1) \qquad \{S3\}$$

$$= A(i+1, i+1, i+1) \qquad \text{induction}$$

$\mathcal{E}_{bottom}$: Once again, the proof requires several subcases, depending on wavefront number:

First, if $k \le i+1$,

$$\begin{aligned}
[\text{bottom}'] (i, N+i) (k) &= [\text{left}] (i, N+i) (k) && \{A1\},\{A4\},\{A5\} \\
&= [\text{right}'] (i, N+i-1) (k) && \{S3\} \\
&= A(i+1, k, k) && \text{induction}
\end{aligned}$$

If $k > i+1$,

$$\begin{aligned}
[\text{bottom}'] (i, N+i) (k) &= [r'] (i, N+i) (k) \\
&\quad * [\text{left}] (i, N+i) (k) && \{A7\} \\
&= A(i+1, i+1, i+1) \\
&\quad * [\text{right}'] (i, N+i-1) (k) && \{\mathcal{E}_3\}, \{S3\}
\end{aligned}$$

The value of [right´] (i, N+i-1) (k) depends on the value of *k*, as follows:

$$[\text{right}'] (i, N+i-1) (k) = \begin{cases} A(i+1, k, i) & \text{if } i+1 < k \le N, \\ A(i+1, k-N, i) & \text{if } N+1 \le k \le N+i, \\ I(i+1, k-N) & \text{if } N+i < k \le 2N \end{cases}$$

It follows that

$$\begin{aligned}
[\text{bottom}'] (i, N+i) (k) &= A(i+1, i+1, i+1) * A(i+1, k, i) \\
&= A(i+1, k, i+1) \text{ if } i+1 < k \le N, \\
&= A(i+1, i+1, i+1) * A(i+1, k-N, i) \\
&= A(i+1, k-N, i+1) \text{ if } N+1 < k \le N+i,
\end{aligned}$$

$$\begin{aligned}
\text{and} \quad &= A(i+1, i+1, i+1) * I(i+1, k-N) \\
&= A(i+1, i+1, i+1) \text{ if } k = N+i+1 \text{ or} \\
&\quad 0.0 \text{ if } N+i+1 < k \le 2N.
\end{aligned}$$

$\mathcal{E}_{right}$: For all *k*,

$$\begin{aligned}
[\text{right}'] (i, N+i) (k) &= [\text{top}] (i, N+i) (k) && \{A1\} \\
&= [\text{bottom}'] (i-1, N+i) (k) && \{S3\} \\
&= \textbf{NorthInput} (N) (k) && \text{induction}
\end{aligned}$$

References

[1] Kung, S. Y, K. S. Arun, R. J. Gal-Ezer, and D. V. B. Rao, "Wavefront array processor:

language, architecture, and applications", *IEEE Transactions on Computers* **31** (1982), 1054-1065.

[2] Rice, M. D., "Covering Spaces, Boundary Sets, and Interconnection Networks", *Annals of the New York Academy of Sciences*, to appear.

[3] M. D. Rice and S. B. Seidman, "Wavefront computation: language and verification" *Proceedings of the Second International Workshop on Array Structures*, Université de Montréal, 1991.

[4] Y. Robert and D. Trystram, "An orthogonal systolic array for the algebraic path problem", *Computing* **39** (1987), 187-199.

[5] G. Rote, "A systolic array algorithm for the algebraic path problem", *Computing* **34** (1985), 191-219.

[6] U. Zimmermann, "Linear and combinatorial optimization in ordered algebraic structures", *Annals of Discrete Mathematics* **10** (1981), 1-380.