A Hierarchical Classification Scheme to Derive Interprocess Communication in Process Networks

Alexandru Turjan Bart Kienhuis Ed Deprettere
Leiden Institute of Advanced Computer Science (LIACS),
Leiden, The Netherlands
e-mail: {aturjan,kienhuis,edd}@liacs.nl

Abstract

The Compaan *compiler automatically derives a Process Network (PN) description from an application written in Matlab. The basic element of a PN is a Producer/Consumer (P/C) pair. Four different communication patterns for a P/C pair have been identified and the complexity of communication structure differs depending on the communication pattern involved. Therefore, in order to obtain cost-efficient process networks our compiler automatically identifies the communication pattern of each P/C pair. This problem is equivalent to integer linear programming and thus in general can not be solved efficiently. In this paper we present simpler techniques that allow to classify the interprocess communication in a PN. However, in some cases those techniques do not allow to find an answer and therefore, an ILP test has still to be applied. Thus, we introduce a hierarchical classification scheme that correctly classifies the interprocess communication, but uses dramatically less integer linear programming. In only 5% of the cases to classify, we still relay on integer linear programming while in the remaining 95%, the techniques presented in this paper are able to classify a case correctly.*

1 Introduction

The aim of the *Compaan* compiler [3, 7] is to automatically derive Process Networks (PN) descriptions from applications written in a standard language like C or Matlab. A PN expresses the application in terms of *distributed memory* and *distributed control* [7]. The distributed memory and control are essential when programming a heterogeneous multiprocessor architecture like the Virtex-Pro from Xilinx [12], the *Picochip* from PicoChip [5], or the SpaceCAKE architecture from Philips [8]. Distributed memory means that not all components read and write to the same memory causing memory bottlenecks. Instead, components exchange locally only data that other components require. Distributed control means that each component can make progress on its own; it is not under the control of some global controller, which fits the loosely coupled components.

In this paper, we investigate a particular problem in the Compaan compiler regarding the ability to classify the interprocess communication in a PN. Within Compaan, we map the static arrays not onto some shared memory structure, but onto a distributed communication structure taking care of the interprocess communication. We have already shown that four types of communication exists, each exhibiting different requirements when realizing them in hardware of software [10]. Within the Compaan compiler, we need to establish certain characteristics of each communication between a Producer and Consumer process. For this purpose, we have already developed and presented in [11] two integer linear programming (ILP) tests i.e., *Multiplicity Test* (MT) and *Reordering Test* (RT). These tests correctly classify each communication structure to one of four possible types. Due to the NP-Complete character of ILP, MT and RT are usually time consuming and memory intensive. Therefore, in this paper we present techniques based on polynomial algorithms (like Smith and Hermite normal form [4]) that allow to do the classification of the interprocess communication. However, in some cases those techniques do not allow to find correct answer and therefore, an ILP test has still to be applied. Thus, we introduce a hierarchical classification scheme that correctly classifies the interprocess communication, but uses dramatically less integer linear programming. In only 5% of the cases, we still relay on integer linear programming while in the remaining 95%, the tests presented in this paper are able to make a correct classification.

2 Linearization

In the original Matlab program, data is being communicating between assignment statements over static memory arrays. To obtain a PN the Matlab code is partitioned into a number of processes that are accessing those static arrays. Therefore, we need to map the static arrays onto some distributed communication structure. We do not want to map the static arrays onto shared memory, which is the typical approach, as it contradicts with the desirable notion of distributed memory. Instead, we would like to replace the static arrays by a truly distributed communication structure. This transformation step in Compaan is called *Linearization* [6].

2.1 Producer/Consumer Model

Compaan creates from the Matlab program a Polyhedral Reduced Dependence Graph (PRDG). In this graph, each edge represents communication between a producer and a consumer node that needs to be linearized. Each such edge can be abstracted to a simple Producer/Consumer (P/C) pair, where the Producer and the Consumer have parameterized polyhedral iteration domains that are related through an affine transformation. Depending on this transformation, described by a mapping matrix M and the structure of the for-loops as represented in the original Matlab program, we have found that four different kinds of communication can be distinguished in PNs. For a more detailed description of P/C pairs, their iteration domains and mapping functions the reader is referred to [9].

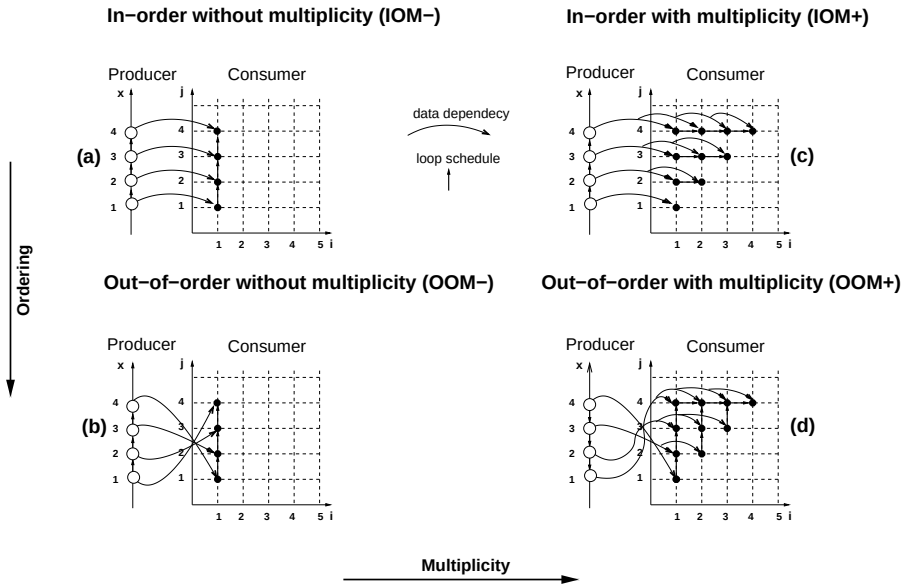

Figure 1. The four types of communication of a static array in a Producer/Consumer pairs.

These four types (patterns) of communicating data between Producer and Consumer are represented graphically in Figure 1. They are determined by the ordering of the iterations at the Producer and the Consumer processes (in order/out of order) and whether a token has *multiplicity*, which means that a token that is sent, is read more then once at the Consumer side. In Figure 1, we show iterations as small circles. These iterations are ordered as indicated by the small arrows representing the loop schedule of the for-loops in the original Matlab program. According to the existent P/C data-dependency, we also show the relationship between the production of data at an iteration and the consumption of that data at one or more iterations. Each communication type has been given a name. For example, the In-order without multiplicity case is given the abbreviated name IOM-.

When communication takes place in order, the distributed communication structure is a FIFO channel. In case of communication out of order, the *Extended Linearization Mechanism* (ELM) is required, which has been introduced in [10]. In that paper, we give the components of the ELM, that are a controller and reordering memory and show how to derive the control used to perform the reordering. In [9], we explain how to derive the life-time control of

a token, needed to determine when to release a token when multiplicity is involved. This control can be added to the ELM or to a FIFO channel. The ELM is needed for the models OOM- and OOM+. The other two models, IOM- and IOM+, use a FIFO channel. The implementations of the linearization models described above increase in their complexity (both Hardware and Software), from IOM- to OOM+. Of the four models identified, OOM+ is the most expensive linearization to be realized. It is also the generic linearization since it subsumes all the other three linearization.

2.2 Definition of Multiplicity and Re-ordering

We now give a formal definition of a P/C pair, multiplicity and reordering.

Definition 1 *A P/C pair is a tuple* $< \mathcal{C}(p), f, P(p), \prec >$*, where* $\mathcal{C}(p) \subset R^n$ *is a parameterized polytope,* $f : R^n \rightarrow R^m$ *is an affine function,* $P(p) = f(\mathcal{C}(p) \cap Z^n)$*, and* $\prec$ *is the lexicographical order.*

The set of the integer points inside the parameterized polytope $\mathcal{C}(p)$ is called the *Consumer domain*, or the *Consumer iteration space*. The affine function f is specified according to the data dependencies, which is represented by an $(m \times n)$ integral matrix M, and an m-dimensional offset vector O, i.e., $f(x) = Mx + O$. To simplify the further presentation, we assume that $O = 0$ such that f is represented only the matrix M. Depending on M, and on the schedule of producing and consuming data as given by the lexicographical order, four different kinds of P/C pairs can be distinguished based on the following two definitions:

Definition 2 *A P/C pair is* ***in-order*** *iff the mapping preserves the order, i.e. every two Consumer iteration points* $x_1 \prec x_2$ *are mapped onto two Producer iteration points* $(y_1 = Mx_1)$ *and* $(y_2 = Mx_2)$ *such that* $y_1 \preceq y_2$*. If the P/C pair is not in order it is called* **out-of-order**.

Definition 3 *A P/C pair is* **without multiplicity** *iff the mapping* $M : (C \cap Z^n) \rightarrow P$ *(i.e., the mapping M restricted to the Consumer domain) is injective. Otherwise we say that the P/C pair is* **with multiplicity**.

According to the previous definitions, *any* arbitrary P/C pair in a static nested loop program belongs to one of the four types given in Figure 1. This is not specific to the Compaan compiler, but to any static nested loop program.

3 Related Work

We have already presented a solution to determine the type of linearization in [10]. In that paper, we presented a technique based on the Ehrhart theory [1] to determine the ordering and multiplicity as pseudo polynomial expressions. By comparing the pseudo polynomial expressions, we could determine if the schedule of producing and consuming data are respectively the same, indicating in-order behavior. We could also determine if the pseudo polynomial was larger than one, indicating that multiplicity was involved. Although the presented procedure gives a definitive answer for the type of linearization, the complexity of the pseudo polynomial calculations, the comparison of the pseudo polynomials, and the fact that the software implementing Ehrhart was not able to always derive a pseudo polynomial, made the Ehrhart approach unsuitable for implementation in Compaan.

In [11], we presented two tests used to classify inter-process communication: the Reordering test and the Multiplicity test. Both tests are based on solving the problems given below using integer linear programming. If a solution exists for the Multiplicity Problem (MP), it means that multiplicity is involved. If a solution exists for the Reordering Problem (RP), it means that reordering is involved.

$$\mathbf{MP}: \begin{cases} x \in (\mathcal{C}(p) \cap Z^n), \\ y \in (\mathcal{C}(p) \cap Z^n), \\ x \neq y, \\ Mx = My. \end{cases} \qquad \mathbf{RP}: \begin{cases} x \in (\mathcal{C}(p) \cap Z^n), \\ y \in (\mathcal{C}(p) \cap Z^n), \\ x \prec y, \\ My \prec Mx. \end{cases} \tag{1}$$

Using both tests, we can always correctly classify a P/C pair, but at the expense of solving many integer linear programming systems, which is time consuming and memory intensive. Although in principle suitable for Compaan, a reduction in the number of systems to solve with integer linear programming is needed.

4 Solution Approach

Instead of solving integer linear programing systems, we want to exploit certain characteristics of the mapping matrix to classify a P/C pair to one of the four types. Just consider, as an example, the simple P/C pair given in Figure 2. In this example, the order of producing and consuming data are exactly the same and the mapping matrix is: $M = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{smallmatrix}\right]$. Suppose now that we change the Consumer such that it consumes tokens in a different order than they are produced. This can be achieved by interchanging the indices of the array a (from $a[x, y]$ to $a[y, x]$) at the Consumer as shown in Figure 3. The Producer and the Consumer polytopes stay the same, while the index change is reflected only in the affine mapping matrix that has become $M = \left[\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right]$.

Apparently, the order change from an in-order case to an out-of-order case is completely reflected in the mapping matrix. Therefore, by analyzing the mapping between Producer and Consumer, we want to decide whether out-of-order communication appears or not. Similarly, we want to say whether multiplicity is involved based on the structure of the mapping matrix. For that purpose, we present in the remainder of this paper three properties that needs to be checked to classify a P/C pair.

```
for i=1:1:N,
  for j=1:1:N,
    a[i,j] = fp();
  end
end
```

```
for x=1:1:N,
  for y=1:1:N,
    fc(a[x,y]);
  end
end
```

Figure 2. An In-order P/C pair

```
for i=1:1:N,
  for j=1:1:N,
    a[i,j] = fp();
  end
end
```

```
for x=1:1:N,
  for y=1:1:N,
    fc(a[y,x]);
  end
end
```

Figure 3. An Out-of-order P/C pair

Although we can classify most of the P/C pairs using the properties only, there are some cases that cannot be detected correctly. In those cases, we still require the use of the Multiplicity or Reordering test. These tests, together with the tests that check the presented properties, will lead to a hierarchical classification scheme. This scheme classifies correctly each P/C pair, but uses only in a fraction of the time integer linear programming compared to when we would use only the Reordering or Multiplicity tests.

5 Order and Multiplicity detection using the P/C mapping

Using only the mapping matrix, we want to know if re-ordering takes place or whether multiplicity is involved. We first look at properties that determine if multiplicity is involved, followed by properties that indicate whether re-ordering is involved. When taking about the *mapping*, we referring to the mapping matrix of P/C pair, but with the offset vector *O* removed (last column) as a translation does not effect the order or multiplicity of a P/C pair.

5.1 Multiplicity

A necessary condition for avoiding multiplicity is that the mapping should be injective, i.e., a one-to-one relationship exists between the iteration points from the Producer and a Consumer. Otherwise the mapping is bijective, and multiplicity is involved as a produced token can be consumed by more than one Consumer iterations. This condition is not conclusive, as there are particular cases where the mapping is not injective, but multiplicity does not appear. To determine whether or not the mapping defined by a matrix is injective can be done easily; the matrix should be full column rank, a condition we check using the Smith Normal Form [4].

5.2 Reordering

A necessary condition for avoiding re-ordering is that the mapping is a lower triangular matrix with positive entries on the diagonal. This leads to the definition of *Property1* as follows:

Property 1 *Consider a P/C pair with a mapping described by the matrix M. If M is an integral $(n \times n)$ lower triangular matrix, with positive diagonal elements, then the P/C pair is in-order.*

If this property holds, we call a mapping matrix an *in-order transformation*. If a transformation does not respect this property, we say it is an *out-of-order transformation.*

Proof: Firstly, we prove that any two Consumer points A and B, where point A is lexicographically smaller than point B (represented as $A \prec B$), are mapped by M into two Producer points $A^{'}$ and $B^{'}$ such that $A^{'} \prec B^{'}$. We take two Consumer points $A = (x_1, ..., x_n)$ and $B = (y_1, ..., y_n)$ such that $A \prec B$; i.e., there exists an $l \in Z$ where $1 \leq l < n$ such that $\forall s \in Z$ such that $1 \leq s \leq l$ it follows that $x_s = y_s$ and $x_{l+1} < y_{l+1}$. Assume that the mapping matrix has the shape:

$$\mathbf{M} = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}.$$

Two points will by mapped through the mapping matrix M into the next distinct Producer points as the mapping is bijective: $A^{'} = MA = (p_1, p_2, ..., p_l, S + a_{l+1,l+1} \cdot x_{l+1}, ...)$ and $B^{'} = MB = (p_1, p_2, ..., p_l, S + a_{l+1,l+1} \cdot y_{l+1}, ...)$, where $p_t = \sum_{i=1}^{t} a_{ti} \cdot x_i$, and $S = \sum_{i=1}^{l} a_{l+1,i} \cdot x_i$. Because $x_{l+1} < y_{l+1}$ and $a_{l+1,l+1} > 0$ it follows that $a_{l+1,l+1} \cdot x_{l+1} < a_{l+1,l+1} \cdot y_{l+1}$, and we conclude $A^{'} \prec B^{'}$.

Now let us take 2 consecutive Consumer points $A \prec B$. As we showed, they will be mapped into $A^{'}$ and $B^{'}$ such that $A^{'} \prec B^{'}$. To stay in-order, we have to show that at the Producer side $A^{'}$ and $B^{'}$ are also consecutive. Assuming that $A^{'}$ and $B^{'}$ are not consecutive, we will establish a contradiction here. We take a point $C^{'}$ at the Consumer side such that $A^{'} \prec C^{'} \prec B^{'}$. Because M is bijective, there exists a C at the Producer side such that $C^{'} = MC$. Since A and B are consecutive, it follows that either $C^{'} \prec A^{'} \prec B^{'}$ or $A^{'} \prec B^{'} \prec C^{'}$. Knowing that M preserves the order of the points, C can not be between A and B. This contradicts the assumption we made; thus $A^{'}$ and $B^{'}$ are consecutive at the Producer side. □

Property1 describes a necessary condition for in-order communication, under the assumption that the dimensions of the Producer and Consumer iteration domains are equal. We can generalize this property to the case when the dimension of the Producer is bigger or equal with the dimension of the Consumer.

Property 2 *Consider a P/C pair with matrix M. Let m and n be the dimensions of the Producer and the Consumer domain, respectively. Let $m \geq n$ and M be a $m \times n$ injective mapping matrix. From the independent rows of M, we construct the $n \times n$ matrix $M^{'}$ in the order they appear in M. If $M^{'}$ is an in-order transformation, then M is also an in-order transformation.*

Proof: Consider two arbitrary Consumer points $A = (x_1, x_2, ..., x_n)$ and $B = (y_1, y_2, ..., y_n)$ such that $A \prec B$. By taking out the independent rows from the matrix M in the order they appear, we obtain the matrix $M^{'}$. Since the matrix M is injective, there are exactly n independent rows and, hence, the matrix $M^{'}$ is square ($n \times n$). Suppose it respects *Property1*. Let : $(a'_1, a'_2, ..., a'_n) = M^{'} A$ and $(b'_1, b'_2, .., b'_n) = M^{'} B$. Hence $(a'_1, a'_2, ..., a'_n) \prec (b'_1, b'_2, ..., b'_n)$. Since $\prec$ is the lexicographical order, there exist $l,\ 1 \leq l < n-1$ such that $a'_1 = b'_1, \dots, a'_l = b'_l$ and $a'_{l+1} < b'_{l+1}$. Let $(a_1, a_2, .., a_m) = MA$ and $(b_1, b_2, ..., b_m) = MB$. Suppose that the $(l+1) - th$ row of the matrix $M^{'}$ is positioned as the $k - th$ row in the matrix M. Then $a_t = b_t$ for each integer t such that $1 \leq t \leq (k-1)$. This follows immediately from the fact that each $a_t, 1 \leq t \leq (k-1)$ is a linear combination of some a'_s, where $1 \leq s \leq l$. Moreover, $a_k = a'_{l+1}$ and $b_k = b'_{l+1}$. Because $a'_{l+1} < b'_{l+1}$ it follows that $MA^{'} \prec MB^{'}$. Using the same technique like in the previous proof, it follows that whatever consecutive Consumer points A and B are chosen, the corresponding Producer points MA and MB are also consecutive. □

Corollary 1 *If M_1 is an in-order matrix transformation then $M_1 M_2$ is in-order if and only if M_2 is in-order.* □

5.2.1 In-order example

Consider a P/C pair with the mapping given by mapping matrix M. Since this matrix is not square, we use *Property2* and construct a matrix $M^{'}$ from the independent rows of M using the Smith decomposition of M. Since M' satisfies *Property1*, we conclude that the mapping M is an in-order transformation.

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 \\ -5 & 1 & 0 \\ 2 & 3 & 0 \\ -1 & 6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{Prop2} \mathbf{M'} = \begin{bmatrix} 1 & 0 & 0 \\ -5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{Prop1} \text{In Order}$$

5.2.2 Out-of-order example

Consider a P/C pair with the mapping given by the non-square matrix M. As in the previous example, we extract the independent rows of matrix M using *Property2*, such that we get $M^{'}$.Since $M^{'}$ satisfies *Property1*, we conclude that M is out-of-order. This result can be easily verified by taking two arbitrary iterations points at the Consumer: $A = (2, 2, 1, 4)$ and $B = (2, 3, 1, 1)$ with $A \prec B$. Applying to these two points the mapping M, the next Producer points result: $A^{'} = (4, 8, 6, 7)$ and $B^{'} = (4, 6, 7, 7)$, such that $B^{'} \prec A^{'}$. Thus, M is indeed an out-of-order transformation.

$$\mathbf{M} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -5 & 1 & 0 & 1 \\ 2 & 3 & 0 & 0 \\ -1 & 6 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix} \xrightarrow{Prop2} \mathbf{M}' = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -5 & 1 & 0 & 1 \\ 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix} \xrightarrow{Prop1} \text{Out Of Order}$$

6 The Consumer domain influence the P/C order

The two properties presented in the previous section check necessary condition for re-ordering, i.e., there are certain cases where these conditions are not satisfied, while the Producer and the Consumer are actually in-order. Consider, for example, The mapping matrix $M = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ in the Figure 4. The matrix does not satisfy *Property1*, but from the figure we can see that the mapping clearly defines an in-order transformation.

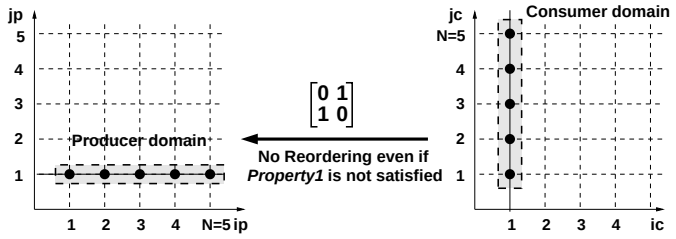

Figure 4. An in-order example, even though the mapping does not satisfy the *Property1*

When equalities are involved at the Consumer side, we cannot concluded from the mapping only whether or not the correspondent P/C pair is in-order. To arrive at an accurate decision, we have to take also into consideration specifics of the Consumer domain, in particular the equalities in the Consumer domain specification. This leads to a procedure that is to be used when equalities are involved. This procedure is based on the use of the Hermite Normal Form [4], which says that for any full row rank matrix L of size $(m \times n)$, there exists an unimodular matrix C and a lower triangle matrix H such that:

$$\begin{aligned} &L \cdot C \ = \ [H, 0], \\ &\forall (i,j) \text{ such that } i = j, \ h_{ij} > 0, \\ &\forall (i,j) \text{ such that } i > j, \ |h_{ij}| \leq h_{ii}. \end{aligned} \tag{2}$$

Matrix H has the property that $H^{-1}C$ is an integer matrix. The matrix H represents by definition an in-order transformation, as it always satisfy *Property1*. The C matrix in the HNF is not unique. Suppose that $[H, 0]$ has k zero columns. Whatever column transformation is applied to these last k columns, another unimodular matrix C' can be generated such that $L \ C' = [H, 0]$.

Consider a Consumer domain containing a number of equalities, i.e., the domain has a *lineality space*. For this case, we propose a procedure that is graphically depicted in Figure 5. It shows a Consumer domain D that lies on the hyper-plane described by the matrix L that has a lineality space. The solution consists of mapping the Consumer domain into a fully dimensional domain through a mapping matrix C^{-1} given by the HNF of L such that C respects *Property1*. According to this transformation, the equality in the Consumer domain does no longer play a reordering role, as it is constant. We denote this special mapping as C_{New}. Once we have C_{New}, we will show that if MC_{New} (See Figure 5) is an in-order transformation, it implies that the initial P/C pair is also in-order.

We will show how C_{New} can be obtained from an arbitrary matrix C, as given by the HNF of L. Therefore, instead of analyzing only the P/C mapping matrix M, we also have to take into consideration the equalities that describe the Consumer domain. This approach is expressed by the next Lemma:

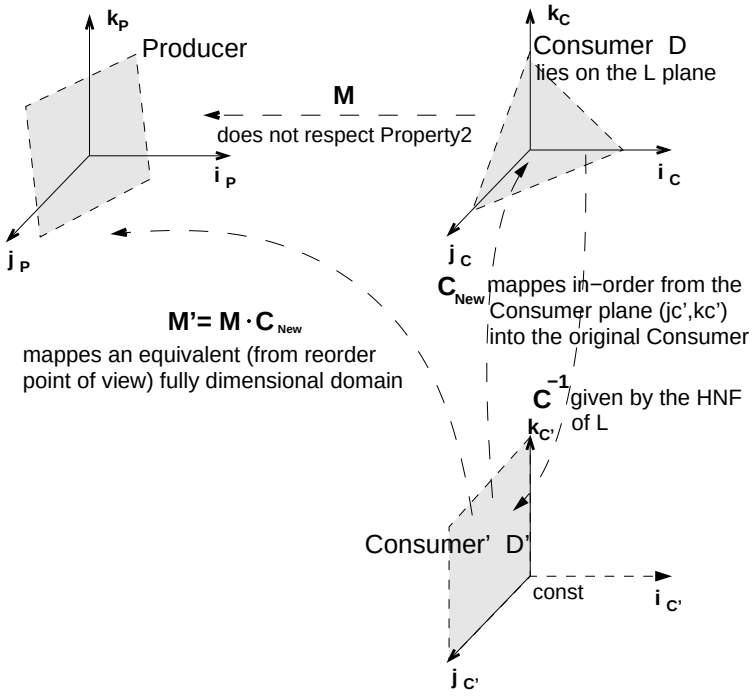

Figure 5. Out-of-order detection schema

Lemma 1 *For any arbitrary domain D described by a polytope that lies on a hyper-plane given by matrix L, there always exists an unimodular matrix that maps in-order the domain D into a fully dimensional domain D′.*

Proof: Let the equalities in the Consumer domain be described as:

$$L\, X_C = const. \tag{3}$$

where $X_C \in D$. L is of full row rank, otherwise the domain is empty as the redundant equalities are removed. Using the HNF, L can be decomposed as:

$$L = [H, 0]\, C^{-1}. \tag{4}$$

where H is a $(p \times p)$ hermite matrix and C is a $(n \times n)$ unimodular matrix. By substituting L in Equation 4, we get $[H, 0]\, C^{-1} X_C = const$. Since $C^{-1} X_C = X'_C$, we may write $[H, 0]\, X'_C = const$. Because H is bijective, independently on C, X'_C is of the form:

$$X'_C = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_p \\ x'_{p+1} \\ \vdots \\ x'_n \end{bmatrix}, \tag{5}$$

where the first p elements α are constants and the last $(n-p)$ elements x' are variable.

Because the first p elements of X'_C are constants, only the last $(n-p)$ columns of C play a role from the reordering point of view. Consider the matrix $\bar{C}$ made of the last $(n-p)$ columns of C. Because C is a full column rank matrix, $\bar{C}$ is full column rank. Let us note with C_{Indep} the matrix made of the first $(n-p)$ independent rows of $\bar{C}$. Therefore, C_{Indep} is bijective (as full row/column rank) which implies that C^{-1}_{Indep} exists. Using *Property1*, the matrix $C'_{New} = \bar{C}\, C^{-1}_{Indep}$ is an in-order transformation (C'_{New} is injective and has the first independent rows represented by an identity matrix I_{n-p}). Attaching to the matrix C'_{New} the first p columns of the original matrix C, the matrix C_{New} is unimodular and LC_{New} is a hermite form of L (because C_{New} is obtained doing simple column operations to the matrix C).

On the other hand, because of the in-order characteristics of the matrix C'_{New},then also C_{New} maps in-order from $X'_{C_{New}}$ to X_C (take into account that the first p elements of a point from $X'_{C_{New}}$ are constant). Therefore, C_{New} is a matrix that maps in-order a fully dimensional domain into. the original domain. □

Based on *Lemma1*, we define the next property:

Property 3 *If matrix M C_{New} corresponds to an in-order P/C pair, then the original P/C pair (corresponding to the matrix M) is also an in-order pair.*

Proof: Because C_{New} is bijective, there is an inverse matrix C^{-1}_{New}. Consider two arbitrary consecutive Consumer points A and B. They will be mapped in-order through the matrix C^{-1}_{New} into the Consumer points A' and B'. On the other hand, because A' and B' are consecutive, they are mapped by M C_{New} into two consecutive Producer points A'' and B''. But A'' and B'' are also the Producer images of A and B through the mapping M. Hence the P/C pair is in-order. □

6.1 Example

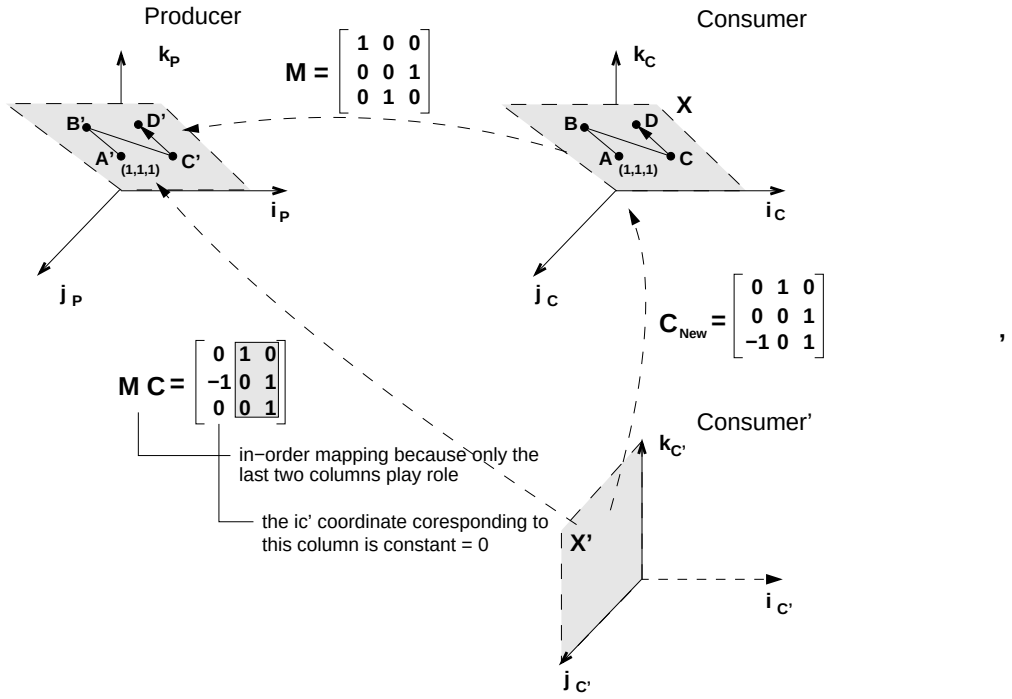

Figure 6. An in-order example

Consider the 3-dimensional Consumer domain X that lies on the plane given by $j_C - k_C = 0$ as shown in Figure 6. This plane is described by the matrix:$L = [\,0\ 1\text{-}1\,]$. Suppose the mapping matrix is

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{6}$$

According to *Property1*, the mapping represents an out-of-order transformation. However, as can be observed in the top part of Figure 6, the Consumer points $(A(1,1,1) \prec B(1,2,2) \prec C(2,1,1) \prec D(2,2,2))$ are mapped in-order through M into the Producer points $(A'(1,1,1) \prec B'(1,2,2) \prec C'(2,1,1) \prec D'(2,2,2))$. Because equalities are involved at the Consumer domain, we apply *Property3*.

Taking the HNF of L which describes the equality plain, the unimodular matrix

$$C = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{7}$$

results. Since there is just one equality involved, the rank of H is 1, and $[H, 0] = L\ C^{-1} = [1\ 0\ 0]$. According to the matrix C, this domain will be mapped (see Figure 6) into a domain of the form

$$X' = \begin{bmatrix} \text{const} \\ j_{C'} \\ k_{C'} \end{bmatrix} \tag{8}$$

Because the first coordinate in the X' domain (i.e., $i_{C'}$) is constant, only the last 2 columns of C play a role from the reordering point of view when we map from X' to X. Using *Property2*, C maps out-of-order from X' to X_C, so C is not the matrix we are looking for. But using *Lemma 1*, we can apply simple transformations on C such that we will obtain the right matrix. We take out from C the 2nd and the 3rd column (since the first column corresponds to a constant coordinate). The new matrix becomes:

$$\bar{C} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \overset{Prop2}{\longrightarrow} C_{Indep} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \longrightarrow C_{Indep}^{-1} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \overset{\bar{C} \times C_{indep}^{-1}}{\longrightarrow} C'_{New} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Now we attach to the matrix C'_{New} the first column of C and we get the matrix C_{New}, that is in fact obtained from C by doing simple matrix operations on the column that corresponds to the zero columns from the HNF of the matrix L. Therefore, by multiplying L with C_{New} we get the same hermite normal form:

$$[H, 0] = L\ C_{New} = [0\ 1\text{-}1] \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = [1\ 0\ 0]. \tag{9}$$

The difference is that now the matrix C_{New} is an in-order mapping. Now we make the product between M and C_{New} and it results that:

$$M\ C_{New} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}. \tag{10}$$

From this new matrix we remove the first column (which corresponds to the constant coordinate $i_{C'}$) and in this way we get the matrix:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \tag{11}$$

which is an in-order mapping as it satisfies *Property1*. This is precisely what we expect from observing Figure 6.

7 Hierarchical Classification Scheme

The aim is to classify correctly a P/C pair to one of the four types given in Figure 1. We do not want to rely only on the Multiplicity and Reordering tests defined in [11] as it requires solving integer linear programming systems. Instead, we want to use as much as possible the three properties defined in this paper. Since these properties are not able to give a 100% correct answer in all cases, we may still need to use the Multiplicity and Reordering test. This leads to a hierarchical classification scheme as given in Figure 7. In the classification scheme, we provide information about the P/C pair and at the bottom of the scheme, we see that the P/C pair is placed in one of the four communication types (IOM-, OOM-, IOM+, or OOM+).

In the scheme, we see the Multiplicity and Reordering tests that are given as shaded boxes, but also the *Full-ColumnRank test* and the *InOrder test*. The FullColumnRank test implements the Smith Normal Form to see if multiplicity is involved or not. When the test determines that no multiplicity is involved, i.e., the matrix is not injective, it might still be that multiplicity is involved. Therefore, we use the Multiplicity test to be sure. The other way around, when the FullColumnRank test indicates that no multiplicity is involved, it is guaranteed to be the

case. The InOrder test uses the 3 properties defined in this paper to see if reordering is involved or not. This test first checks if the Consumer has an lineality space. If that is the case, *Property3* is applied (which implies the use of *Property2* and *Property1*). If no lineality space is involved, the tests checks if the matrix is square. If not, it uses *Property2* and *Property1* to determine if reordering is involved. If the matrix is square, it uses only *Property1* to determine if reordering is involved. There are cases for which the InOrder test indicates that the mapping is an out-of-order transformation, while it is not. Therefore, we apply the Reordering test. The other way around, when the InOrder test determines that the mapping is in-order, it is guaranteed to be the case.

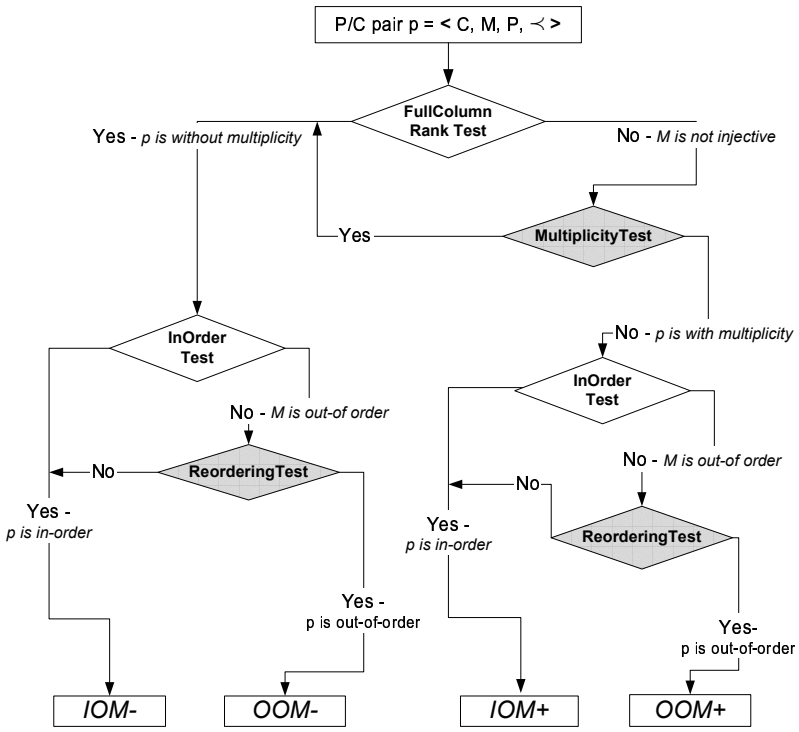

Figure 7. The Hierarchical Classification Scheme

7.1 Experimental Results

In Table 1, we present the network configurations for 8 image and signal processing algorithms. These are the same example as used in [11]. In the table, we show how many times a Reordering Test (RT) and Multiplicity Test (MT) is done in the hierarchical classification scheme. The values given between brackets are the number of tests performed when only the RT and MT are used as done in [11]. The classification that uses only MT and RT, solved 4516 systems for the Reordering test and 1780 systems for the Multiplicity test for the 8 examples. Using the hierarchical classification scheme presented in Figure 7, which exploits the properties presented in this paper, only 237 systems are solved for the Reordering test and 102 systems for the Multiplicity test. This means that the hierarchical classification scheme reduces the number of system to solve by 95%. In only 5% integer linear programming is still used. In 95% of the cases, the three properties presented are able to correctly classify a P/C pair. This is a big reduction in compute time and memory requirements.

It is interesting to mention that Table 1 clearly shows that most occurring type is IOM- (80%) and the least occurring type is OOM+ (1%). This is important from an implementation point of view, as the IOM- requires only a simple FIFO buffer. The OOM+ requires also re-ordering memory and a controller, but hardly appears in networks. The second most occurring type is IOM+ (10%) which is also nice, as only a FIFO buffer is needed with some simple additional control to keep track of the life time of a token. Together with type IOM-, we can say that in 90% of the cases, a FIFO buffer can indeed be used to linearize a static array in a Matlab program. In the other cases, the ELM is needed to linearize a static array.

Algorithm	P/C pairs	RT	MT	IOM-	IOM+	OOM-	OOM+
QR-Decomp	12	0 (73)	0 (30)	12	0	0	0
SVD	118	171 (1283)	51 (565)	84	4	30	0
Faddeev	28	4 (205)	7 (78)	24	3	1	0
Gauss-Elimin	11	17 (17)	6 (6)	7	0	1	3
DigBeamFormer	98	18 (408)	4 (196)	88	4	6	0
Motion Estim	98	0 (882)	0 (294)	98	0	0	0
M-JPEG	50	24 (178)	34 (93)	33	17	0	0
Stereo Vision	173	3 (1470)	0 (518)	172	0	1	0

Table 1. Classifying P/C pairs in real-life examples to the four communication types using the Hierarchical Classification Scheme.

8 Conclusion

Within Compaan, static arrays found in the original Matlab are replaced with a distributed communication structure that takes care of the interprocess communication. We have shown that four different types of communication exists, each exhibiting different requirements when realizing them in hardware of software. As a consequence, we need a technique to classify the interprocess communication to one of the four types. For this purpose we have already developed the Multiplicity and the Reordering test, but they are memory and time inefficient . In this paper, we presented additional tests based on polynomial algorithms like Smith and Hermite normal forms of an integer matrix [2]. By combining these tests with the original Multiplicity and Reordering test, we defined a hierarchical classification scheme that exactly classifies the interprocess communication. Hence, in only 5% of the cases to classify, we still rely on integer linear programming; in the remaining 95%, the tests presented in this paper are able to classify a case correctly. The hierarchical classification scheme is implemented within Compaan.

References

[1] Philippe Clauss. Counting solutions to linear and nonlinear constraints through ehrhart polynomials: Applications to analyse and transform scientific programs. In *10th International Conference on Supercomputing, Philadelphia*, May 1996.

[2] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. In *SIAM J. Comput., 8 4*, pages 499–507, 1979.

[3] Bart Kienhuis, Edwin Rypkema, and Ed Deprettere. Compaan: Deriving process networks from matlab for embedded signal processing architectures. In *Proceedings of the 8th International Workshop on Hardware/Software Codesign (CODES)*, San Diego, USA, May 2000.

[4] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-interscience, 1988.

[5] http://www.picochip.com.

[6] Edwin Rijpkema. *From Piecewise Regular Algorithms to Dataflow Architectures*. PhD thesis, Delft University of Technology, 2001.

[7] Todor Stefanov, Claudiu Zissulescu, Alexandru Turjan, Bart Kienhuis, and Ed Deprettere. System Design using Kahn Process Networks: The compaan/laura approach. In *Proceedings of DATE2004*, Paris, France, Feb 16 – 20 2004.

[8] Paul Stravers and Jan Hoogerbrugge. Homogeneous multiprocessoring and the future of silicon design paradigms. In *Proceedings of the Int. Symposium on VLSI Technology, Systems, and Applications*, April 2001.

[9] Alexandru Turjan and Bart Kienhuis. Storage management in process networks using the lexicographically maximal preimage. In *Proceedings of the IEEE 14th Int. Conf. on Application-specific Systems, Architectures and Processors (ASAP'03)*, The Hague, The Netherlands, June 24 –26 2003.

[10] Alexandru Turjan, Bart Kienhuis, and Ed Deprettere. A compile time based approach for solving out-of-order communication in Kahn Process Networks. In *Proceedings of the IEEE 13th Int. Conf. on Application-specific Systems, Architectures and Processors (ASAP'02)*, San Jose, USA, July 17 – 19 2002.

[11] Alexandru Turjan, Bart Kienhuis, and Ed Deprettere. An Integer Linear Programming Approach to Classify Communication in Process Networks. In *8th International Workshop on Software and Compilers for Embedded Systems (SCOPES)*, Amsterdam, September 2–3 2004.

[12] http://www.xilinx.com.