Converting Affine Recurrence Equations to Quasi-Uniform Recurrence Equations

Yoav Yaacoby
Rafael, Dept. 84
P.O. Box 2250
Haifa, 31021, Israel

Peter Cappello
Department of Computer Science
University of California
Santa Barbara, CA 93106

April 4, 1994

Abstract

Most work on the problem of synthesizing a systolic array from a system of recurrence equations is restricted to systems of uniform recurrence equations. Recently, researchers have begun to relax this restriction to include systems of affine recurrence equations. A system of uniform recurrence equations typically can be embedded in spacetime so that the distance between a variable and a dependent variable does not depend on the problem size. Systems of affine recurrence equations that are not uniform do not enjoy this property. A method is presented for converting a system of affine recurrence equations to an equivalent system of recurrence equations that is uniform, except for points near the boundaries of its index sets. Necessary and sufficient conditions are given for an affine system to be amenable to such a conversion, along with an algorithm that checks for these conditions, and a procedure that converts those affine systems which can be converted.

The characterization of convertible systems brings together classical ideas in algebraic geometry, number theory, and matrix representations of groups. While the proof of this characterization is complex, the characterization itself is simple, suggesting that the mathematical ideas are well chosen for this difficult problem in array design.

Keywords: affine recurrence equation, fundamental region, matrix representation of a group, parallel computation, processor array, systolic array, uniform recurrence equation.

1 Introduction

An important property of any VLSI system is physical regularity. Systolic arrays [19, 18] and wavefront arrays [20] have an iterative form of physical regularity. Array architecture is suited to VLSI technology; replicating a processing element reduces design cost, and neighbor communication between processing elements reduces operation cost. Leiserson, Rose, and Saxe [23, 22] show how to convert a finite network without zero-delay cycles to an equivalent network that functions systolically (see also [34]). Melhem and Rheinboldt [24] give a mathematical model for the verification of systolic networks.

A system of uniform recurrence equations, as defined by Karp, Miller, and Winograd [17, 16], maps especially well onto a systolic/wavefront array. This is noted explicitly by Chen and Mead [5], and Quinton [30], for example. Linearly mapping a system of recurrence equation's index sets into spacetime, has been pursued by Cappello and Steiglitz [3, 4]. Fortes et al. [14] survey seventeen methodologies for the design of systolic arrays. Systematic translation from either a program fragment or a system of uniform recurrence equations to systolic/wavefront arrays has been studied by Moldovan [27] [26], Quinton [30], Winkler and Miranker [25], Delosme and Ipsen [11], and Moldovan and Fortes [28]. Rao [32] introduces and analyzes a class of algorithms, called regular iterative algorithms, which contains systolic algorithms.

Miranker and Winkler [25], among others, describe how to convert a system of recurrence equations that overwrites array variables to one that does not. Since this can always be done, one's attention typically is restricted to systems of recurrence equations where variable overwriting does not occur. This restriction is made here.

A system of uniform recurrence equations typically can be mapped linearly into spacetime so that interprocessor communication requires only a fixed amount of memory, and fixed-length interconnections. There is no such linear mapping into spacetime for systems of *affine* recurrence equations that are not uniform. Delosme and Ipsen [12] present the basic elements of a methodology for determining systolic array schedules for 2-dimensional systems of affine recurrence equations. Choffrut and Culik [6] treat a related problem. They apply a geometric transformation to a systolic array, such that an output can be fed back as an input via physically neighboring connections. They fold the array, eliminating long wires for connections between elements (in a 2-dimensional array) that are related by reflections and/or rotations. Reflections also have been used by Rajopadhye, Mui, and Kiaei [31] to find piecewise linear schedules for systems of affine recurrence equations. Indeed, it has been shown, for systems of uniform recurrence equations, that piecewise linear schedules are optimal [9, 39]. These results build on the linear scheduling work of Shang and Fortes [33].

Our paper treats systems of affine recurrence equations of any finite dimension. We formulate a 'generalized fold', and consider the following question:

Which systems of affine recurrence equations can be converted, by a generalized fold, to an equivalent system that is uniform, except for points near the 'folds'?

These latter systems are called systems of *quasi-uniform* recurrence equations. Where linear embeddings fail, a generalized fold may succeed in enabling a VLSI array implementation.

By making use of algebraic geometry, number theory, and matrix representations of groups, we provide:

- a characterization of those systems of affine recurrence equations that can be converted, by a generalized fold, to an equivalent system of quasi-uniform recurrence equations (Theorem. 3.12 on page 25).
- an algorithm for deciding if a system of affine recurrence equations can be so converted (Algorithm 3.1 on page 22);
- a procedure for converting a system of affine recurrence equations to an equivalent system of quasi-uniform recurrence equations (Procedure 3.2 on page 28).

The balance of this paper is organized as follows. Definitions and examples are contained in § 2. A sequence of theorems, presented in § 3, includes the bullet items mentioned above. To save space, we have omitted some proofs, which appear in [35]. Conclusions are given in § 4.

2 Definitions

We present some basic definitions which are necessary for the paper to be self-contained. Some of our definitions are technically different from those appearing in the literature. New definitions also are presented.

2.0.1 Notation

In this paper a column vector sometines is written as $(r_1, r_2, \ldots, r_n)$; a row vector is written as $[c_1, c_2, \ldots, c_n]$.
$\mathbf{R}^n$ refers to this vector space with the Euclidean metric (i.e., the Euclidean n-space is denoted by $\mathbf{R}^n$). The set of integer lattice points in $\mathbf{R}^n$ are denoted by $\mathbf{Z}^n$. A lattice in $\mathbf{Z}^n$ is denoted by $\mathbf{L}^n$ (e.g., the n-vectors with even entries).

Definition 2.1 *Index set*: A finite set $S \subseteq \mathbf{L}^n$; $p \in S$ is referred to as an [*index*] *point*.

Definition 2.2 *System of recurrence equations (SRE)*: A set of equations of the form

$$a_i(p) = \alpha_i(a_{k_1}(\delta^1_{k_1 i}(p)), a_{k_2}(\delta^2_{k_2 i}(p)), \ldots, a_{k_{r(i)}}(\delta^{r(i)}_{k_{r(i)} i}(p)))$$

$$\forall p \in C_i,\ 1 \le i \le m,\ 1 \le k_j \le m$$

where

1. a_i is an array whose index set is $S_i \subset \mathbf{L}_i^n$, where $\mathbf{L}_i^n$ is a lattice.
2. $C_i \subset S_i$, the *domain of computation of* a_i, is the set of points for which a_i is computed[1].
3. α_i is a function with an $r(i)$-ary domain.
4. $\delta^l_{k_l i}$ is a function called a *dependence map*, which has domain C_i and range $\hat{C}^l_{k_l i}$. (The superscript l is introduced because when an array is used as an argument more than once in the same function, then its uses may have different dependences. For example, even if $k_1 = k_2$, we can have $\delta^1_{k_1 i} \neq \delta^2_{k_2 i}$. The superscript is omitted when the context is clear.)
5. $S_i = C_i \bigcup_{l,j} \hat{C}^l_{ij}$ (i.e., S_i is the set of all points where a_i is either computed or used).

Array a_i may be computed differently in different subsets of C_i (i.e., by more than one equation). In this case, the number of such subsets, and associated equations, must be fixed. (In order to denote such a subset, we refer to the recurrence equation by number.)

2.1 Example 1

The SRE[2] below, due to Delosme and Ipsen [12], factors a symmetric Toeplitz matrix and its inverse into LDL^T:

$$\begin{aligned}
&1 \le i \le n, && a_1(i,0) \equiv t_{i-1} && (1)\\
&2 \le i \le n, && a_2(i,0) \equiv t_{i-1} && (2)\\
&1 \le j \le n-1, && a_3(j) = a_2(j+1,j-1)/a_1(j,j-1) && (3)\\
&\quad j+1 \le i \le n, && a_1(i,j) = a_1(i-1,j-1) - a_3(j)a_2(i,j-1) && (4)\\
&\quad j+2 \le i \le n, && a_2(i,j) = -a_3(j)a_1(i-1,j-1) + a_2(i,j-1) && (5)\\
& && a_2(1,0) = 1 && (6)\\
&1 \le j \le n-1, && && \\
&\quad 1 \le i \le j+1, && a_2(i,j) = -a_3(j)a_2(j+2-i,j-1) + a_2(i,j-1) && (7)
\end{aligned}$$

The arrays in this example are a_1, a_2, a_3. For this SRE, a_1 has the following domain of computation: $C_1 = \{(i,j) | 1 \le j \le n-1, j+1 \le i \le n\}$.

[1] For $p \in S_i - C_i$, $a_i(p)$ is an input.

[2] Systolic arrays have been investigated for factoring a symmetric Toeplitz matrix into LDL^T, and solving the Toeplitz system (see, e.g., [21, 10, 12]).

Some of the dependence maps in this SRE are:

$$\delta_{23}(j) = (j+1, j-1) \quad \delta_{13}(j) = (j, j-1) \qquad \delta_{11}(i,j) = (i-1, j-1)$$
$$\delta_{31}(i,j) = (j) \qquad\qquad \delta_{21}(i,j) = (i, j-1)$$

The range of δ_{21} is: $\hat{C}_{21} = \{(i,j) | j+1 \leq i \leq n, 0 \leq j \leq n-2\}$.

The index set of the array a_1 is: $S_1 = \{(i,0) | 1 \leq i \leq n-1\} \bigcup C_1$, where the first part includes the index points in which a_1 is used and not computed (i.e., input), and the second part includes the index points in which a_1 is computed.
In Eq. (7) of the SRE,

$$\delta^1_{22}(i,j) = (j+2-i, j-1), \quad \delta^2_{22}(i,j) = (i, j-1) \ .$$

These are distinct dependence maps of array a_2 on itself. Superscripts hence are used to distinguish them. Array a_2 also is computed in another equation.

Definition 2.3 *Dependence graph*: A finite directed graph G, related to an SRE, whose set of nodes $N = \{a_i(p) | a_i$ is an array in the SRE, and $p \in S_i\}$, and whose set of arcs $A = \{(a_i(p_1), a_j(p_2)) | p_2 = \delta^l_{ji}(p_1)$ for some l in the SRE $\}$.

The dependence graph of the SRE in Ex. 1 is given for $n = 3$ in Fig. 1.

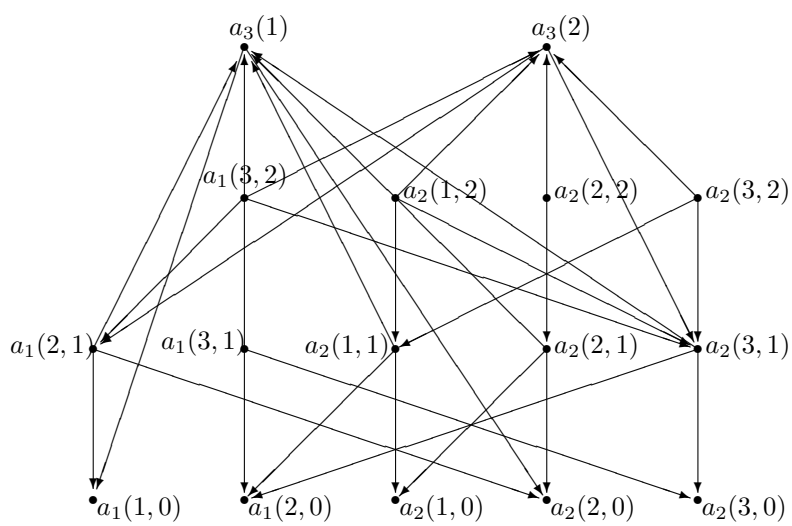


Figure 1: Dependence graph for the system of recurrence equations in Ex. 1.

Definition 2.4 *Reduced dependence graph (RDG)*: A directed multigraph related to a system of recurrence equations, with a node for each array a_i which is computed in the SRE, and an arc from a_i to a_j for each dependence map δ^l_{ji} in the SRE. Arcs may be labeled by the corresponding dependence maps.

An RDG for the system of recurrence equations of Ex. 1 is depicted in Fig. 2.

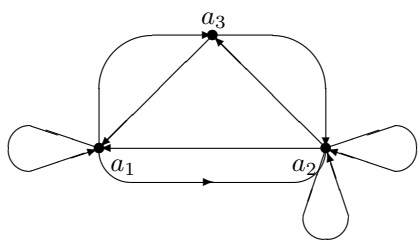


Figure 2: An RDG for the system of recurrence equations of Ex. 1.

Definition 2.5 *Strongly connected system of recurrence equations*: A system of recurrence equations that has a strongly connected RDG[3].

The RDG shown in Fig. 2 is strongly connected, and thus the SRE in Ex. 1 is strongly connected. Any SRE can be decomposed into strongly connected SREs, where each strongly connected SRE depends only on the output of previous strongly connected SREs. This is described in [12] and [1]. In this paper, it is assumed that such a decomposition has already been done: SREs are assumed to be strongly connected. This assumption is removed at the end of this paper.

Definition 2.6 *Direct dependence*: $a_j(p_1)$ is said to depend directly on $a_i(p_2)$ if and only if $p_2 = \delta^l_{ij}(p_1)$ for some l (i.e., $a_i(p_2)$ is an argument of α_j in the computation of $a_j(p_1)$).

Definition 2.7 *Dependence*: An element of array $a_j(p_1)$ depends on an element of array $a_i(p_2)$ if and only if there exists a path in the dependence graph from the node corresponding to $a_j(p_1)$ to the node corresponding to $a_i(p_2)$.

In Ex. 1, $a_1(3,2)$ depends on $a_1(2,0)$, but this is not a direct dependence. Array element $a_1(3,2)$, for example, depends directly on $a_2(3,1)$.
In this paper, we concentrate on the two special kinds of dependence maps defined below.

[3]Several similar terms appear in the literature. A strongly connected component is defined in [17]. A π block, is defined in [1] for a Fortran-like Do loop (where overwriting is allowed), where the nodes in the graph correspond to statements. A 'step' (although more restrictive) is defined in [12]. A tightly connected component is defined in [32].

Definition 2.8 *Affine dependence*: A dependence map of the form:

$$\forall p \in C_j,\ \delta_{ij}(p) = D_{ij}p + d_{ij}$$

where $D_{ij} \in \mathbf{Z}^{n\times n}$, and $d_{ij} \in \mathbf{Z}^n$.

In the remainder of this paper, we assume that D_{ij} is nonsingular and integer, unless specified otherwise.

Definition 2.9 *Uniform dependence*: An affine dependence of the form: $\delta_{ij}(p) = p + d_{ij}$ (i.e., $D_{ij} = I$).

In Eq. (7) of Ex. 1, the dependence map δ^1_{22} is affine, but not uniform, because it has the form:

$$\delta^1_{22}(i,j) = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

In Eq. (4) of Ex. 1, the dependence map δ_{11} is uniform. The dependence map δ_{23} is not affine, since the domain and range have different dimensions. In such cases, one can always add dimensions to the array of lower dimension and insert a 0 in all missing dimensions. This procedure leads to a dependence map with a singular linear part. However, there exists a procedure which 'augments' the dimension of the lower dimension arrays, such that all arrays have the same dimension, and the linear parts are nonsingular. In this procedure, the lower dimension arrays are computed (or input) on the boundary only, and then 'propagated' to the other dimensions. This procedure is presented in [25], among others. After 'augmenting' the SRE in Ex. 1, we get the following SRE:

2.2 Example 1A

$$1 \le i \le n, \qquad a_1(i,0) \equiv t_{i-1} \tag{1}$$
$$2 \le i \le n, \qquad a_2(i,0) \equiv t_{i-1} \tag{2}$$
$$1 \le j \le n-1, \qquad a_3(j,j) = a_2(j+1,j-1)/a_1(j,j-1) \tag{3}$$
$$j+1 \le i \le n, \qquad a_3(i,j) = a_3(i-1,j) \tag{4}$$
$$j+1 \le i \le n, \qquad a_1(i,j) = a_1(i-1,j-1) - a_3(i-1,j)a_2(i,j-1) \tag{5}$$
$$j+2 \le i \le n, \qquad a_2(i,j) = -a_3(i-1,j)a_1(i-1,j-1) + a_2(i,j-1) \tag{6}$$
$$a_2(1,0) = 1 \tag{7}$$
$$1 \le j \le n-1,$$
$$1 \le i \le j-1, \qquad a_3(i,j) = a_3(i+1,j) \tag{8}$$
$$1 \le i \le j+1, \qquad a_2(i,j) = -a_3(i+1,j)a_2(j+2-i,j-1) + a_2(i,j-1) \tag{9}$$

Definition 2.10 *A system of affine [uniform] recurrence equations (SARE [SURE])*: A system of recurrence equations, where the dependence maps are affine [uniform], and every array is computed in one recurrence equation for its entire domain of computation.

In Ex. 1 and 1A, the SRE is not affine because the array a_2 is computed in two different equations for two subsets of its domain of computation. We may split the SRE in Ex. 1A into three parts: 1) Eqs. (1-6); 2) Eq. (8); and 3) Eq. (9). The result is three SREs. The first one is not affine (because the array a_3 has two equations); the second system of recurrence equations is uniform; the third is affine.

Definition 2.11 *Domain size parameters*: Those parameters in the SRE which instantiate the domains of computation. By changing these parameters, we change the size of domains of computation; the recurrence equations change in no other way.

Domain size parameters can be the input's size for example, as in Ex. 1 where the only parameter is the size, n, of the matrix to be factored.

Definition 2.12 *Cycle dependence map*: A composition of dependence maps associated with the arcs of a directed cycle in the RDG (not necessarily simple).

2.2.1 Notation

We denote by δ_i a cycle dependence map which starts at a_i, and by δ_{ij} a direct dependence map of a_j on a_i. This notation is changed only after a conversion is done, when all the subscripts are doubled (e.g., in Thm. 3.8).

In Ex. 1, the following dependence maps constitute a cycle dependence map: δ_{22}, δ_{32}, δ_{13}, δ_{21}. This cycle dependence map might be denoted by δ_2. Since there is more than one cycle which starts at a_2, we denote it by δ_2^1, preventing confusion. (When it does not matter which cycle we refer to, we omit the superscript).

Definition 2.13 *n-dimensional system of recurrence equations*: A system of affine recurrence equations, which satisfies the following properties:

1. $C_i \subset \mathbf{L}_i^n$, where C_i is the domain of computation of the array a_i and $\mathbf{L}_i^n$ is a lattice.
2. All its dependence maps have linear parts $D_{ij} \in \mathbf{Z}^{n \times n}$.
3. For every cycle dependence map δ_i, $\forall k \in \mathbf{N}$, there exist domain size parameters such that the domain of computation C_i contains H, a k^n hypercube, and $\gamma_i(H) > 1$.

In Ex. 1A, the SRE defined in Eq. (8) and the SRE defined in Eq. (9) are 2-dimensional. This is because the SRE is affine, the domains of computation are in $\mathbf{L}_2^2 = \mathbf{L}_3^2 = \mathbf{Z}^2$, and the linear parts of the dependence maps are 2×2 integer matrices. Finally, the domain size parameter n allows the third requirement of 2-dimensionality to be satisfied.

Definition 2.14 *Computable system of recurrence equations*: An SRE is computable if and only if there are no cycles in its dependence graph (i.e., no variable depends on itself).

Definition 2.15 *Partitioned Linear Transformation (PLT)*: Let V be an n-dimensional vector space. Let $P_0, P_1, \ldots, P_{m-1}$ be m convex polytopes[4], which partition V. Let $T = \{T_0, T_1, \ldots, T_{m-1}\}$ be a set of nonsingular $n \times n$ matrices. A *Partitioned Linear Transformation* $\mathbf{T} : V \to V$ with respect to $\{P_i\}$ and T is such that $\mathbf{T}(x) = T_i(x)$ if and only if $x \in P_i$. The PLT may be referred to as an m-fold PLT.

A 1-fold PLT is a linear transformation when $P_0 = V$. A PLT can be restricted to any $Q \subset V$. In what follows when no confusion can arise, we sometimes describe how Q is partitioned (by its intersection with the P_i's), and do not define the polytopes in V.

In Fig. 3 $Q \subset \mathbf{Z}^2 \subset \mathbf{R}^2$. Q is partitioned into Q_0, Q_1, Q_2, Q_3 by its intersection with four polytopes. A PLT is applied to Q such that

$$T_0 = I, \;\; T_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \;\; T_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \;\; T_3 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The resulting set is shown in Fig. 4. Since there are four subsets, this is a 4-fold PLT.

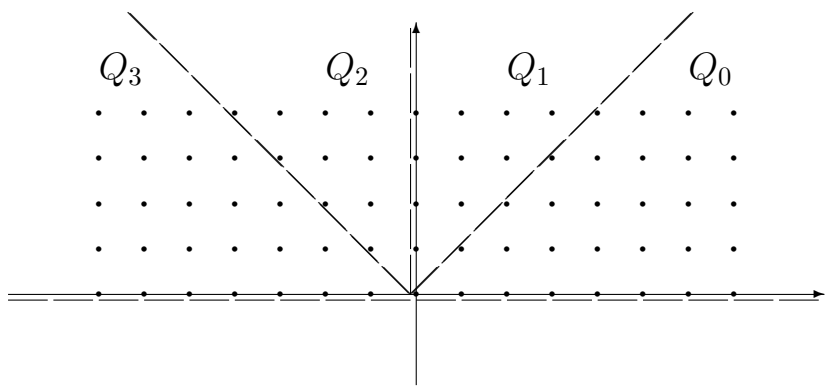


Figure 3: The set Q and its partitioning before the PLT.

Definition 2.16 The union of the domains of computation is composed of two subsets, the *uniform subdomain*, and the *nonuniform subdomain*. The uniform subdomain is the largest subset $U \subseteq \bigcup_i C_i$ (where C_i is the domain of computation of the array a_i), such that the SRE defined on this subset is uniform. The nonuniform subdomain $NU = \bigcup_i C_i - U$.

[4]By *convex polytope* we mean the intersection of a finite number of open or closed half spaces.

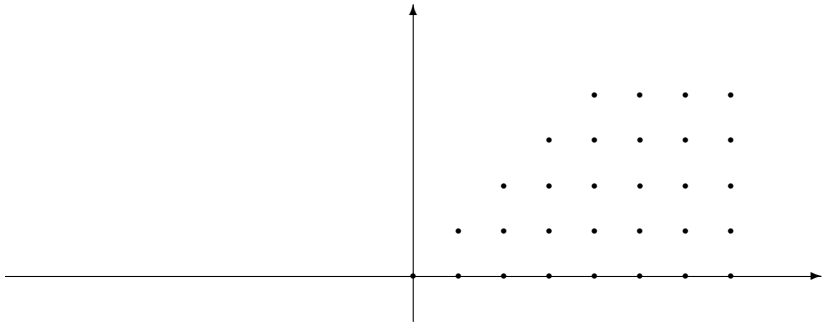


Figure 4: The set that results from applying the PLT to Q.

We define the set of points which are 'closest' to a set $Q \subset \mathbf{L}^n$. Since this set technically is not Q's boundary, it is referred to as a *border* of Q.

Definition 2.17 Given a set $Q \subset \mathbf{L}^n$, denote $d(x, Q) = \min_{x' \in \mathbf{L}^n - Q} \|x' - x\|$.
The set $\{x' \in \mathbf{L}^n - Q \mid d(x, Q) = \|x - x'\|,\ x \in Q\}$ is called the *border* of Q.
$d(x, Q)$ is called the *distance* from a point $x \in Q$ to the border of Q.

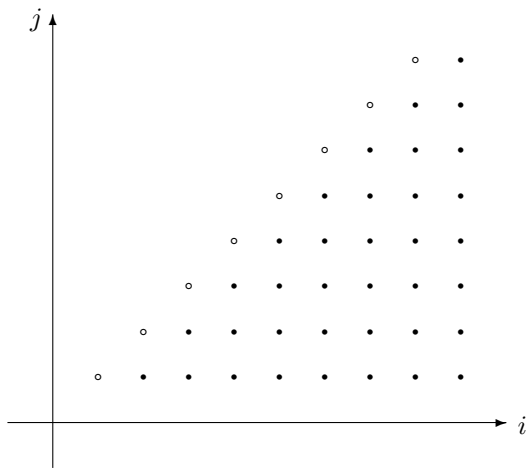
Definition 2.18 A *system of quasi-uniform recurrence equations* (SQURE): An SRE that satisfies

1. The SRE defined on the uniform subdomain is n-dimensional, for some n;
2. $\exists b \in \mathbf{R}$ so that for every value of the domain size parameters, if x is in the nonuniform subdomain, then $\max_i d(x, C_i) < b$.

An SQURE is said to have *quasi-uniform* dependences.

In Ex. 1A, Eqs. (1-6) are not an SARE, as has been noted before. These equations however are quasi-uniform because the nonuniform subdomain is the set $\{(j, j) | 1 \leq j \leq n-1\}$, and for every point (j, j) in this set, there is a point outside the whole set which has a distance of 1 from it (e.g., the point $(j, j+1)$). The domains of computation of this part of Ex. 1A are shown in Fig. 5.

Definition 2.19 A system of affine recurrence equations is *PLT convertible* when it can be converted to an equivalent SQURE (by equivalent we mean that the I/O relationship is the same) by 1) applying an m-fold PLT to the arrays' index sets, and 2) renaming each array such that a_i defined on the index points which are in the polytope P_j of the PLT, is called $a_{ij}, 0 \leq j < m$. Moreover, m does not depend on the domain size parameters.



• : index point in the uniform subdomain
∘ : index point in the nonuniform subdomain

Figure 5: Uniform and nonuniform subdomains of the SRE in Ex. 1A Eqs. (1-6).

In Ex. 1A, the system of affine recurrence equations in Eq. (9) is PLT convertible as is shown below. The index set of the array a_2 is:

$$S_2 = \{(i,j)|0 \leq j \leq n-1, 1 \leq i \leq j+2\} - \{(n-1, n+1)\}$$

The index set of the array a_3 is:

$$S_3 = \{(i,j)|1 \leq j \leq n-1, 2 \leq i \leq j+2\}$$

These index sets are shown in Fig. 6. The PLT is as follows. The polytopes are:

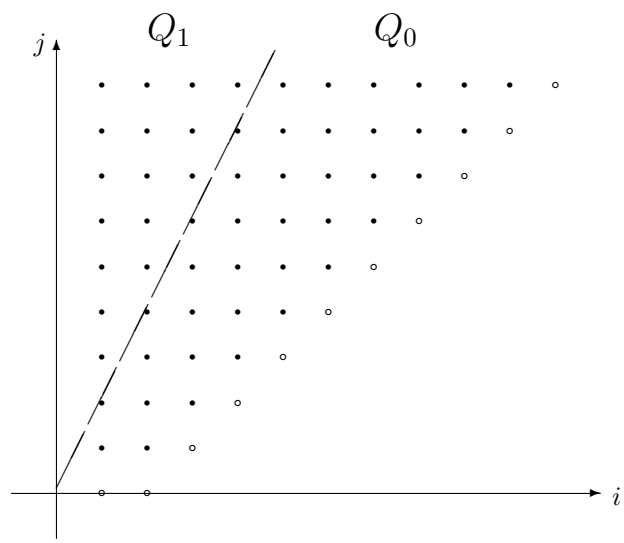
$$P_0 = \{(i,j)|2i \geq j\} \quad P_1 = \{(i,j)|2i < j\}$$

The index sets of the arrays are partitioned such that:

$$Q_0 = P_0 \bigcap (S_3 \bigcup S_2) \quad Q_1 = P_1 \bigcap (S_3 \bigcup S_2)$$

These subsets are shown in Fig. 6. The subset Q_0 remains in place (i.e., $T_0 = I$), while the subset Q_1 is multiplied by

$$T_1 = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}.$$



• : the domains of computation
∘ : input point in the index set

Figure 6: The index sets of a_2 and a_3 in Ex. 1A Eq. (9).

Equation (8) also is affected, but since array a_3 is propagated only (i.e., it does not change its value while propagating), we rewrite it such that it is not propagated back and forth after the conversion. This rewriting is done by repeating the augmentation process after the conversion. In Eqs. (1-7) array a_2 is renamed a_{20}. The system of recurrence equations after the PLT is as follows (details of this conversion are given in §§ 3.8):

2.3 Example 1B

$$
\begin{array}{lll}
1 \le i \le n, & a_1(i,0) \equiv t_{i-1} & (1) \\
2 \le i \le n, & a_2(i,0) \equiv t_{i-1} & (2) \\
1 \le j \le n-1, & a_3(j,j) = a_{20}(j+1,j-1)/a_1(j,j-1) & (3) \\
\quad j+1 \le i \le n, & a_3(i,j) = a_3(i-1,j) & (4) \\
\quad j+1 \le i \le n, & a_1(i,j) = a_1(i-1,j-1) - a_3(i-1,j)a_{20}(i,j-1) & (5) \\
\quad j+2 \le i \le n, & a_{20}(i,j) = -a_3(i-1,j)a_1(i-1,j-1) + a_{20}(i,j-1) & (6) \\
 & a_{20}(1,0) = 1 & (7) \\
1 \le j \le n-1, & & \\
\quad \lceil j/2 \rceil \le i \le j-1, & a_3(i,j) = a_3(i+1,j) & (8) \\
\quad \lceil j/2 \rceil \le i \le \min(\lceil j/2 \rceil + 2, j+1), & a_{20}(i,j) = -a_3(i+1,j)a_{20}(j+2-i,j-1) + a_{20}(i,j-1) & (9a) \\
\quad \lceil j/2 \rceil + 3 \le i \le j+1, & a_{20}(i,j) = -a_3(i+1,j)a_{21}(i-3,j-1) + a_{20}(i,j-1) & (9b) \\
\quad \lceil j/2 \rceil + 1 \le i \le j-1, & a_{21}(i,j) = -a_3(i+1,j)a_{20}(i+2,j-1) + a_{21}(i-1,j-1) & (9c) \\
3 \le j \text{ odd} \le n-1, & & \\
\quad i = (j+1)/2, & a_{21}(i,j) = -a_3(i+1,j)a_{20}(i+2,j-1) + a_{20}(i-1,j-1) & (9d)
\end{array}
$$

The SRE 9a - 9d presented in Ex. 1B is quasi-uniform, because Eqs. (9a),(9d) compute values for the arrays a_{20} and a_{21} in points which have a distance less than 3 from a point outside the domains of computation. We thus get that the three SREs in Ex. 1B: Eqs. (1-6), (7-8), and (9a-9d) are SQUREs[5]. The uniform and nonuniform subdomains of the SRE in Eqs. (9a-9d) are shown in Fig. 7, for $n = 12$.

Definition 2.20 A system of affine recurrence equations is *convertible* if it can be transformed, by affine transformations on the index sets of the arrays, into an SARE that is PLT convertible (different transformations may be applied to different index sets).

An example of a convertible SRE is given below.

[5]After the PLT, the index sets of each array can be translated. This changes only the translation part of the dependence maps. This translation can be made to minimize the distances between points that depend directly on one another (e.g., in Ex. 1B we can translate the index space of the array a_{21} by $(2,0)$, getting shorter distances).

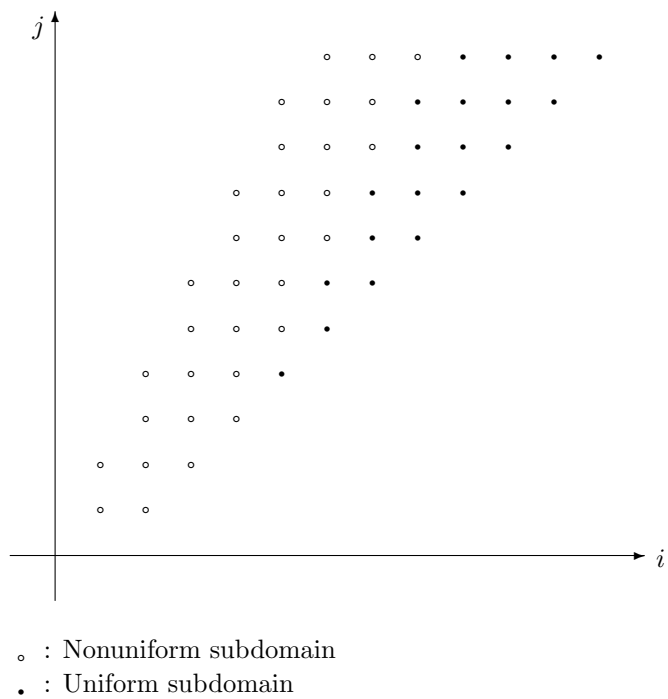


Figure 7: The uniform and nonuniform subdomains for Ex. 1B Eqs. 9a - 9d.

2.4 Example 2

$1 \le i \le m, \;\; -n \le j \le n, \;\; -n \le k \le n$

$$a_1(i,j,k) = a_1(i-1,j-1,k-1) + a_2(i-1,j+k-2,k-1) + a_2(i-1,j-k,j) \quad (1)$$

$1 \le i \le m, \;\; -n \le k \le n, \;\; k-n \le j \le k+n$

$$a_2(i,j,k) = a_1(i-1,j-k,-k) + a_2(i-1,j,k) \quad (2)$$

The system of affine recurrence equations in Ex. 2 is not PLT convertible[6], but if we apply a linear transformation

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

on the index set of array a_2, the following changes take place:

1. The domain of computation C_2 is multiplied by M and becomes identical to C_1.
2. $\delta_{21}(p) \leftarrow M\delta_{21}(p)$ (There are two of these).
3. $\delta_{22}(p) \leftarrow M\delta_{22}(M^{-1}p)$
4. $\delta_{12}(p) \leftarrow \delta_{12}(M^{-1}p)$

and we get the following SARE:

2.5 Example 2A

$1 \le i \le m, \;\; -n \le j \le n, \;\; -n \le k \le n$

$$a_1(i,j,k) = a_1(i-1,j-1,k-1) + a_2(i-1,j-1,k-1) + a_2(i-1,-k,j) \quad (1)$$

$$a_2(i,j,k) = a_1(i-1,j,-k) + a_2(i-1,j,k) \quad (2)$$

This system of affine recurrence equations is PLT convertible. A PLT can be constructed by splitting the index sets into the subsets shown in Fig. 8. The first axis is deleted for simplicity. The linear transformations for these subsets are:

$$T_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad T_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad T_6 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad T_7 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

[6]For a sketch of a proof, see [35, App. B].

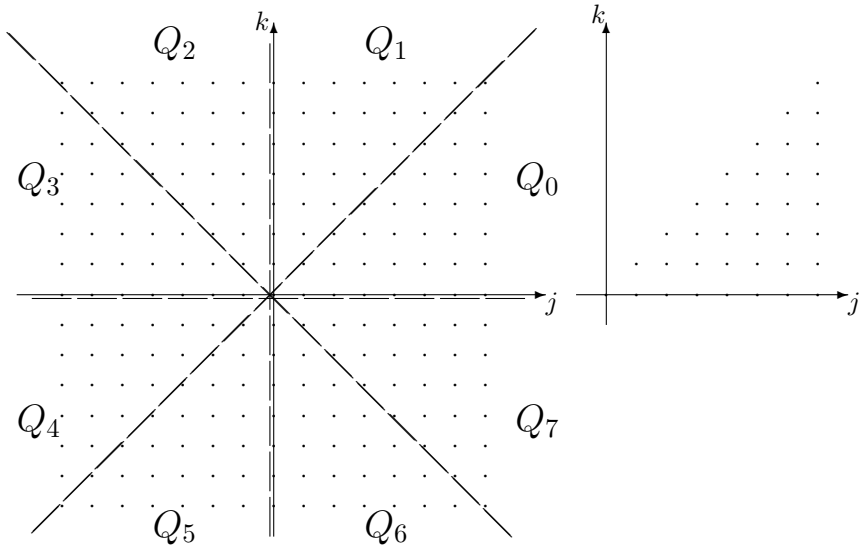


Figure 8: A PLT for Ex. 2.

The set resulting from the PLT is shown on the right in Fig. 8. The system of recurrence equations, after the PLT, has sixteen arrays a_{1i}, a_{2i} for, $0 \leq i \leq 7$, and has quasi-uniform dependences. The SRE in Ex. 2 thus is convertible.

3 Properties of Systems of Affine Recurrence Equations

Properties of systems of affine recurrence equations are investigated in this section. Again, all SREs are assumed to be strongly connected. An SARE can be converted to an equivalent SQURE, by 1) linearly transforming the domains of computation of the arrays, and 2) applying an m-fold PLT. Necessary and sufficient conditions are proven for m to be fixed with respect to the domain size parameters. Such convertibility is important because an SQURE typically can be realized with a VLSI systolic array, that has fixed wire lengths and memory sizes for each processing element (with the possible exception of some boundary elements). If a system of affine recurrence equations is not convertible, then any PLT embedding will require communication in spacetime over a distance that depends on the problem size: as the problem size grows, either processing element memory size and/or interconnection length also must grow. In this section, we concentrate on n-dimensional SAREs, as defined in the previous section. Our goals are: 1) to find necessary and sufficient conditions for an SARE to be convertible, 2) to find an algorithm which checks for these conditions, 3) to find a procedure for the construction of the conversion whenever it is possible.

3.1 Necessary conditions for convertibility

We first prove a necessary condition for an SARE to be PLT convertible. As mentioned earlier, all matrices are assumed to have integer entries, unless specified otherwise.

If K and k are to be integers, then the Lemma is still true, but $k = \lfloor c \cdot K \rfloor$.

Theorem 3.1 *Let SARE A be n-dimensional. If A is m-fold PLT convertible, then for every cycle dependence map δ_i (not necessarily simple) the linear part D_i satisfies $D_i^{L_i} = I$, for some $L_i \leq m$.*

As will be seen from Thm. 3.4, since D is integer, there is another upper bound on L, if $D^L = I$. L therefore is bounded by the minimum of these two upper bounds.

Also, if $L = \text{lcm}(L_i)$, then $\forall i, D_i^L = I$. As an example, consider Ex. 2A in § 2. The SARE in that example is PLT convertible. For every one of its cycle dependence maps, the linear part satisfies $D^4 = I$.

The property of n-dimensionality in the above theorem is crucial. In case this property is missing, the above theorem is not true (see an example in [35]).

Definition 3.1 *Semicycle dependence map*: A composition of dependence maps that corresponds to a semicycle[7] in the RDG, such that if an arc is traversed opposite to its sense in the semicycle, then the inverse of the corresponding dependence map is used.

The following theorem specifies a necessary condition for an n-dimensional SARE to be convertible. By 'cycle' we mean any cycle in the graph, not necessarily a simple cycle. An RDG has an infinite number of such cycles (again, we assume that it is strongly connected).

Theorem 3.2 *If an n-dimensional SARE is convertible, then the linear parts of all the cycle dependence maps are roots of I.*

Since there are an infinite number of distinct cycles in an RDG, the above criterion does not form the basis of an algorithm for checking convertibility. Later, we present theorems that do provide such a basis.

Let A be a (strongly connected) SARE. Affine transformations can be applied to the index sets of A's arrays such that the resulting RDG has a spanning tree whose arcs have uniform dependence maps associated with them. Such a transformation has been suggested by Delosme and Ipsen [12]. We now describe a similar procedure, based on their ideas. Choose a spanning tree of the RDG. Start from the root of the tree, and proceed down level by level, applying the affine transformation δ_{ij}^{-1} on the index set of the array a_i (i.e., $p \rightarrow D_{ij}^{-1}p - D_{ij}^{-1}d_{ij}$), where a_j is its parent[8]. All the

[7] In a digraph, a semicycle is a cycle in the underlying graph (i.e., a 'cycle' in the digraph where the arcs may be traversed independent of their sense)[15].

[8] The transformation δ_{ij}^{-1} is used instead of D_{ij}^{-1}, ensuring that integer lattice points map to integer lattice points.

appropriate dependence maps are updated after each transformation. This procedure is called a *tree conversion*. The tree is not unique.

Such a tree conversion is shown in Fig. 9. The linear parts of the dependence maps are shown

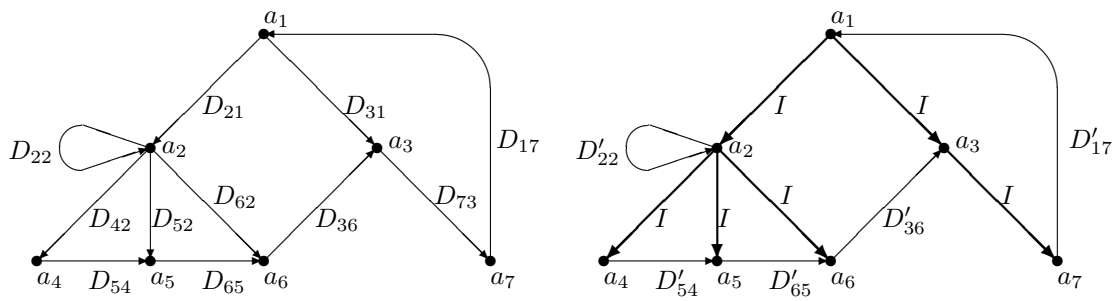


Figure 9: The RDG before (on the left) and after (on the right) the tree conversion.

near each arc. The array a_1 was chosen to be the root, and then the index set of the array a_2 is mapped by δ_{21}^{-1}. By doing so, the dependence map δ_{21} becomes uniform: its linear part becomes I as shown in the figure. The updating that should be done after this transformation is: $\delta_{22} \to \delta_{21}^{-1}\delta_{22}\delta_{21}$, $\delta_{42} \to \delta_{42}\delta_{21}$, $\delta_{52} \to \delta_{52}\delta_{21}$, $\delta_{62} \to \delta_{62}\delta_{21}$. The index set of a_2 also is updated. The next step is to change δ_{31} into a uniform dependence map. This is done by mapping the index set of a_3 by δ_{31}^{-1}. The same process is applied; this continues until the whole tree has uniform dependences associated with its arcs. The result also is shown in Fig. 9. The new linear parts which do not correspond to tree arcs are:

$$D'_{22} = D_{21}^{-1}D_{22}D_{21} \qquad D'_{54} = D_{21}^{-1}D_{52}^{-1}D_{54}D_{42}D_{21}$$
$$D'_{65} = D_{21}^{-1}D_{62}^{-1}D_{65}D_{52}D_{21} \quad D'_{36} = D_{31}^{-1}D_{36}D_{62}D_{21} \qquad D'_{17} = D_{17}D_{73}D_{31}$$

The translation parts are updated accordingly. The general affect of transforming index sets is as follows. Let there be a dependence map from a_j to a_i, namely $\delta_{ij}(p) = D_{ij}p + d_{ij}$. If the index set of a_i is transformed by $p \to Bp + b$, and the index set of a_j is transformed by $p \to Ap + a$, then the dependence map becomes: $\delta'_{ij}(p) = B(D_{ij}A^{-1}(p - a) + d_{ij}) + b$. The new linear part thus is $BD_{ij}A^{-1}$. In a tree conversion, an array's index set is transformed by the inverse of the composition of the dependence maps of the arcs in the path from the root of the tree to the node in the RDG. Suppose we have an arc from a_j to a_i labeled D, and the path from the root to a_j on the tree consists of the linear parts $A_1, A_2, \ldots, A_k$, (A_1 emanates from the root), and those for a_i are $B_1, B_2, \ldots, B_l$. Then the index set of a_j is transformed by the linear part $(A_k \cdots A_2 \cdot A_1)^{-1}$; the index set of a_i is transformed by the linear part $(B_l \cdots B_2 \cdot B_1)^{-1}$, and the linear part D becomes $B_1^{-1} \cdot B_2^{-1} \cdots B_l^{-1} \cdot D \cdot A_k \cdots A_2 \cdot A_1$.

We define the following sets for the next theorem:

A : The distinct linear parts of all cycle dependence maps in the RDG.

B : The distinct linear parts of all cycle dependence maps starting in a_i.

C : The distinct linear parts of all cycle dependence maps starting in a_i after the tree conversion T.

D : The distinct linear parts of all cycle dependence maps in the RDG after the tree conversion T.

E : All distinct compositions of linear parts of dependence maps after the tree conversion T.

F : The distinct linear parts of all semicycle dependence maps in the RDG.

G : The distinct linear parts of all semicycle dependence maps starting in a_i.

H : The distinct linear parts of all semicycle dependence maps starting in a_i after the tree conversion T.

J : The distinct linear parts of all semicycle dependence maps in the RDG after the tree conversion T.

Theorem 3.3 *If an n-dimensional SARE is convertible, then its sets $A, B, \ldots, J$ have only matrices which are roots of I.*

This theorem gives nine equivalent necessary conditions for convertibility. The most important one of these is the one which states that all compositions of the dependence maps of the RDG after any tree conversion are roots of I. This is because 1) the set of such compositions is enumerable, and 2) as will be seen later, the number of such compositions is finite.

3.2 Deciding a necessary condition for convertibility

Polynomials in the following discussion are assumed to have only rational coefficients. For any *primitive* r root of 1 (i.e., this number is not a k root of 1 for any $k < r$), its *degree* is $\phi(r)$ where degree means the minimum degree of a polynomial (called minimal polynomial) which has this number as a root, and $\phi(r)$ (the Euler function) is the number of relatively prime numbers to r which are less than or equal to it [2, 29]. Also, the minimum polynomial is the *cyclotomic polynomial* $\Phi_r(x)$ where

$$\Phi_r(x) = \prod_{\substack{\gcd(k,r)=1 \\ 1 \le k \le r}} (x - \xi^k), \quad \text{where} \quad \xi = e^{\frac{2\pi j}{r}}$$

Thus if $r_1 \neq r_2$ and both $\Phi_{r_1}(x)$ and $\Phi_{r_2}(x)$ divide $P(x)$ then $P(x)$ has all the roots of both cyclotomic polynomials. But the roots of these cyclotomic polynomials are distinct. (Because if for example $\xi_1^k = \xi_2^l$, then $\frac{l}{r_2} = \frac{k}{r_1}$. Since both are reduced ratios $r_1 = r_2$, which is a contradiction.) The degree of $P(x)$ thus is at least $\phi(r_1)+\phi(r_2)$. This also is true for any finite number of cyclotomic polynomials which divide $P(x)$. We thus have the following lemma.

Lemma 3.4.1 *Given k distinct numbers $r_1, r_2, \ldots, r_k \in \mathbf{N}$, and a polynomial $P(x)$, if the cyclotomic polynomials $\Phi_{r_i}(x)$ $1 \leq i \leq k$ divide $P(x)$, then the degree of $P(x)$ is at least $\sum_{i=1}^{k} \phi(r_i)$.*

The following theorem characterizes the numbers L such that an integral matrix D raised to the power L is I.

Theorem 3.4 *If $D \in \mathbf{Z}^{n\times n}$ and D is a primitive*[9] *L root of I, then $L = \mathrm{lcm}(r_j)$ where $\sum \phi(r_j) \leq n$. Moreover, $L \leq f(n)$, where $f : \mathbf{N} \rightarrow \mathbf{N}$.*

Proof. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of D. Each eigenvalue is a root of 1. Let $r_1, r_2, \ldots, r_k$ be the distinct values for which λ_j is a primitive r_i root of 1. From this it follows that $L = \mathrm{lcm}(r_1, r_2, \ldots, r_k)$. All eigenvalues are roots of the characteristic polynomial, and thus $\Phi_{r_i}(x)$ for all $1 \leq i \leq k$ divide the characteristic polynomial; the cyclotomic polynomial divides any polynomial which has a common root [2, § 4], [29, Thm. 2 pp. 63]. Thus, according to Lemma 3.4.1, $\sum_{i=1}^{k} \phi(r_i) \leq n$ (i.e., the sum of all the degrees of the cyclotomic polynomials for all r_i is less than or equal to the dimension n).

There is a monotonically increasing lower bound on ϕ. Therefore, given n, there is an upper bound on the numbers r_i such that $\phi(r_i) \leq n$, and thus there is an upper bound on the value of $\mathrm{lcm}(r_i)$. ■

For example, if we have a matrix $D \in \mathbf{Z}^{2\times 2}$ and it is a primitive L root of I, then $L = \mathrm{lcm}(r_j)$ where $\sum \phi(r_j) \leq 2$. Then $r_j \leq 6$, because if $r > 6$ then $\phi(r) > 2$. The possible r_i lists are: $(1,1), (1,2), (2,2), (3), (4), (6)$, and the least common multiples give the values that L can take: $1, 2, 3, 4, 6$.

The following two theorems are needed in what follows. The proof of the first can be found in [8], and the second in [29, pp. 52].

Theorem 3.5 [Burnside]. *Let G be a subgroup of $GL(n, \mathbf{C})$ of exponent*[10] *r. Then G is finite.*

[9] D is not a K root of I for any $K < L$.

[10] G is of exponent r if $x^r = 1$ for all $x \in G$.

Theorem 3.6 *Let G be a finite group of integral $n \times n$ matrices. Let $t_1 = n, t_2, t_3, \ldots, t_r$ be the distinct values assumed by the traces of the elements of G. Then the order of G divides $(n - t_2)(n - t_3) \cdots (n - t_r)$.*

From Thm. 3.6, it follows that $|G|$ must divide $(2n)!$; each t_i is the sum of n roots of unity. Consequently, $|t_i| \leq n$.

Theorem 3.7 *If $D_1, D_2, \ldots, D_k \in \mathbf{Z}^{n \times n}$, and all finite compositions of these matrices are roots of I, then there are a finite number of distinct matrices generated by these compositions. Moreover, this number is bounded by $h(n)$, where h: $\mathbf{N} \rightarrow \mathbf{N}$.*

Proof. The group generated by the D_i, denoted $< D_1, D_2, \ldots, D_k >$, is a subgroup of $GL(n, \mathbf{C})$ (the general linear group of nonsingular matrices over the complex field)[11].

From Thm. 3.4, $D_i^L = I$, for some constant L (take $L = f(n)!$ in Thm. 3.4, for example). The above mentioned subgroup therefore has a finite exponent. From Burnside's theorem (Thm. 3.5), it follows that this subgroup is finite.

Since $< D_1, D_2, \ldots, D_k >$ is finite and its members are integral $n \times n$ matrices, based on Thm. 3.6, its order divides $(2n)!$ as mentioned earlier, and therefore is bounded by a function of n. ■

For example, for 2×2 matrices, the number of matrices in the group must divide $(2 \cdot 2)! = 24$.

As mentioned previously, for computable SARE's, one of the eigenvalues of the linear part of every cycle dependence map must be 1. In those cases, the traces t_i of n-dimensional integral matrices which generate a finite group satisfy $t_i \geq -(n-2)$ and thus the order of the group divides $(2n-2)!$ (In case k of the eigenvalues are known to be 1, the order of the group divides $(2n-2k)!$). For the 2×2 case, the group has 1 or 2 matrices, the eigenvalues are either $1, 1$ or $1, -1$.

Consider the SARE after a tree conversion has taken place. The dependence maps resulting from a tree conversion have linear parts which generate a group that is similar to the group of linear parts of cycle dependence maps before tree conversion. The order of this group is thus 1 or 2, and the same eigenvalue pairs $1, 1$ or $1, -1$ are still the only possibilities for the linear parts of the dependence maps. From this, and previous theorems, we get the following corollary, which has been derived differently in [12].

Corollary 3.7.1 *Let A be a computable, convertible 2-dimensional SARE after tree conversion. Dependence maps that do not have I as their linear part all have the same linear part D. D's eigenvalues are 1 and -1.*

The number of cycle or semicycle dependence maps in the (strongly-connected) RDG is infinite. The number of compositions of dependence maps in the RDG after a tree conversion is infinite. But,

[11] Here composition is matrix product.

from Thm. 3.7, we see that the number of distinct linear parts is finite. It moreover is bounded by a function of n (where n is the dimension).

The algorithm below is derived from the foregoing. It determines, for a given n-dimensional SARE, whether or not the necessary conditions exist for it to be convertible. If they do *not* exist, then the algorithm returns *false*.

3.3 Algorithm 3.1

begin

extract the RDG from the system of recurrence equations;
perform a tree conversion on the SARE;
$N \leftarrow \{D|\delta(x) = Dx + d$ *is a dependence map* $\}$;
generate the group $G =< N >$, *such that after each generated member* $D \in G$ *do*

```
    {
    if    (D's characteristic polynomial is not in the set of the
          possible characteristic polynomials for the known dimension)
          then return(false) /* the SARE is not convertible[12] */
    if    (the current size |G| > h(n) )
          then return(false); /*h(n) being the known bound for the dimension of the SARE */
    }
return (true); /* necessary conditions are met */
```

end

This algorithm terminates because there is a bound on the number of matrices in the group (for a known dimension), as proved in Thm. 3.7. There also are only a constant number of possible characteristic polynomials for a known dimension, since the eigenvalues are L roots of 1, and according to Thm. 3.4, L is bounded. Another way to check for convertibility is to see if each matrix is a root of I, stopping if not. This is more difficult to do, but leads to the same result, if a matrix is not a root of I, then the number of matrices in the list grows until the group bound is exceeded, at which time the algorithm halts.

[12] This part of the algorithm is not needed. Its inclusion increases the algorithm's worst case complexity, but may improve its average case complexity (though this is not yet known).

3.4 Sufficient conditions for convertibility

Let SARE S have k dependence maps whose linear parts $D_1, D_2, \ldots, D_k$ are roots of I. Moreover, compositions of these linear parts are roots of I. According to Thm. 3.7 these matrices generate a finite group. The procedure below constructs a specific PLT for S[13]. It is used later in Thm. 3.8 to establish the sufficient condition for convertibility.

3.5 Procedure 3.1: Construct a PLT for an SARE.

1. *Let $G = \{D_i\}_{i=0}^{|G|-1}$ be the group generated by the D_i, where $D_0 = I$.*
2. *Construct*[14] *a point $p \in \mathbf{R}^n$ such that $D^T p \neq p, \forall D \in G - I$.*
3. *Construct matrix A_0, whose $row_i = p^T(D_i - I)$, $\forall D_i \in G - I$.*
4. *$\forall\ D_i \in G$, form the matrix $A_i = A_0 \cdot D_i^{-1}$.*
5. *For $0 \leq i < |G|$, define $F_i = \{x | A_i x \leq 0\}$.*
6. *Construct*[14] *a point $q \in F_0^i$*[15] *such that for $0 \leq i < |G|; 1 \leq j < |G|,\ [A_i]_j \cdot q \neq 0$.*
7. *Construct the linear constraints for P_i from those of F_i as follows. If $[A_i]_j \cdot q < 0$ then use $[A_i]_j \cdot x \leq 0$; else use $[A_i]_j \cdot x < 0$. These polytopes are the P_i.*
8. *Corresponding to part P_i, is the linear transformation $T_i = D_i^{-1}$.*

All index points are mapped to P_0 by the PLT constructed above. Moreover, since D_i is integer and is a root of I, D_i^{-1} is integer. Thus, all index points are mapped to integer points.

Lemma 3.8.1 *Let S be an n-dimensional SARE after a tree conversion, such that all the linear parts of its dependence maps, and all compositions of them, are roots of I. Let the partitioning of $\mathbf{R}^n$ constructed by Proc. 3.1 be $P = \{P_1, P_2, \ldots, P_{|G|}\}$. If the index sets of S's arrays are partitioned according to P, then, for any domain of computation C_i, any polytope $P_r \in P$, and any $k \in \mathbf{N}$, there exist domain size parameters such that $P_r \bigcap C_i$ contains a k^n hypercube.*

Lemma 3.8.2 *Let $H_1 \subset V$ be a K^n hypercube, where V is an n-dimensional vector space. If a nonsingular linear transformation T is applied to V, then H_1 is transformed into a parallelopiped which contains a k^n hypercube H_2, where $k = c \cdot K$ and c is a constant (depending only on T and n).*

[13] A detailed proof that this procedure defines a partitioning of $\mathbf{R}^n$ appears in [36].

[14] One procedure for such a construction can be found in [37].

[15] F_0^i is the interior of F_0.

The following theorem gives a sufficient condition for an SARE after a tree conversion to be PLT convertible. It uses the PLT construction defined by Proc. 3.1.

Theorem 3.8 *If an n-dimensional SARE after a tree conversion is such that all the linear parts of its dependence maps, and all their compositions, are roots of I, then it is PLT convertible.*

Proof. Since all linear parts and all their compositions are roots of I, according to Thm. 3.7, they generate a finite group G. We prove that the PLT defined by Proc. 3.1 leads to an SQURE. From the definition of SQURE it suffices to prove the following properties:

1. $\exists b \in \mathbf{R}$, such that $\forall a_{ij}$ all points in $\{x \in C_{ij}$[16] — $\max_{rs} d(x, C_{rs}) > b\}$[17] have the same dependence maps (from a_{ij} to some array), all of which are uniform.

2. For every cycle dependence map δ_{ij}, and $\forall\ k \in \mathbf{N}$, there exist domain size parameters, such that array a_{ij}'s domain of computation C_{ij} contains H, a k^n hypercube, and $\gamma_{ij}(H) > 1$.

To prevent confusion we denote the linear parts of a dependence map *before* the PLT by D'. The matrices constituting the group G are denoted by D. That is, for any D'_i (the linear part of some cycle dependence map before the PLT), or any D'_{ij} (the linear part of a direct dependence map from a_j to a_i), there exists a corresponding matrix D_k in G.

First, we prove property 1. Let $P = \{P_0, P_1, \ldots, P_{|G|-1}\}$ be the partition constructed by Proc. 3.1. All points are mapped to P_0 by this PLT. An array a_{ik} (i.e., that portion of a_i whose domain of computation is in P_k before the PLT is applied), has a domain of computation $C_{ik} = D_k^{-1}(C_i \bigcap P_k)$. For every point $x \in C_{ik} \bigcap P_0^i$ (P_0^i is the interior of P_0), we claim that $a_{ik}(x)$ depends on $a_{lj}(D_j^{-1}(D'_{li} D_k x + d_{li}))$, for all l such that a_i depends on a_l in the original SARE. This follows for two reasons. First, array element $a_{ik}(x)$ originally was array element $a_i(D_k x)$, where $D_k x \in P_k$, and according to the original SARE, $a_i(p)$ depends on $a_l(D'_{li} p + d_{li})$, for all l such that a_i depends on a_l in the original SARE. Second, if the point $D'_{li} \cdot p + d_{li} \in P_j$ before the PLT is applied, then applying the PLT transforms it to $D_j^{-1}(D'_{li} \cdot p + d_{li})$.

If $x \in P_0^i$ (i.e., x is in the interior of P_0), as is the case here, then $\forall i\ \ D_i x \in P_i$ (for detailed proof, see [37]). Consider the case when points $D'_{li} D_k x$, $D'_{li} D_k x + d_{li} \in P_j$. Then $D_j = D'_{li} D_k$ because if $D_j \neq D'_{li} D_k = D_r$ then $D'_{li} D_k x \in P_r \neq P_j$. By substitution, $\forall x \in C_{ik} \bigcap P_0^i$, $a_{ik}(x)$ depends on $a_{lj}(x + D_j^{-1} d_{li})$ (i.e., it is a uniform dependence map).

Let $\sigma_{\min} = \min_{D \in G}\{\sigma_D\}$, where σ_D is the minimum singular value of D. Consider the case when the distance from x to the boundary of P_0 is greater than $\max_l \|d_{li}\|/\sigma_{\min}$, where d_{li} are the translation

[16] The second index results from applying the PLT which partitions each domain of computation. C_{ij} denotes the domain of computation which results from applying the PLT to $C_i \cap P_j$. Other associated terms also take on this additional index.

[17] The distance from x to a point outside the domain of computation.

parts of all dependence maps of the arrays a_i on any other array a_l. If $y = D'_{li} D_k x \in P_j$, then the distance from y to the boundary of P_j is greater than $\max_l \|d_{li}\|$. Therefore, $y + d_{li} \in P_j$. Property 1 is satisfied when the distance from x to the boundary of P_0 is greater than $\max_l \|d_{li}\|/\sigma_{\min}$. Let t be a point closest to x on the boundary of P_0 (i.e., $\|x - t\|$ is the distance from x to the boundary of P_0). Let z be the lattice point closest to t but not in P_0 and let f be the lattice point outside P_0 (and thus outside C_{ik} because $C_{ik} \subset P_0$) closest to x (see Fig. 10). Surely

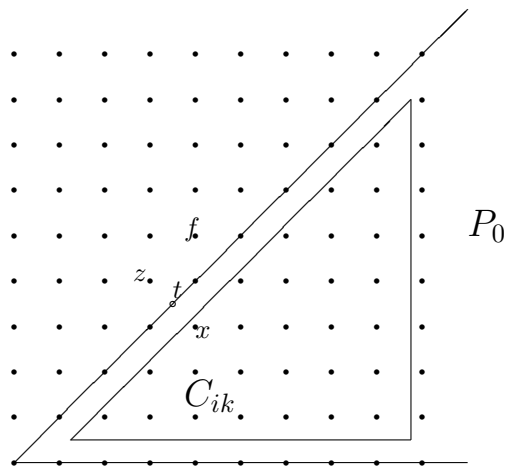


Figure 10: Distances from the boundary and border of a set.

$\|x-z\| \leq \|x-t\|+\|t-z\|$. But $\|t-z\| \leq \sqrt{n}$, thus $\|x-z\| \leq \|x-t\|+\sqrt{n}$. However $\|x-z\| \geq \|x-f\|$, thus $\|x - f\| \leq \|x - t\| + \sqrt{n}$. Therefore choosing $b = \sqrt{n} + \max_l \|d_{li}\|/\sigma_{\min}$ satisfies property 1 (because $b < \max_{rs} d(x, C_{rs}) \leq \|x - f\|$, and the distance from x to the boundary of P_0 is $\|x - t\|$).

Now, we prove property 2. According to property 1, all cycle dependence maps in C_{ij} (except for points 'near' the border) are uniform. Therefore δ_{ij} applies only a translation. It thus suffices to prove that $\forall\ k \in \mathbf{N}$, there exist domain size parameters such that C_{ij} contains h, a k^n hypercube. By Lemma 3.8.1, $\forall\ k_1 \in \mathbf{N}$, there exist domain size parameters such that $(C_i \bigcap P_j)$ contains h_1, a k_1^n hypercube. After applying the PLT, h_1 is transformed by a linear transformation to a convex polyhedron $H \subset C_{ij}$. According to Lemma 3.8.2, $\forall\ k_2 \in \mathbf{N}$, there exist domain size parameters such that H contains h_2, a k_2^n hypercube. Property 2 thus is satisfied. ■

3.6 Characterizing convertibility

We now are ready to state our main result.

Theorem 3.9 *An n-dimensional SARE is convertible if and only if after a tree conversion, the linear parts of the dependence maps generate a finite group.*

Proof. As proved in Thm. 3.3, the linear parts of the dependence maps and all their compositions are roots of I. Using Burnside's theorem (Thm. 3.5), the 'only if' part follows.

We now prove the 'if' part. If after a tree conversion the linear parts of the SARE's dependence maps generate a finite group, then every matrix in the group is a root of I. By Thm. 3.8, after the tree conversion, the SARE is PLT convertible. Since the tree conversion applies affine transformations to the index sets of the arrays, the original SARE is convertible. ■

This condition is decided by Alg. 3.1 (see page 22): it returns true if and only if the SARE is convertible.
From the above theorem, we have the following corollary:

Corollary 3.9.1 *The following statements are equivalent:*

1. *The SARE is convertible.*
2. *The linear parts of all cycle dependence maps in the RDG are roots of I.*
3. *The linear parts of all cycle dependence maps starting in a_i are roots of I.*
4. *The linear parts of all semicycle dependence maps in the RDG are roots of I.*
5. *The linear parts of all semicycle dependence maps starting in a_i are roots of I.*
6. *After a tree conversion, the linear parts of the dependence maps generate a finite group.*
7. *After a tree conversion, the linear parts of all cycle dependence maps starting in a_i are roots of I.*
8. *After a tree conversion, the linear parts of all cycle dependence maps in the RDG are roots of I.*
9. *After a tree conversion, all compositions of the linear parts of dependence maps are roots of I.*
10. *After a tree conversion, the linear parts of all semicycle dependence maps starting in a_i are roots of I.*
11. *After a tree conversion, the linear parts of all semicycle dependence maps in the RDG are roots of I.*

All the properties above impose an upper bound on the root, as proved in Thm. 3.4. The number of folds needed in statement 1 of the above corollary also is bounded, as proved in Thm. 3.1.

The above corollary provides many equivalent conditions for convertibility, not all of which are easy to decide. Alg. 3.1 decides if conditions (6,9) hold. The next corollary gives a property that is simpler to decide, in the special case of a two dimensional SARE. The "if" part of the corollary below, in a slightly different setting, is due to Delosme and Ipsen [12].

Corollary 3.9.2 *A computable 2-dimensional SARE is convertible if and only if all the nonuniform dependences after a tree conversion have the same linear part, and its eigenvalues are 1 and -1.*

Proof. Since the 'only if' part is proved in Cor. 3.7.1, we only have to prove the 'if' part. If there is only one linear part $D \neq I$, and D has eigenvalues 1 and -1, then D can be decomposed into its Jordan form: $F \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} F^{-1}$. Thus $D^2 = I$. From Thm. 3.9, the SARE is convertible. ■

For example, in Ex. 1A in § 2 there is only one linear part which is not I, and its eigenvalues are 1 and -1. As seen in Ex. 1B, it is convertible.

The following example illustrates the construction of a PLT for the 2-dimensional case.

3.7 Example 3

From Cor. 3.9.2, a computable and convertible 2-dimensional SARE has at most one nonidentity linear part D after a tree conversion. The eigenvalues of D are 1 and -1. For this group of matrices (namely D, I), we define the PLT:

$$P_0 = \{x | a^T x \leq 0\}, \quad P_1 = \{x | a^T x > 0\}, \quad T_0 = I, \quad T_1 = D$$

where $l = \{x | a^T x = 0\}$ is the line of all eigenvectors of the eigenvalue 1 of D (there is exactly one such line).

The above example is a special case of Proc. 3.1, since if we choose $p \in \mathbf{R}^2$ such that $D^T p \neq p$ and choose any eigenvector v of an eigenvalue 1 of D, we get $p^T(D-I)v = 0$. Therefore, $c \cdot p^T(D-I) = a^T$ for some constant c, because there is exactly one line l of eigenvectors for the eigenvalue 1, as mentioned before. The vector a can be chosen to be any nonzero vector which is perpendicular to an eigenvector of an eigenvalue 1 of D. Also, $T_1 = D^{-1} = D$.

3.8 Conversion of an SARE to an SQURE

Given a convertible SARE S, procedure 3.2 below converts S into an SQURE.

3.9 Procedure 3.2: Convert an SARE to an SQURE.

1. Perform a tree conversion on S.

2. Invoke Proc. 3.1 with $G = \{D_i\}_{i=0}^{|G|-1}$, the group generated by the linear parts of the direct dependence maps. This procedure produces a set P of polytopes $\{P_i\}_{i=0}^{|G|-1}$, and a set of linear transformations $\{T_i = D_i^{-1}\}_{i=0}^{|G|-1}$ which define a PLT.

3. Partition each domain of computation C_i as follows:

$$\forall\, 0 \le j \le |G| - 1 \quad C_{ij} = C_i \bigcap P_j.$$

   Each recurrence equation is split into up to $|G|$ equations (some C_{ij}s may be empty).

4. Invoke Proc. 3.3, partitioning $\mathbf{R}^n$ into $P^{ki} = \{\delta_{ki}^{-1}(P_j)\}_{j=0}^{|G|-1}$, for all direct dependences δ_{ki}.

5. For each array a_i, partition $\mathbf{R}^n$ by super-imposing all the partitions P^{ki} computed above.

6. Each C_{ij} is further partitioned into $C_{ij}^{(0)}, C_{ij}^{(1)}, \ldots, C_{ij}^{(h)}$ (the corresponding equations also are split) by restricting to C_{ij} the partition of $\mathbf{R}^n$ for a_i in step 5. This partition satisfies:

$$\forall\, 0 \le t \le h, \;\; \forall k, \;\; \delta_{ki}(C_{ij}^{(t)}) \subset P_{s(i,j,k,t)} \quad \text{for some} \;\; 0 \le s(i,j,k,t) < |G|.$$

   We denote by $C_{ij}^{(0)}$ the domain of computation that satisfies: $\forall k, \;\; D_{ki}(C_{ij}^{(0)}) \subset P_{s(i,j,k,0)}$. This step ensures that each $C_{ij}^{(t)}$ is mapped by any dependence map of a_i on some array a_k to only one part in partition P.

7. For each equation that computes a_i in the domain of computation $C_{ij}^{(t)}$ do

   (a) update the domain of computation as follows: $C_{ij}^{(t)} \to D_j^{-1} C_{ij}^{(t)}$ (update the boundaries and the lattice[18]),

   (b) rename the array on the left side of the equation: $a_i \to a_{ij}$,

   (c) for each term $a_k(\delta_{ki}(p))$ on the right side of the equation, given that $\delta_{ki}(C_{ij}^{(t)}) \subset P_l$ (i.e., $s(i,j,k,t) = l$), do

      i. rename the array: $a_k \to a_{kl}$,

      ii. update and rename the dependence map[19]:

$$\delta_{ki} \to \delta_{kl,ij}, \;\; \text{where } D_{kl,ij} = D_l^{-1} D_{ki} D_j, \quad d_{kl,ij} = D_l^{-1} d_{ki}.$$

[18] The lattice in $\mathbf{Z}^n$ is based on n independent vectors. We update the lattice by multiplying these vectors by D_j^{-1}.

[19] Superscripts may be used to differentiate names in different equations or different terms in the same equation.

The matrices D_l, D_j are members of the group G, while D_{ki}, d_{ki} are the original linear part and translation parts of δ_{ki}. (The above follows from the choice of $T_i = D_i^{-1}$ in Proc. 3.1.)

According to the construction of the PLT, for $C_{ij}^{(0)}$, we get $D_{ki} \to I$.

Proc. 3.2 converts an SARE by a specific PLT constructed in its first two steps. If one wants to convert the SARE by another PLT, the first two steps are omitted and step 7 is changed accordingly (for more detail, see [35]).

The procedure below uses a the notion of *fundamental regions*, introduced by Coxeter [7]:

> *"We have now reached a suitable place to introduce the important notion of a* fundamental region*. For any group of transformations of a plane (or of space), this means a region whose transforms just cover the plane (or space), without overlapping and without interstices. In other words, every point is equivalent (under the group) to some point of the region, but no two points of the region are equivalent unless both are on the boundary."*

3.10 Procedure 3.3: Partition $\mathbf{R}^n$ into $\{P_j^{ki}\}$.

Let $\delta_{ki}(p) = D_{ki}p + d_{ki}$, $\delta_{ki}^{-1}(p) = D_{ki}^{-1}p - D_{ki}^{-1}d_{ki}$, and $\{F_i\}_{i=0}^{|G-1|}$ be the parts constructed in Proc. 3.1 step 5, where $F_i = \{x|A_i x \leq 0\}$. Let $q \in F_0^i$ be the point constructed in Proc. 3.1 step 6.

1. Construct[20] $q_r = D_r q \; \forall D_r \in G$.
2. Construct the linear constraints for P_j^{ki} from those of F_j as follows.

   If $[A_j]_l(D_{ki}^{-1}q) < 0$
   then use $[A_j]_l(x + D_{ki}^{-1}d_{ki}) \leq 0$
   else use $[A_j]_l(x + D_{ki}^{-1}d_{ki}) < 0$.
3. For every part P_j^{ki} constructed in step 2, find the subscript r that satisfies $P_j^{ki} = \delta_{ki}^{-1}(P_r)$ by finding the vector q_r that satisfies $q_r = D_{ki}q_j$.

The above construction is valid since $D_{ki}^{-1} \in G$, and thus $D_{ki}^{-1}(F_j) = F_l$ for some l. If we let j go through 0 to $|G|-1$, we get that l goes through 0 to $|G|-1$ too. Thus $\{D_{ki}^{-1}(F_j)\}_{j=0}^{|G|-1} = \{F_j\}_{j=0}^{|G|-1}$. The part P_j is constructed in Proc. 3.1 step 7 from F_j by using the point $q \in F_0^i$. Thus, the point $D_{ki}^{-1}P_j$ is constructed from $D_{ki}^{-1}F_j = \{x|A_j D_{ki} x \leq 0\}$, by using the point $D_{ki}^{-1}q$. The term $x + D_{ki}^{-1}d_{ki}$ replaces x, since the translation part of δ_{ki}^{-1} is $-D_{ki}^{-1}d_{ki}$.

The points $\{q_r\}_{r=0}^{|G|-1}$ have a one-to-one correspondence with the matrices $\{D_r\}_{r=0}^{|G|-1}$. This follows from the fact that $q \in F_0^i$, and F_0 is a fundamental region [36]. Thus, the test $D_{ki}q_j = q_r$ in the last step ensures that $D_{ki}F_j = F_r$. And thus $\delta_{ki}P_j^{ki} = P_r$, or equivalently $P_j^{ki} = \delta_{ki}^{-1}(P_r)$.

[20] This is done once for all direct dependences.

3.11 Generalizing to non-strongly connected SAREs

We have assumed that the SARE is strongly connected. If not, the results derived cannot be used. We however can modify the definition of n-dimensionality so that the same results follow in this more general setting. The modification is as follows. In the definition of an n-dimensional SARE, and all the lemmata, theorems, and corollaries, the word 'cycle' is replaced by 'semicycle', and 'directed path' is replaced by 'semipath'. Also, in the definition of 'tree conversion', the tree is a spanning tree in the *underlying graph*[21] of the RDG. The only theorems which are not correct for this case are Cor. 3.7.1 and Cor. 3.9.2; these special-case corollaries require one of the eigenvalues of every linear part to be 1. This property need not hold when the linear parts are associated with an RDG that is not strongly connected.

4 Conclusions

A system of uniform recurrence equations typically can be mapped linearly into spacetime so that interprocessor communication requires only a fixed amount memory, and fixed-length interconnections. There is no such linear mapping into spacetime for systems of *affine* recurrence equations that are not uniform.

The first step in eliminating non-local connections by folding, was made by Choffrut and Culik [6]. They treat a specific I/O problem: folding the systolic array so processing elements on the periphery of the array that produce output are neighbors of, and provide input to, processing elements on the periphery of the array that consume input. Choffrut and Culik deal with folds that can eliminate 2-dimensional reflections and/or rotations, whereas this paper deals with 1) any affine dependence (with a nonsingular linear transformation) between array elements, and 2) arrays of any dimension.

Delosme and Ipsen [12] also consider systems of affine recurrence equations. Their paper treats systems of affine recurrence equations such that:

1. The range of any cycle dependence map must be a translation of its domain.

2. All arrays in the system of recurrence equations must have the same domain of computation.

From these properties, they show that the linear parts of the dependence maps are roots of I. Other results in their paper, for systems of affine recurrence equations, apply only to 2-dimensional systems. Specifically, they show that in a 2-dimensional system of affine recurrence equations, there exists only one linear part of a cycle dependence map (or any dependence map after a tree conversion), which is not I in the reduced dependence graph.

[21] The *underlying graph* of a digraph is the graph resulting from the digraph if the orientation of the arcs is ignored[13].

Recently, Zheng and Kiaei [38] have contributed to the understanding of affine recurrence equations that are not uniform. They define a subclass of affine recurrence equations called Directional Affine Recurrence Equations, and deal with the non-uniformities via a Multi-Rate Array.

Our work is inspired primarily by that of Delosme and Ipsen. We formulate a 'generalized fold', and provide the following:

- a proof that a system of affine recurrence equations can be converted, by a generalized fold, to an equivalent system of quasi-uniform recurrence equations if and only if the linear parts of the cycle dependence maps in the reduced dependence graph generate a finite group. (Cor. 3.9.1 gives many equivalent conditions.)
- an algorithm for deciding if a system of affine recurrence equations can be so converted;
- a procedure for converting a system of affine recurrence equations to an equivalent system of quasi-uniform recurrence equations.

Where linear embeddings fail, a generalized fold may succeed in enabling a VLSI array implementation.

The finiteness of the SARE's associated group follows from a theorem by Burnside, requiring it to have a matrix representation. Our characterization of convertible systems brings together classical ideas in algebraic geometry (fundamental regions —see page 29), number theory (§3.2), and matrix representations of groups (Theorems 3.8 and 3.9). While the complete proof of this characterization is complex, the characterization itself is simple, suggesting that these classical mathematical ideas are well chosen for this difficult problem in array design.

Currently, systems of affine recurrence equations that are not uniform are rarely used by algorithmic researchers. We hope that the results in this paper encourage researchers to use these systems.

4.1 Acknowledgement

We wish to thank Professor Morris Newman for pointing out the existence of Burnside's theorem.

References

[1] U. Banerjee, S-C Chen, D. J. Kuck, and R. A. Towle. Time and parallel processor bounds for Fortran-like loops. *IEEE Trans. Computers*, C-28(9):660–670, Sep 1979.

[2] E. R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill Book Company, New York, 1968.

[3] Peter R. Cappello. *VLSI Architectures for Digital Signal Processing.* PhD thesis, Princeton University, Princeton, NJ, Oct 1982.

[4] Peter R. Cappello and Kenneth Steiglitz. Unifying VLSI array design with linear transformations of space-time. In Franco P. Preparata, editor, *Advances in Computing Research*, volume 2: VLSI theory, pages 23–65. JAI Press, Inc., Greenwich, CT, 1984.

[5] Marina C. Chen and Carver Mead. Concurrent algorithms as space-time recursion equations. In S-Y Kung, H. J. Whitehouse, and Thomas Kailath, editors, *VLSI & Modern Signal Processing.* Prentice-Hall, Englewood Cliffs, 1985.

[6] C. Choffrut and K. Culik II. Folding of the plane and the design of systolic arrays. *Information Processing Letters*, 17:149–153, 1983.

[7] H. S. M. Coxeter. *Regular Polytopes.* Dover Publications, Inc., New York, third edition, 1973.

[8] C. W. Curtis and I. Reiner. *Representation Theory of Finite Groups and Associative Algebras.* Interscience Publishers (John Wiley & Sons), New York, 1962.

[9] Alain Darte, Leonid Khachiyan, and Yves Robert. Linear scheduling is close to optimal. In José Fortes, Edward Lee, and Teresa Meng, editors, *Application Specific Array Processors*, pages 37–46. IEEE Computer Society Press, August 1992.

[10] J.-M. Delosme and I. C. F. Ipsen. Parallel solution of symmetric positive definite systems with hyperbolic rotations. *Linear Algebra Appl.*, 77:75–111, 1986.

[11] Jean-Marc Delosme and Ilse Ipsen. Efficient systolic arrays for the solution of toeplitz systems: An illustration of a methodology for the construction of systolic architectures in vlsi. In W. Moore et al., editor, *Systolic Arrays*, pages 37–46. Adam Hilger, 1987.

[12] Jean-Marc Delosme and Ilse C. F. Ipsen. Systolic array synthesis: Computability and time cones. Technical Report Yale/DCS/RR-474, Yale, May 1986.

[13] Shimon Even. *Graph Algorithms.* Computer Science Press, Inc, Rockville, MD, 1979.

[14] José A. B. Fortes, King-Sun Fu, and Benjamin W. Wah. Systematic approaches to the design of algorithmically specified systolic arrays. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 300–303, Tampa, 1985.

[15] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1972.

[16] Richard M. Karp, Richard E. Miller, and Shmuel Winograd. Properties of a model for parallel computations: Determinacy, termination, queueing. *SIAM J. Appl. Math*, 14:1390–1411, 1966.

[17] Richard M. Karp, Richard E. Miller, and Shmuel Winograd. The organization of computations for uniform recurrence equations. *J. ACM*, 14:563–590, 1967.

[18] H.-T. Kung. Why systolic architectures. *Computer*, 15(1):37–45, Jan 1982.

[19] H.-T. Kung and Charles E. Leiserson. Algorithms for VLSI processor arrays. In *Introduction to VLSI Systems*, pages 271–292. Addison-Wesley Publishing Co, Menlo Park, CA, 1980.

[20] S-Y Kung. On supercomputing with systolic/wavefront array processors. *Proceedings of the IEEE*, July 1984.

[21] Sun-Yuan Kung and Y. H. Hu. A highly concurrent algorithm and pipelined architecture for solving toeplitz systems. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 31(1):66–74, Feb. 1983.

[22] Charles E. Leiserson, Flavio M. Rose, and James B. Saxe. Optimizing synchronous circuitry by retiming. In *Proc. Third Caltech Conf. on VLSI*, Rockville, MD, 1983. Computer Science Press.

[23] Charles E. Leiserson and James B. Saxe. Optimizing synchronous systems. In *Proc. IEEE 22nd Annual Symp. Foundations of Computer Science*, Oct 1981.

[24] R. G. Melhem and W. C. Rheinboldt. A mathematical model for the verification of systolic networks. *SIAM J. Computing*, 13(3):541–565, Aug. 1984.

[25] Willard L. Miranker and Andrew Winkler. Spacetime representations of computational structures. *Computing*, 32:93–114, 1984.

[26] Dan I. Moldovan. On the analysis and synthesis of VLSI algorithms. *IEEE Trans. Comput.*, C-31:1121–1126, Nov. 1982.

[27] Dan I. Moldovan. On the design of algorithms for VLSI systolic arrays. *Proc. IEEE*, 71(1):113–120, Jan. 1983.

[28] Dan I. Moldovan and José A. B. Fortes. Partitioning and mapping algorithms into fixed systolic arrays. *IEEE Trans. Comput.*, C-35(1):1–12, Jan. 1986.

[29] M. Newman. Matrix representation of groups. In *Applied Mathematics Series - 60*. Institute for Basic Standards, National Bureau of Standards, Washington D.C. 20234, July 1968.

[30] Patrice Quinton. Automatic synthesis of systolic arrays from uniform recurrent equations. In *Proc. 11th Ann. Symp. on Computer Architecture*, pages 208–214, 1984.

[31] Sanjay Rajopadhye, Lap Mui, and Sayfe Kiaei. Piecewise linear schedules for recurrence equations. In Kung Yao, Rajeev Jain, Wojtek Przytula, and Jan Rabaey, editors, *VLSI Signal Processing V*, pages 375–384. IEEE, October 1992.

[32] Sailesh K. Rao. *Regular Iterative Algorithms and Their Implementation on Processor Arrays*. PhD thesis, Stanford University, October 1985.

[33] Weijia Shang and José A. B. Fortes. Time optimal linear schedules for algorithms with uniform dependencies. *IEEE Trans. Comput.*, 40(6):723–742, June 1991.

[34] Jeffrey D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, Inc, Rockville, MD 20850, 1984.

[35] Yoav Yaacoby. *Computing Systems of Affine Recurrence Equations on Processor Arrays*. PhD thesis, University of California, Santa Barbara, Santa Barbara, CA, May 1988.

[36] Yoav Yaacoby, Peter R. Cappello, D. Witt, and K. C. Millett. Computing a fundamental region for a finite matrix group acting on a Euclidean space. Technical Report 4, Dept. Computer Science, UCSB, Santa Barbara, CA 93106, Feb. 1988.

[37] Yoav Yaacoby, Peter R. Cappello, D. Witt, and K. C. Millett. Computing a fundamental region for a finite matrix group acting on a Euclidean space. **Submitted to** *SIAM J. Computing*, February 1988.

[38] Yuepeng Zheng and Sayfe Kiaei. Multi-rate transformation of directional affine recurrence equations. In Luigi Dadda and Benjamin Wah, editors, *Proc. Int. Conf. Application-Specific Array Processors*, pages 392–403. IEEE Computer Society, Venice, ITALY, October 1993.

[39] Xiaoxiong Zhong and Sanjay Rajopadhye. Optimal parallel schedules for uniform recurrence equations. Computer science, University of Oregon, Eugene, 1992.